Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліні “ Об’єктно - орієнтоване програмування”

КАСА АЕРОФЛОТУ

Керівник , асистент Ляпота В.М.

Студент гр. ПІ-15-2 Джафаров Е.Е.

Комісія: проф. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дудар З.В.

проф. каф. ПІ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бондарєв В.М.

Харків 2016

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра: ***Програмної інженерії***

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

Спеціальність: ***Програмна інженерія***

Курс 1 .Група *ПІ-15-2* . Семестр *2* .

***ЗАВДАННЯ***

***на курсовий проект студента***

***Джафарова Еміля Ельшан огли***

1. Тема проекту:*Програмна система під назвою «Каса аерофлоту»*

*Расписание: номер рейса, маршрут, пункты промежуточной посадки, время отправления, дни полета. Количество свободных мест на каждом рейсе. Выбор ближайшего рейса до заданного пункта (при наличии свободных мест), оформление заданного числа билетов по согласованию с пассажиром (с уменьшением числа свободных мест), оформление посадочной ведомости* .

1. Термін здачі студентом закінченого проекту: ***“28” - травня - 2016 р***.
2. Вихідні дані до проекту: *Специфікація програми, методичні вказівки до виконання курсової роботи.*

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 1-03-2016 р. |
| 2 | Формулювання вимог до програми | 5-03-2016 – 10-03-2016 р. |
| 3 | Розробка підсистеми зберігання та пошуку рейсів. | 10-03-2016 – 15-03-2016 р. |
| 4 | Розробка підсистеми зберігання та пошуку пасажирів. | 16-03-2016 – 25-03-2016 р. |
| 5 | Розробка функцій бронювання квитків | 28-03-2016 – 3-04-2016 р. |
| 6 | Розробка функцій зберігання та завантаження даних | 5-04-2016 – 10-04-2016 р. |
| 7 | Тестування і доопрацювання розробленої програмної системи. | 15-04-2016 – 29-04-2016 р. |
| 8 | Оформлення пояснювальної записки, додатків, графічного матеріалу | 15-05-2016 – 27-05-2016 р. |
| 9 | Захист | 28-05-2016 р. |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ляпота В. М.

(Прізвище, Ім'я, По батькові)

« 1 » березня 2016 р.

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 38 с., 9 рис., 6 табл.,

1 додаток, 5 джерел.

Метою роботи є розробка програми “Каса Аерофоту” на засадах об'єктно-орієнтованого програмування.

Методи розробки базуються на використанні середи розробки Microsoft Visual Studio 2015, Windows Forms, платформи .NET Framework 4.5, мови програмування C#.

В результаті отримана програма під назвою “Каса аерофлоту” для роботи з базою рейсів, з інформаційною базою даних, чию роль виконують текстові файли. Програма дозволяє зберігати розклад рейсів, відомості про кожний рейс, такі як: номер, кількість вільних місць, маршрут, дата та час вильоту, дата та час прибуття, пункти проміжної посадки. Є можливість додавати, редагувати, видаляти рейси та бронювати квитки на обраний рейс.

ПРОГРАМА, ОБ’ЕКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, КЛАС, МОВА ПРОГРАМУВАННЯ C#, ПЛАТФОРМА .NET, КАСА АЕРОФЛОТУ, РЕЙС, ВІДОМІСТЬ, БРОНЮВАННЯ.

ЗМІСТ

ВСТУП ..6

1 СПЕЦИФІКАЦІЯ ПРОГРАМИ ..7

1.1 Вимоги до програми, основні її функції ..7

1.2 Інтерфейс програми ..8

2 ПРОЕКТНА СПЕЦИФІКАЦІЯ. ..9

2.1 Проектна модель програми ..9

2.2 Реалізація функцій програми 12

3 ІНСТРУКЦІЯ КОРИСТУВАЧА 18

ВИСНОВКИ 23

ПЕРЕЛІК ПОСИЛАНЬ.. 24

Додаток А – Код програми.. 25

ВСТУП

Зараз, у ХХІ столітті, авіа перельоти стали невід’ємною складовою нашого життя. Кожну годину з різних кутів планети відлітають сотні літаків з тисячами пасажирів на борту. Проте, мабуть, ніхто з них не замислювався над тим, які ж процеси відбуваються при звичайному бронюванні квитків і на скільки вони складні.

Каса аерофлоту - це дуже складний механізм, яким було б дуже складно керувати без комп’ютерних технологій. Адже база данних авіакомпаній величезна і звичайному касиру не можливо пам’ятати всі відомості про всі рейси. Тому перед програмістами постає задача розробити таку програмну систему, яку з легкістю можна було б опанувати без попередньої підготовки. Водночас ця система повина бути досить швидкою та надійною.

Темою цього курсового проекту є розробка програмної системи для каси аерофлоту. Основні задачі, які потрібно було виконати: розробити зрозумілий інтерфейс користувача та спроектувати базу данних. Базою даних служать файли формату \*txt. Інтерфейс користувача побудований досить інтуїтивно, а отже програма є простою у використанні.

За основу розробки програми було узято об’єктно-орієнтований підхід, завдяки якому програму досить легко супроводжувати під час її експлуатації. У якості головних інструментів розробки було використано середовище Microsoft Visual Studio 2015 та мову програмування C#.

1. СПЕЦИФІКАЦІЯ ПРОГРАМИ

1.1 Вимоги до програми, основні її функції

В даній курсовій роботі буде розроблено емулятор каси аерофлоту, в якому буде реалізовано більшість функцій, що є в справжній касі аерофлоту. Кінцева програма буде здійснювати пошук рейсів за заданим маршрутом, проводити операції з базою даних рейсів, а також сформовувати посадкову відомість для кожного рейсу.

Для розробки проекту було обрано середовище Microsoft Visual Studio 2015, яке найбільш відповідає сучасним вимогам моделювання програмного забезпечення і володіє зручним інтерфейсом.

У якості бази даних будуть використовуватися файли формату \*. txt, до кожного рейсу з бази будуть прив'язані під-бази даних пасажирів. Кожен рейс має свій унікальний номер. Завдяки цьому і здійснена реєстрація пасажирів на обраний рейс.

Програма складається з п’ятьох вікон:

* + - 1. головне вікно - табло с розкладом рейсів;
      2. вікно для редагування;
      3. вікно для отримання усієї необхідної інформації про рейс;
      4. вікно для створення нового рейсу;
      5. вікно для бронювання квитків.

У головному вікні реалізовано табло, щоб користувач міг побачити список рейсів та інформацію про них. А також мав змогу видалити та редагувати обрані рейси за допомогою відповідних клавіш. У головному меню програми користувач має можливість перейти до бронювання квитків або створити новий рейс. Інтерфейс програми є досить інтуїтивним та зрозумілим, що є не менш важливим.

* 1. Інтерфейс програми

Користувацький інтерфейс складається з п’ятьох вікон. Для почаку розглянемо детальніше головне вікно програми. Воно складається з таблиці розкладу рейсів, меню програми та клавіш для більш зручної навігації. Меню у свою чергу складається з таких трьох пунктів: «бронювання квитків», «створення нового рейсу», «вихід з програми». Таблиця з розкладом

представляє всі головні відомості про рейси: номер, кількість вільних місць, маршрут, час та дата відправлення, час та дата прибуття, пункти проміжної посадки. Знизу, під розкладом рейсів, є декілька важливих клавіш: «Видалити», «Редагувати», «Відомість» та «Вийти». Завдяки цим клавішам дуже зручно та легко вибрати у розкладі якийсь рейс, щоб видалити, редагувати, отримати всі необхідні дані про нього.

Вікно для створення нового рейса має усі необхідні поля для його створення: номер, маршрут, час та дата відправлення, час та дата прибуття, загальна кількість місць та пункти проміжної посадки. Для внесення до бази нового рейсу користувачеві потрібно ввести коректні дані. Якщо користувач ввів некоректні дані, програма відреагує і вкаже на подальші дії.

Вікно для редагування рейсів відображає основні відомості про рейс. Деякі з них неможливо змінити, а деякі – можливо. Наприклад, користувач не може поміняти основний маршрут рейсу. Можна змінювати час та дату відправлення і прибуття, але так, щоб вони були більше поточного часу. Також є можливість додати якийсь новий пункт проміжної посадки.

Вікно для перегляду посадкової відомості відображає номер рейсу, кількість вільних місць, загальну кількість місць, маршрут, час та дату відправлення, час та дату прибуття, пункти проміжної посадки, а також список пасажирів, зареєстрованих на обраний рейс.

Вікно для бронювання квитків має поля для введення ім’я, прізвища та по батькові пасажира й поля для введення маршруту. Саме у цьому вікні відбувається пошук необхідного рейсу та реєстрація пасажира на обраний.

2 ПРОЕКТНА СПЕЦИФІКАЦІЯ

2.1 Проектна модель програми

У графічному інтерфейсі цієї програми містяться декілька форм заповнення даних, отже в проекті присутні спеціальні допоміжні функції, які зчитують дані, перевіряють їх коректність і зберігають.

Перший компонент – зчитування даних, відбувається при завантаженні програми. В ході виконання цього компонента відбувається зчитування з бази даних і заповнення головного списку рейсів, з яким надалі взаємодіють всі інші компоненти програми.

Компонент роботи з даними складається з функцій і подій, які безпосередньо впливають і змінюють головний список рейсів і його властивості.   
 Компонент перевірки даних складається з декількох функцій, які перевіряють різні вхідні дані на правильність і доречність. У разі успішного завершення тієї чи іншої функції програма продовжить свою роботу і будуть виконані ті команди, які були запрошені користувачем, а в разі введення некоректних даних, буде опрацьовано відповідний виняток, з якого потім програма сформує вікно попередження з подальшими підказками для користувача.

Компонент збереження в базу даних оновлює і записує зміни в базу даних, використовуючи відповідні бібліотеки класів.

В ООП головним елементом є клас, що включає безліч об'єктів з однаковими властивостями, операціями і відносинами. Клас має внутрішнє (реалізацію) і зовнішнє подання (інтерфейс).

У даній програмі реалізовані класи Flight(рейс), FlightCollection(список усіх рейсів), Passenger(пасажир), CollectionOfPassengers(список пасажирів на рейсі).

На рисунку 2.1 продемонстрована схема класів.

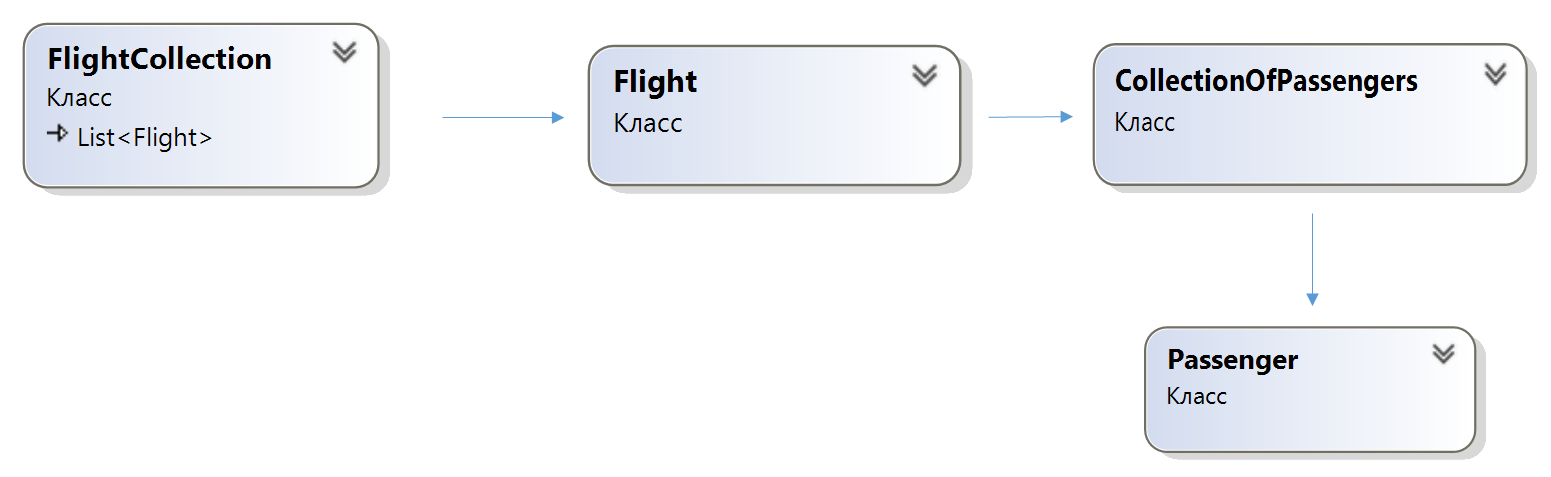


Рисунок 2.1 – Об’єктна специфікація програми

Клас «Flight» є одним з найголовніших у програмі і саме через його властивості відбувається взаємодія з іншими частинами програми. Клас містить всі відомості, які властиві звичайному рейсу, а саме: номер рейсу, маршрут, місця проміжної зупинки, кількість місць та часові рамки (відправлення і прибуття).



Рисунок 2.2 – Властивості та методи класу «Flight»

Клас «FlightCollection» містить усі можливі рейси. Саме в об’єкт цього класу дані считуються з бази. Також, для зручної роботи цей клас наслідує клас List. На рисунку 2.3 зображені методи цього класу.



Рисунок 2.3 – Методи класу «FlightCollection»

Клас «Passenger» є класом, об’єкти якого зберігають дані про пасажира (ім’я, прізвище, по батькові). На рисунку 2.4 представлені властивості та методи цього класу.



Рисунок 2.4 – Властивості та методи класу «Passenger»

2.2 Реалізація функцій програми

При старті програми відбувається підключення до бази даних. Сам код підключення реалізовано в конструкторі класу «FlightCollection».

У таблиці 2.1 приведено код, який виконується при старті програми.

|  |
| --- |
| public void OnLoadFuction()  {  foreach (DataGridViewColumn colum in FlightTable.Columns)  {  colum.SortMode = DataGridViewColumnSortMode.NotSortable;  }  try  {  FlightTable.Rows.Clear();  FlightCollection list = new FlightCollection("Flights.txt");  list.SortingByDays();  for (int i = list.Count - 1; i >= 0; i--)  {  FlightTable.Rows.Add(  list[i].Number,  list[i].FreePlaces,  list[i].DepartureFrom,  list[i].ArrivalIn,  list[i].Departure.TimeOfDay.ToString("hh':'mm"),  list[i].Departure.Date.ToString().Split(' ')[0],  list[i].Arrival.TimeOfDay.ToString("hh':'mm"),  list[i].Arrival.Date.ToString().Split(' ')[0],  Program.GetStopStations(list[i].StopStation)  );  }  if (FlightTable1.Rows.Count == 0)  {  button3DeleteFlight.Visible = false;  button5InfomOfFlight.Visible = false;  button2EditFlight.Visible = false;  }  else  {  button3DeleteFlight.Visible = true;  button5InfomOfFlight.Visible = true;  button2EditFlight.Visible = true;  }  }  catch  {  this.Show();  MessageBox.Show("Ошибка в базе данных!", "Оповещение");  this.Close();  }  } |

Таблиця 2.1 – Код при старті програми

Для створення нового рейсу користувачу потрібно у головному меню програми вибрати пункт «Додати рейс». Після чого на екрані з’явиться вікно з відповідними полями для створення нового рейсу. У таблиці 2.2 представлений код, який виконується при додаванні нового рейсу до бази.

|  |
| --- |
| private void button1\_Click(object sender, EventArgs e)  {  try  {  string Number = Program.RemoveSpaces(textBox1Number.Text);  string depDay = dateTimePicker1DepartureDay.Value.Day.ToString()+"."  + dateTimePicker1DepartureDay.Value.Month.ToString() + "."  + dateTimePicker1DepartureDay.Value.Year.ToString();  string arrDay = dateTimePicker2ArrivalDay.Value.Day.ToString()+"."  + dateTimePicker2ArrivalDay.Value.Month.ToString() + "."  + dateTimePicker2ArrivalDay.Value.Year.ToString();  string depTime = numericUpDown4DepHours.Value + ":" +  numericUpDown5DepMinutes.Value;  string arrTime = numericUpDown1ArrHours.Value + ":" +  numericUpDown2ArrMinutes.Value;  string[]StopStations =  Program.RemoveSpaces(textBox1StopStations.Text).Split(',');  string DepartureFrom = Program.RemoveSpaces(textBox1DepartureFrom.Text);  string ArrivalIn = Program.RemoveSpaces(textBox1ArrivalIn.Text);  int countPlaces = Convert.ToInt32(numericUpDown6countPlaces.Value);  Flight newFlight;  FlightCollection file = new FlightCollection("Flights.txt");  Flight Current = file.FindByNumber(Convert.ToInt32(Number));  if (Convert.ToDateTime(depDay + " " + depTime) <= DateTime.Now)  {  MessageBox.Show("Время и дата отправления должны быть больше  текущих.", "Оповещение");  return;  }  if (DepartureFrom == "" || ArrivalIn == "")  {  MessageBox.Show("Укажите маршрут!", "Оповещение");  return;  }  if (Current == null)  {  newFlight = new Flight(Number, DepartureFrom, ArrivalIn, depTime, depDay, arrTime, arrDay, countPlaces, StopStations);  file.AddFlightAndWriteInFile(newFlight);  }  else  {  MessageBox.Show("Рейс с таким номером уже  существует.","Оповещение");  return;  }  this.Close();  Program.MainForm.OnLoadFuction();  }  catch (FormatException)  {  MessageBox.Show("Проверьте правильность введённых  данных.","Оповещение");  }  catch(ArithmeticException)  {  MessageBox.Show("Номер рейса слишком длинный.", "Оповещение");  }  catch (Exception my)  {  MessageBox.Show(my.Message, "Оповещение");  }  } |

Таблиця 2.2 – Додавання нового рейсу

Додання сформованого рейсу до бази відбувається безпосередньо у класі «FlightCollection»,а саме за допомогою метода «АddFlightAndWriteInFile()». Код цього метода представлений у таблиці 2.3.

|  |
| --- |
| public void AddFlightAndWriteInFile(Flight obj)  {  this.Add(obj);  using (StreamWriter wr = new StreamWriter(Path, true))  {  wr.WriteLine(obj.ToString());  wr.Close();  }  } |

Таблиця 2.3 – метод для внесення нового рейсу у базу

Для редагування рейса користувачеві потрібно обрати необхідний рейс у розкладі та натиснути на клавішу «Редагувати». Після цього на екрані з’явиться вікно з можливими полями для редагування, тобто не всі дані рейса можуть бути змінені. У таблиці 2.4 представлений код, який виконується при підтвердженні редагування, тобто при натсканні на клавішу «Готово».

|  |
| --- |
| private void button1Ok\_Click(object sender, EventArgs e)  {  try  {  FlightCollection File = new FlightCollection("Flights.txt");  Flight Current = File.FindByNumber(Number);  string depDay = dateTimePicker1DepartureDay.Value.Day.ToString() + "." + dateTimePicker1DepartureDay.Value.Month.ToString() + "."  + dateTimePicker1DepartureDay.Value.Year.ToString();  string arrDay = dateTimePicker2arrivalDay.Value.Day.ToString() + "."+ dateTimePicker2arrivalDay.Value.Month.ToString() + "."  + dateTimePicker2arrivalDay.Value.Year.ToString();  string depTime = numericUpDown1DepHour.Value + ":" + numericUpDown2DepMinutes.Value;  string arrTime = numericUpDown1ArrHours.Value + ":" + numericUpDown1ArrMinutes.Value;  string StopStation = Program.RemoveSpaces(textBox3stopstat.Text);  if (StopStation != "")  {  Current.AddStopStation(StopStation);  }  Current.Departure = Convert.ToDateTime(depDay + " " + depTime);  Current.Arrival = Convert.ToDateTime(arrDay + " " + arrTime);  File.ClearFile();  File.WriteInFile();  this.Close();  Program.MainForm.OnLoadFuction();  }  catch (Exception myexp)  {  MessageBox.Show(myexp.Message, "Оповещение");  }  } |

Таблиця 2.4 – Редагування рейса

Для реєстрації пасажира на відповідний рейс потрібно відкрити головне меню програми та обрати пункт «Бронювання квитків». Після цього відкриється вікно, у якому потрібно ввести реєстраційні дані (ім’я, прізвище, по батькові) та ввести необхідний маршрут. За згодою пасажира можна забронювати декілька квитків. У таблиці 2.5 наведено фрагмент коду, який виконується при реєстрації пасажара.

|  |
| --- |
| private void button2Ok\_Click(object sender, EventArgs e)  {  try  {  string Name = textBox1Name.Text;  string Surname = textBox2Surname.Text;  string Middlename = textBox3Middlename.Text;  string num = FlightTable.CurrentRow.Cells[0].Value.ToString();  decimal count = numericUpDown1Count.Value;  Passenger user = new Passenger(Name, Surname, Middlename);  FlightCollection arr = new FlightCollection("Flights.txt");  Flight Current = arr.FindByNumber(Convert.ToInt32(num));  Current.AddPassenger(user, count);    MessageBox.Show("Пассажир добавлен!", "Оповещение");  this.Close();  Program.MainForm.OnLoadFuction();  }  catch (NullReferenceException)  {  MessageBox.Show("Не выбрано ни одного рейса!", "Оповещение");  }  catch (Exception my)  {  MessageBox.Show(my.Message, "Оповещение");  }  } |

Таблиця 2.5 – Реєстрація пасажира

Щоб побачити усю інформацію стосовно рейсу необхідно вибрати потрібний у розкладі рейс і натиснути на клавішу «Відомість». Після чого відкриється нове вікно з усією інформацією: номер рейсу, кількість вільних місць, загальна кількість місць, час та дата відправлення, час та дата прибуття, пункти проміжної посадки. Також буде показаний список пасажирів, зареєстрованих на цей рейс.

У таблиці 2.6 представлений фрагмент коду, який виконується при натисканні на клавішу «Відомість».

|  |
| --- |
| private void Info\_Load(object sender, EventArgs e)  {  try  {  foreach (DataGridViewColumn colum in ListOfPassengers.Columns)  {  colum.SortMode = DataGridViewColumnSortMode.NotSortable;  }  FlightCollection file = new FlightCollection("Flights.txt");  Flight Current = file.FindByNumber(Number);  List<string> rez = new List<string>();  rez.Add("Номер: " + Current.Number);  rez.Add("Откуда отправляется: " + Current.DepartureFrom);  rez.Add("Куда направляется: " + Current.ArrivalIn);  rez.Add("Дата отправления: " +  Current.Departure.Date.ToString().Split(' ')[0]);  rez.Add("Время отправления: " +  Current.Departure.TimeOfDay.ToString("hh':'mm"));  rez.Add("Дата прибытия: " +  Current.Arrival.Date.ToString().Split(' ')[0]);  rez.Add("Время прибытия: " +  Current.Arrival.TimeOfDay.ToString("hh':'mm"));  rez.Add("Количество посадочных мест: " + Current.CountPlaces);  rez.Add("Свободные места: " + Current.FreePlaces);  rez.Add("Пункты промежуточной посадки: " +  Program.GetStopStations(Current.StopStation));  listBox1info.DataSource = rez;  for (int i = 0; i < Current.ListOfPassengers.Count; i++)  {  ListOfPassengers.Rows.Add(Current.ListOfPassengers[i].Name,  Current.ListOfPassengers[i].Surname,  Current.ListOfPassengers[i].MiddleName);  }  }  catch(NullReferenceException)  {  MessageBox.Show("Ошибка: время отправления этого рейса стало  меньше текущего.", "Оповещение");  this.Close();  Program.MainForm.OnLoadFuction();  }  } |

Таблиця 2.6 – Посадкова відомість

1. ІНСТРУКЦІЯ КОРИСТУВАЧА

При старті програми користувач бачить на екрані головне вікно, яке відображає табло з усіма рейсами, час відправлення яких ще не настав. На рисунку 3.1 представлено зображення стартового вікна програми.

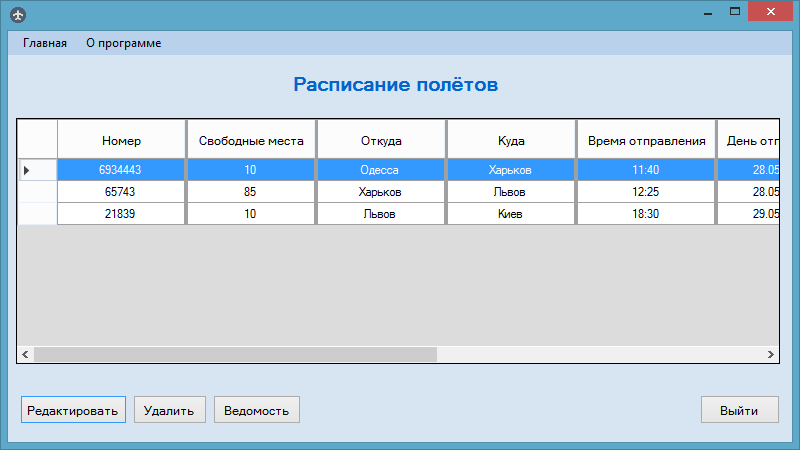


Рисунок 3.1 – Головне вікно програми

Під таблом з розкладом рейсів присутні декілька клавіш, завдяки яким дуже легко одразу почати роботу з основними функціями програми. Якщо ж на табло не буде жодного рейса, цих клавіш користувач не побачить. Це, передусім, зроблено для того, щоб передбачити можливі помилки у програмі.

Стартове вікно програми є достатньо інтуїтивним, що значно полегшує роботу з нею.

Для редагування рейсу потрібно натиснути на відповідну клавішу. На рисунку 3.2 представлено зображення вікна редагування. Завдяки тому, що у класі форми для редагування зберігається номер рейсу, інформацію про який потрібно змінити, у вікні відображається поточна інформація про рейс. Це, передусім, зроблено для того, щоб надати користувачеві найбільш зручні умови для редагування.

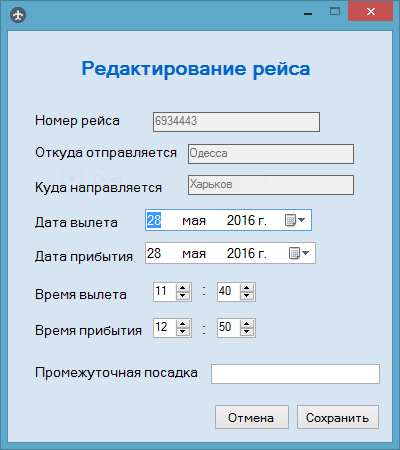


Рисунок 3.2 – Редагування рейсу

Для того, щоб переглянути всю інформацію про рейс необхідно вибрати потрібний та натиснути на клавішу «Відомість». На рисунку 3.3 представлено зображення вікна, яке містить усі дані про рейс, включаючи список зареєстрованих пасажирів.

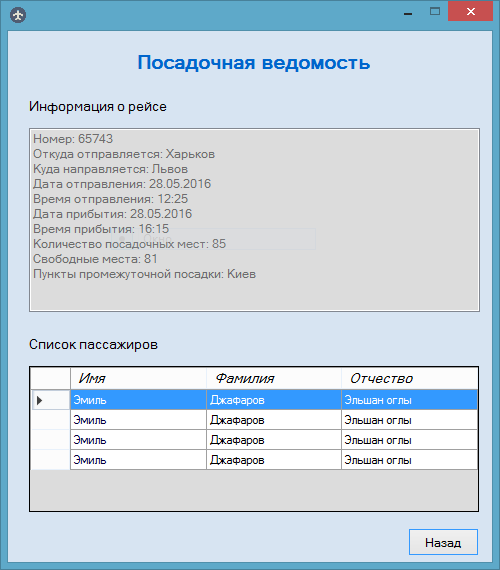


Рисунок 3.3 – Посадкова відомість

У головному меню програми є можливість додати новий рейс. Щоб це зробити потрібно натиснути на пункт меню «Головна» та обрати «Додавання нового рейсу». Після цього користувач побачить нове вікно, яке містить багато полів для введення даних, якими володіють звичайні рейси. Якщо користувач введе коректні дані, рейс буде одразу записан до бази і з ним можна буде проводити вищезазначені операції. В противному випадку програма буде видавати повідомлення, яких потрібно дотримуватись для успішного додавання рейсу. На рисунку 3.4 представлено зображення вікна для додавання нового рейсу.

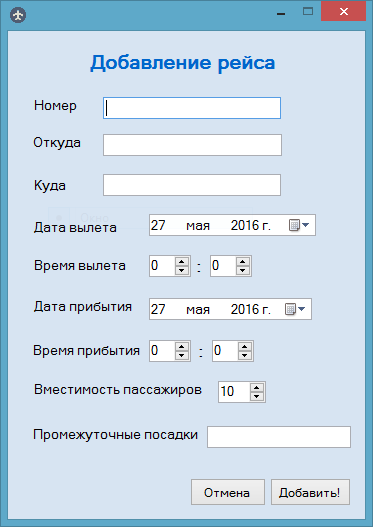


Рисунок 3.4 – Додавання нового рейсу

Мабуть, найголовніша функція цієї курсової роботи полягає саме у реєстрації пасажира на обраний рейс. Для того, щоб це зробити, користувач повинен обрати у головному меню пункт «Бронювання квитків». Після цього буде відкрито вікно з необхідними полями для реєстрації: ім’я, прізвище, по батькові пасажира та маршрут. Є можливість бронювання декількох квитків на пасажира за його згодою.

На рисунку 3.5 представлено зображення вікна для реєстрації пасажира.

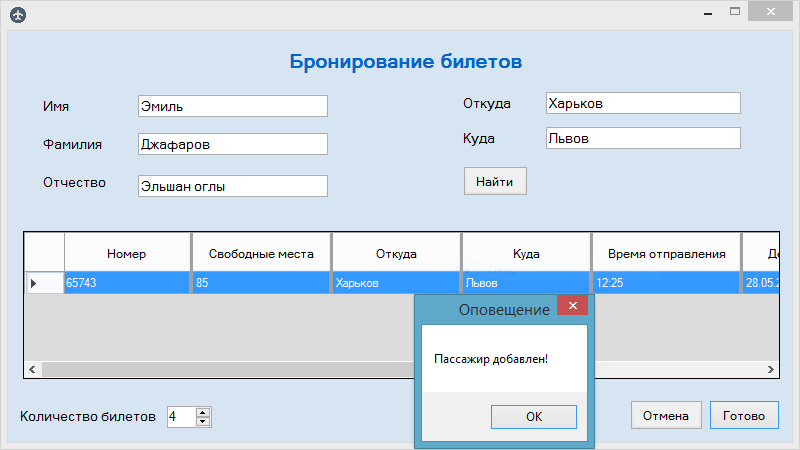


Рисунок 3.5 – Реєстрація пасажира

ВИСНОВКИ

Результатом виконання даної курсової роботи стала програма – емулятор «Каса аерофлоту». Дана програма багато в чому нагадує справжню касу аерофлоту і може виконувати ті ж функції.

У проекті представлений об'єктно-орієнтований підхід в проектуванні програмного забезпечення інформаційного плану, що дає можливість на ранніх етапах розробки врахувати всі нюанси майбутньої програми, необхідний набір функцій, склад і структуру баз даних, що надалі виключає необхідність переробки вже написаних компонентів програми.

У програмі є простий і зрозумілий інтерфейс, так що повністю розібратися у функціоналі програми не складе труднощів навіть для недосвідченого користувача. Цю програмну систему розроблено для полегшення роботи касира.

Плани на майбутнє: домогтися повної автоматизації, тобто провести ряд удосконалень, які дадуть можливість отримувати дані з сервера, фільтрувати ці дані, на основі цих даних формувати завершені рейси і планувати майбутні, підключити в проект бази даних, наприклад MongoDB.

ПЕРЕЛІК ПОСИЛАНЬ

1. Страуструп Б. Язык программирования С++. Специальное издание.-М., СПб.:”Издательство БИНОМ” – „Невский Диалект”, 2001г.-999 с.

2. Дейтел Х., Дейтел П. Как программировать на С++. Третье издание.-М.:ЗАО „Издательство БИНОМ”, 2001г.-1152с.

3. Эндрю Троельсен. C# и платформа .NET. Библиотека программиста. – СПб.: Питер, 2005. – 796 с.

4. Бондарев В.М., Объектно-ориентированное программирование на С#. Учебное пособие. – Харьков: СМИТ, 2009г. – 224 с.

5. Бондарев В.М., Программирование на С++. Учебное пособие. – Харьков: СМИТ, 2004г. – 294 с.

Додаток А – Код програми

Клас Passenger.

public class Passenger

{

public Passenger(string name, string surname, string middlename)

{

if (String.IsNullOrWhiteSpace(name) || String.IsNullOrWhiteSpace(surname) || String.IsNullOrWhiteSpace(middlename))

{

throw new Exception("Укажите регистрационные данные!");

}

Name = Program.RemoveSpaces(name);

Surname = Program.RemoveSpaces(surname);

MiddleName = Program.RemoveSpaces(middlename);

}

public string Name { get; set; }

public string Surname { get; set; }

public string MiddleName { get; set; }

public override string ToString()

{

return string.Format("{0}\_{1}\_{2}", Name, Surname, MiddleName);

}

public void WriteInFile(int num)

{

using (StreamWriter wr = new StreamWriter("Passengers\\" + num + "Passengers.txt", true))

{

wr.WriteLine(ToString());

wr.Close();

}

}

}

Клас CollectionOfPassengers.

public class CollectionOfPassengers

{

public Passenger[] passengers;

public int Count { get; private set; }

public CollectionOfPassengers(int k)

{

passengers = new Passenger[k];

Count = 0;

}

public Passenger this[int index]

{

get

{

return passengers[index];

}

set

{

passengers[index] = value;

}

}

public void AddPassenger(Passenger user)

{

if (Count < passengers.Length)

{

passengers[Count] = user;

Count++;

}

}

}

Клас Flight.

public class Flight

{

private DateTime \_arrival; // скрытые поля для отображения отправления и прибытия

private DateTime \_departure;

public Flight(string num, string dep, string arrival,string deptime, string depday,string arrtime, string arrday,int countPlace, params string[] mas)

{

Number = Convert.ToInt32(num);

DepartureFrom = Program.RemoveSpaces(dep);

ArrivalIn = Program.RemoveSpaces(arrival);

\_departure = Convert.ToDateTime(depday + " " + deptime);

\_arrival = Convert.ToDateTime(arrday + " " + arrtime);

if (DepartureFrom == "" || ArrivalIn == "")

{

throw new Exception("Укажите полный маршрут!");

}

if (\_departure >= \_arrival)

{

throw new Exception("Время и дата прибытия должны быть больше времени и даты отправления!");

}

StopStation = mas;

CountPlaces = countPlace;

ReadFromFile();

}

public int Number { get; set; } // Номер рейса

public string DepartureFrom { get; set; } // Откуда отправляется

public string ArrivalIn { get; set; } // Куда направляется

public CollectionOfPassengers ListOfPassengers { get; set; } // Список пассажиров, зарегистрированных на рейс

public DateTime Departure // Время и день вылета

{

get

{

return \_departure;

}

set

{

if (value < DateTime.Now)

{

throw new Exception("Время и дата отравления должны быть больше текущих!");

}

else

\_departure = value;

}

}

public DateTime Arrival // Время и день прибытия

{

get

{

return \_arrival;

}

set

{

if (value <= Departure)

{

throw new Exception("Время и дата прибытия должны быть больше времени и даты отправления!");

}

else

\_arrival = value;

}

}

public int CountPlaces { get; set; } // Общее количество мест

public string[] StopStation { get; set; } // Места промежуточной посадки

public int FreePlaces // Свойство, которое показывает количество свободных мест на борту

{

get

{

return ListOfPassengers.passengers.Length - ListOfPassengers.Count;

}

}

public void ReadFromFile() // Метод, предназначенный для подключения к файлу определенного рейса для получения списка пассажиров

{

ListOfPassengers = new CollectionOfPassengers(CountPlaces);

Directory.CreateDirectory("Passengers");

using (FileStream file = new FileStream("Passengers\\" + Number + "Passengers.txt", FileMode.OpenOrCreate))

{

StreamReader pass = new StreamReader(file);

while (!pass.EndOfStream)

{

string s = pass.ReadLine();

string[] arr = s.Split('\_');

if (arr.Length == 3)

ListOfPassengers.AddPassenger(new Passenger(arr[0], arr[1], arr[2]));

else

throw new ArgumentOutOfRangeException(); //Выбрасываем исключение если не содержит трёх элементов (имя, фамилия, отчество)

}

pass.Close();

file.Close();

}

}

public void AddPassenger(Passenger user, decimal count) // Метод для добавления пассажира на рейс

{

if (count <= this.FreePlaces)

{

while (count != 0)

{

this.ListOfPassengers.AddPassenger(user);

user.WriteInFile(this.Number);

count--;

}

}

else

{

throw new Exception("Недостаточно свободных мест!");

}

}

public void AddStopStation(string stopStat) // Метод для добавления промежуточных станций

{

if (this.StopStation.Contains(stopStat) || this.DepartureFrom == stopStat || this.ArrivalIn == stopStat)

{

throw new Exception("Данный пункт уже включен в пункты промежуточной посадки этого рейса.\nИли данный рейс отправляется или прибывает в заданный пункт. ");

}

else

{

List<string> arr = this.StopStation.ToList<string>();

arr.Add(stopStat);

StopStation = arr.ToArray<string>();

}

}

public override string ToString()

{

return string.Format("{0}|{1}|{2}|{3}|{4}|{5}|{6}|{7}|{8}",

Number,

DepartureFrom,

ArrivalIn,

Departure.TimeOfDay.ToString("hh':'mm"),

Departure.ToString().Split(' ')[0],

Arrival.TimeOfDay.ToString("hh':'mm"),

Arrival.ToString().Split(' ')[0],

CountPlaces,

Program.GetStopStations(StopStation));

}

}

Клас FlightCollection.

/\*

Класс FlightCollection наследуемый от List<Flight> является важнейшим во всей программе. В конструкторе происходит

подключение к указанному файлу, который содержит в себе все рейсы. Благодаря тому, что конструктор может принимать от 0 до 2 параметров,

в коллекцию попадают только те рейсы, которые удовлетворяют параметрам конструктора.

\*/

class FlightCollection : List<Flight>

{

public string Path;

public FlightCollection(string \_path,string dep = null, string arrival = null)

{

Path = \_path;

this.CheckFile();

DateTime now = DateTime.Now;

using (StreamReader flights = new StreamReader(Path))

{

while (!flights.EndOfStream)

{

string s = flights.ReadLine();

string[] arr = s.Split('|');

if (arr.Length != 9)

{

throw new Exception();

}

if (dep != null && arrival != null)

{

if (Convert.ToDateTime(arr[3] + " " + arr[4]) >= now &&

(arr[1] == dep && arr[2] == arrival || arr[1] == dep && arr[8].IndexOf(arrival) != -1))

{

this.Add(new Flight(arr[0], arr[1], arr[2], arr[3],arr[4], arr[5], arr[6], Convert.ToInt32(arr[7]), arr[8].Split(',')));

}

}

else if ((dep == null || arrival == null) && Convert.ToDateTime(arr[3] + " " + arr[4]) >= now)

{

this.Add(new Flight(arr[0], arr[1], arr[2], arr[3],arr[4], arr[5], arr[6], Convert.ToInt32(arr[7]), arr[8].Split(',')));

}

}

flights.Close();

}

}

public void SortingByDays() // Метод сортировки пузырьком рейсов по отправлению.

{

for (int i = 0; i < this.Count - 1; i++)

{

for (int j = i + 1; j < this.Count; j++)

{

if (this[j].Departure > this[i].Departure)

{

var t = this[i];

this[i] = this[j];

this[j] = t;

}

}

}

}

public void WriteInFile() // Метод, который записывает в файл построчно данные о всех рейсах

{

using (StreamWriter wr = new StreamWriter(Path, true))

{

foreach(Flight ob in this)

{

wr.WriteLine(ob.ToString());

}

wr.Close();

}

}

public void ClearFile() // метод, который очищает файл, который хранит все рейсы.

{

using (StreamWriter wr = new StreamWriter(Path))

{

wr.Write("");

wr.Close();

}

}

public Flight FindByNumber(int num) // Метод, который находит заданный по номеру рейс

{

foreach(Flight ob in this)

{

if (ob.Number == num)

{

return ob;

}

}

return null;

}

public void AddFlightAndWriteInFile(Flight obj) // Метод, который добавляет рейс и сразу записывает его в файл

{

this.Add(obj);

using (StreamWriter wr = new StreamWriter(Path, true))

{

wr.WriteLine(obj.ToString());

wr.Close();

}

}

public void RemoveFlightAndFile(string num) // Метод для удаления рейса и файла, содержащего список пассажиров, зарегистрированных на этот рейс

{

Flight Current = this.FindByNumber(Convert.ToInt32(num));

if (Current != null)

{

this.Remove(Current);

if (File.Exists("Passengers\\" + num + "Passengers.txt") == true)

{

File.Delete("Passengers\\" + num + "Passengers.txt");

}

}

}

public void CheckFile() // Метод для проверки на существование файла со всеми рейсами

{

if (File.Exists(Path) == false)

{

if (Directory.Exists("Passengers"))

{

Directory.Delete("Passengers", true);

}

FileStream file = new FileStream(Path,FileMode.Create);

file.Close();

}

}

}

Клас Schedule : Form

public partial class Schedule : Form

{

public Schedule()

{

InitializeComponent();

}

private void Schedule\_Load(object sender, EventArgs e)

{

OnLoadFuction();

}

public void OnLoadFuction()

{

foreach (DataGridViewColumn colum in FlightTable1.Columns)

{

colum.SortMode = DataGridViewColumnSortMode.NotSortable;

}

try

{

FlightTable1.Rows.Clear();

FlightCollection list = new FlightCollection("Flights.txt");

list.SortingByDays();

for (int i = list.Count - 1; i >= 0; i--)

{

FlightTable1.Rows.Add(list[i].Number,

list[i].FreePlaces,

list[i].DepartureFrom,

list[i].ArrivalIn,

list[i].Departure.TimeOfDay.ToString("hh':'mm"),

list[i].Departure.Date.ToString().Split(' ')[0],

list[i].Arrival.TimeOfDay.ToString("hh':'mm"),

list[i].Arrival.Date.ToString().Split(' ')[0],

Program.GetStopStations(list[i].StopStation));

}

if (FlightTable1.Rows.Count == 0)

{

button3DeleteFlight.Visible = false;

button5InfomOfFlight.Visible = false;

button2EditFlight.Visible = false;

}

else

{

button3DeleteFlight.Visible = true;

button5InfomOfFlight.Visible = true;

button2EditFlight.Visible = true;

}

}

catch

{

this.Show();

MessageBox.Show("Ошибка в базе данных!", "Оповещение");

this.Close();

}

}

private void бронированиеБилетовToolStripMenuItem\_Click(object sender, EventArgs e)

{

Reserve ob = new Reserve();

ob.Show();

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

private void оПрограммеToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show(@"Тема курсового проекта: Касса аэрофлота

Разработчик: ст.группы ПИ-15-2 Джафаров Эмиль

Электронная почта: emil.dzhafarov@nure.ua

Версия программы: 1.0", "О программе");

}

private void выходToolStripMenuItem1\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button3\_Click(object sender, EventArgs e)

{

string num = FlightTable1.CurrentRow.Cells[0].Value.ToString();

FlightCollection file = new FlightCollection("Flights.txt");

file.RemoveFlightAndFile(num);

file.ClearFile();

file.WriteInFile();

OnLoadFuction();

}

private void button2\_Click(object sender, EventArgs e)

{

string num = FlightTable1.CurrentRow.Cells[0].Value.ToString();

EditFlight ob = new EditFlight(num);

ob.Show();

}

private void button5\_Click(object sender, EventArgs e)

{

string num = FlightTable1.CurrentRow.Cells[0].Value.ToString();

Info ob = new Info(num);

ob.Show();

}

private void добавитьРейсToolStripMenuItem\_Click\_1(object sender, EventArgs e)

{

AddFlight ob = new AddFlight();

ob.Show();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}

Клас Reserve:Form

public partial class Reserve : Form

{

public Reserve()

{

InitializeComponent();

}

private void Reserve\_Load(object sender, EventArgs e)

{

foreach (DataGridViewColumn colum in FlightTable.Columns)

{

colum.SortMode = DataGridViewColumnSortMode.NotSortable;

}

}

private void button2Ok\_Click(object sender, EventArgs e)

{

try

{

string Name = textBox1Name.Text;

string Surname = textBox2Surname.Text;

string Middlename = textBox3Middlename.Text;

string num = FlightTable.CurrentRow.Cells[0].Value.ToString();

decimal count = numericUpDown1Count.Value;

Passenger user = new Passenger(Name, Surname, Middlename);

FlightCollection arr = new FlightCollection("Flights.txt");

Flight Current = arr.FindByNumber(Convert.ToInt32(num));

Current.AddPassenger(user, count);

MessageBox.Show("Пассажир добавлен!", "Оповещение");

this.Close();

Program.MainForm.OnLoadFuction();

}

catch (NullReferenceException)

{

MessageBox.Show("Не выбрано ни одного рейса!", "Оповещение");

}

catch (Exception my)

{

MessageBox.Show(my.Message, "Оповещение");

}

}

private void SearchFlight\_Click(object sender, EventArgs e)

{

try

{

FlightTable.Rows.Clear();

string dep = Program.RemoveSpaces(textBox4DepartureFrom.Text);

string arrival = Program.RemoveSpaces(textBox5arrivalIn.Text);

DateTime now = DateTime.Now;

FlightCollection list = new FlightCollection("Flights.txt",dep, arrival);

list.SortingByDays();

for (int i = list.Count - 1; i >= 0; i--)

{

if (list[i].Departure > now)

{

FlightTable.Rows.Add(list[i].Number,

list[i].FreePlaces,

list[i].DepartureFrom,

list[i].ArrivalIn,

list[i].Departure.TimeOfDay.ToString("hh':'mm"),

list[i].Departure.Date.ToString().Split(' ')[0],

list[i].Arrival.TimeOfDay.ToString("hh':'mm"),

list[i].Arrival.Date.ToString().Split(' ')[0],

Program.GetStopStations(list[i].StopStation)

);

}

}

if (FlightTable.RowCount == 0) // Проверка на существование рейсов в базе данных

{

MessageBox.Show("Не найдено соответствующего рейса", "Оповещение");

return;

}

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void button3Reset\_Click(object sender, EventArgs e)

{

this.Close();

}

}

Клас Info:Form

public partial class Info : Form

{

private int Number;

public Info(string str)

{

Number = Convert.ToInt32(str);

InitializeComponent();

}

private void Info\_Load(object sender, EventArgs e)

{

try

{

foreach (DataGridViewColumn colum in ListOfPassengers.Columns)

{

colum.SortMode = DataGridViewColumnSortMode.NotSortable;

}

FlightCollection file = new FlightCollection("Flights.txt");

Flight Current = file.FindByNumber(Number);

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

rez.Add("Номер: " + Current.Number);

rez.Add("Откуда отправляется: " + Current.DepartureFrom);

rez.Add("Куда направляется: " + Current.ArrivalIn);

rez.Add("Дата отправления: " + Current.Departure.Date.ToString().Split(' ')[0]);

rez.Add("Время отправления: " + Current.Departure.TimeOfDay.ToString("hh':'mm"));

rez.Add("Дата прибытия: " + Current.Arrival.Date.ToString().Split(' ')[0]);

rez.Add("Время прибытия: " + Current.Arrival.TimeOfDay.ToString("hh':'mm"));

rez.Add("Количество посадочных мест: " + Current.CountPlaces);

rez.Add("Свободные места: " + Current.FreePlaces);

rez.Add("Пункты промежуточной посадки: " + Program.GetStopStations(Current.StopStation));

listBox1info.DataSource = rez;

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

{

ListOfPassengers.Rows.Add(Current.ListOfPassengers[i].Name,

Current.ListOfPassengers[i].Surname,

Current.ListOfPassengers[i].MiddleName);

}

}

catch(NullReferenceException)

{

MessageBox.Show("Ошибка: время отправления этого рейса стало меньше текущего.", "Оповещение");

this.Close();

Program.MainForm.OnLoadFuction();

}

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

}

Клас EditFlight:Form

public partial class EditFlight : Form

{

private int Number;

public EditFlight(string ob)

{

Number = Convert.ToInt32(ob);

InitializeComponent();

}

private void button1Ok\_Click(object sender, EventArgs e)

{

try

{

FlightCollection File = new FlightCollection("Flights.txt");

Flight Current = File.FindByNumber(Number);

string depDay = dateTimePicker1DepartureDay.Value.Day.ToString() + "."

+ dateTimePicker1DepartureDay.Value.Month.ToString() + "."

+ dateTimePicker1DepartureDay.Value.Year.ToString();

string arrDay = dateTimePicker2arrivalDay.Value.Day.ToString() + "."

+ dateTimePicker2arrivalDay.Value.Month.ToString() + "."

+ dateTimePicker2arrivalDay.Value.Year.ToString();

string depTime = numericUpDown1DepHour.Value + ":" + numericUpDown2DepMinutes.Value;

string arrTime = numericUpDown1ArrHours.Value + ":" + numericUpDown1ArrMinutes.Value;

string StopStation = Program.RemoveSpaces(textBox3stopstat.Text);

if (StopStation != "")

{

Current.AddStopStation(StopStation);

}

Current.Departure = Convert.ToDateTime(depDay + " " + depTime);

Current.Arrival = Convert.ToDateTime(arrDay + " " + arrTime);

File.ClearFile();

File.WriteInFile();

this.Close();

Program.MainForm.OnLoadFuction();

}

catch (Exception myexp)

{

MessageBox.Show(myexp.Message, "Оповещение");

}

}

private void button2\_Click(object sender, EventArgs e)

{

this.Close();

}

private void EditFlight\_Load(object sender, EventArgs e)

{

try

{

FlightCollection File = new FlightCollection("Flights.txt");

Flight Current = File.FindByNumber(Number);

textBox1dep.Text = Current.DepartureFrom;

textBox1arr.Text = Current.ArrivalIn;

textBox1num.Text = Current.Number.ToString();

numericUpDown1DepHour.Value = Current.Departure.TimeOfDay.Hours;

numericUpDown2DepMinutes.Value = Current.Departure.TimeOfDay.Minutes;

numericUpDown1ArrHours.Value = Current.Arrival.TimeOfDay.Hours;

numericUpDown1ArrMinutes.Value = Current.Arrival.TimeOfDay.Minutes;

dateTimePicker1DepartureDay.Value = Current.Departure;

dateTimePicker2arrivalDay.Value = Current.Arrival;

}

catch(NullReferenceException)

{

MessageBox.Show("Ошибка: время отправления этого рейса стало меньше текущего.","Оповещение");

this.Close();

Program.MainForm.OnLoadFuction();

}

}

}

Клас AddFlight:Form

public partial class AddFlight : Form

{

public AddFlight()

{

InitializeComponent();

}

private void button2\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

string Number = Program.RemoveSpaces(textBox1Number.Text);

string depDay = dateTimePicker1DepartureDay.Value.Day.ToString() + "."

+ dateTimePicker1DepartureDay.Value.Month.ToString() + "."

+ dateTimePicker1DepartureDay.Value.Year.ToString();

string arrDay = dateTimePicker2ArrivalDay.Value.Day.ToString() + "."

+ dateTimePicker2ArrivalDay.Value.Month.ToString() + "."

+ dateTimePicker2ArrivalDay.Value.Year.ToString();

string depTime = numericUpDown4DepHours.Value + ":" + numericUpDown5DepMinutes.Value;

string arrTime = numericUpDown1ArrHours.Value + ":" + numericUpDown2ArrMinutes.Value;

string[] StopStations = Program.RemoveSpaces(textBox1StopStations.Text).Split(',');

string DepartureFrom = Program.RemoveSpaces(textBox1DepartureFrom.Text);

string ArrivalIn = Program.RemoveSpaces(textBox1ArrivalIn.Text);

int countPlaces = Convert.ToInt32(numericUpDown6countPlaces.Value);

Flight newFlight;

FlightCollection file = new FlightCollection("Flights.txt");

Flight Current = file.FindByNumber(Convert.ToInt32(Number));

if (Convert.ToDateTime(depDay + " " + depTime) <= DateTime.Now)

{

MessageBox.Show("Время и дата отправления должны быть больше текущих.", "Оповещение");

return;

}

if (DepartureFrom == "" || ArrivalIn == "")

{

MessageBox.Show("", "Оповещение");

return;

}

if (Current == null)

{

newFlight = new Flight(Number, DepartureFrom, ArrivalIn, depTime, depDay, arrTime, arrDay, countPlaces, StopStations);

file.AddFlightAndWriteInFile(newFlight);

}

else

{

MessageBox.Show("Рейс с таким номером уже существует.", "Оповещение");

return;

}

this.Close();

Program.MainForm.OnLoadFuction();

}

catch (FormatException)

{

MessageBox.Show("Проверьте правильность введённых данных.", "Оповещение");

}

catch(ArithmeticException)

{

MessageBox.Show("Номер рейса слишком длинный.", "Оповещение");

}

catch (Exception my)

{

MessageBox.Show(my.Message, "Оповещение");

}

}

}