

**Минобрнауки России**  
**Юго-Западный государственный университет**

Кафедра программной инженерии

**КУРСОВАЯ РАБОТА**

по дисциплине «Языки объектно-ориентированного программирования »  
(наименование дисциплины)  
на тему «Программа для моделирования сервиса для продажи  
авиабилетов»

Направление подготовки (специальность) 09.03.04  
(код, наименование)  
Программная инженерия

Автор работы (проекта) Е.И. Машенцева  
(инициалы, фамилия) \_\_\_\_\_ (подпись, дата)

Группа ПО-32б

Руководитель работы (проекта) Е.И. Аникина  
(инициалы, фамилия) \_\_\_\_\_ (подпись, дата)

Работа (проект) защищена \_\_\_\_\_  
(дата)

Оценка \_\_\_\_\_

Члены комиссии \_\_\_\_\_  
(подпись, дата) \_\_\_\_\_ (инициалы, фамилия)

\_\_\_\_\_ (подпись, дата) \_\_\_\_\_ (инициалы, фамилия)

\_\_\_\_\_ (подпись, дата) \_\_\_\_\_ (инициалы, фамилия)

Курск 2024 г.

**Минобрнауки России**  
**Юго-Западный государственный университет**

Кафедра программной инженерии

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ (ПРОЕКТ)**

Студент Е.И. Машенцева шифр 23-06-0017 группа ПО-326  
(инициалы, фамилия)

1. Тема Программа для моделирования сервиса для продажи авиабилетов

2. Срок представления работы (проекта) к  
защите « \_\_\_\_ » \_\_\_\_\_ 20 \_\_\_\_ г.

3. Исходные данные (*для проектирования, для научного исследования*):

Определяются требованиями пользователя.

4. Содержание пояснительной записки курсовой работы (проекта):

4.1 ВВЕДЕНИЕ

4.2 Анализ и моделирование предметной области приложения

4.3 Техническое задание

4.4 Технический проект

4.5 Рабочий проект

4.6 ЗАКЛЮЧЕНИЕ

4.7 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

4.8 ПРИЛОЖЕНИЕ А

4.9 ПРИЛОЖЕНИЕ Б

5. Перечень графического материала: диаграмма прецедентов, диаграмма

классов, внешний вид интерфейса, макет интерфейса, таблицы тестовых наборов, таблица

описания объектов интерфейса.

Руководитель работы (проекта)

Е.И. Аникина

(инициалы, фамилия)

(подпись, дата)

Задание принял к исполнению

Е.И. Машенцева

(инициалы, фамилия)

(подпись, дата)

## РЕФЕРАТ

Данный текстовый документ имеет объем 144 страница и включает в себя 56 рисунков, 11 таблиц, 2 приложения, 15 библиографических источников.

Перечень ключевых слов: авиабилеты, билеты, перелёты, рейсы, поиск авиабилетов, аэропорты, цены на авиабилеты.

Целью работы является программная реализация на языке C# программы для моделирования сервиса для продажи авиабилетов. Программный продукт предназначен для оптимизации процесса продажи авиабилетов.

При создании программного продукта с локальной архитектурой применялись технология объектно-ориентированного программирования.

## ABSTRACT

This text document has a volume of 144 pages and includes 56 figures, 11 tables, 2 appendices, 15 bibliographic sources.

The list of keywords: air tickets, tickets, flights, flights, flight search, airports, air ticket prices.

The purpose of the work is the software implementation in C# of a program for modeling a service for the sale of air tickets. The software product is designed to optimize the process of selling air tickets.

When creating a software product with a local architecture, object-oriented programming technology was used.

## СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ.....</b>	<b>7</b>
<b>1. Анализ и моделирование предметной области приложения .....</b>	<b>8</b>
1.1. Описание предметной области .....	8
1.2. Моделирование вариантов использования .....	17
1.2.1. Варианты использования.....	17
1.2.2. Диаграмма прецедентов .....	19
1.2.3. Сценарии прецедентов программного изделия .....	21
1.2.4. Словарь предметной области приложения.....	26
1.2.5. Моделирование предметной области с помощью диаграммы классов.....	27
<b>2. Техническое задание .....</b>	<b>29</b>
2.1. Основание для разработки .....	29
2.2. Назначение разработки .....	29
2.3. Требования к программе или программному изделию.....	29
2.3.1. Требования к функциональным характеристикам .....	29
2.3.2. Требования к входным и выходным данным.....	30
2.3.3. Требования пользователя к интерфейсу .....	30
2.3.4. Требования к надежности .....	41
2.3.5. Условия эксплуатации .....	41
2.3.6. Требования к составу и параметрам технических средств.....	41
2.3.7. Требования к информационной и программной совместимости.	41
2.4. Требования к программной документации .....	42
2.5. Стадии и этапы разработки.....	42
2.6. Порядок контроля и приемки .....	42

<b>3. Технический проект.....</b>	<b>44</b>
3.1. Моделирование последовательности действий.....	44
3.2. Проектирование архитектуры программной системы.....	45
3.3. Описание структур и форматов данных.....	46
3.4. Схемы алгоритмов .....	46
<b>4. Рабочий проект .....</b>	<b>47</b>
4.1. Спецификация компонентов и классов программ.....	47
4.1.1. Модули и объекты интерфейса пользователя .....	47
4.1.2. Описание объектов интерфейса программы .....	47
4.1.3. Описание классов .....	73
4.2. Тестирование программной системы .....	78
<b>ЗАКЛЮЧЕНИЕ .....</b>	<b>112</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....</b>	<b>113</b>
<b>ПРИЛОЖЕНИЕ А.....</b>	<b>115</b>
<b>ПРИЛОЖЕНИЕ Б.....</b>	

## ВВЕДЕНИЕ

Целью курсовой работы является создание на языке C# компьютерной программы для моделирования сервиса для продажи авиабилетов.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Разработка архитектуры приложения.
2. Разработка интерфейса приложения.
3. Разработка функции обработки данных.
4. Разработка функции ввода данных.
5. Разработка функции выбора места в самолёте.
6. Разработка функции оплаты билета.
7. Разработка функции очищения текстовых полей.
8. Реализация хранения данных.
9. Реализация алгоритма генерации рейсов.
10. Реализация прекращения работы приложения.

Основные результаты. В ходе разработки получены следующие результаты:

1. Была разработана архитектура приложения.
2. Был разработан интерфейс программы.
3. Была разработана функция обработки данных.
4. Была разработана функция ввода данных.
5. Была разработана функция выбора места в самолёте.
6. Была разработана функция оплаты билета.
7. Была разработана функция очищения текстовых полей.
8. Было реализовано хранение данных.
9. Был реализован алгоритм генерации рейсов.
10. Было реализовано прекращение работы приложения.

## **1. Анализ и моделирование предметной области приложения**

### **1.1 Описание предметной области**

Приложение разрабатывается с целью показать работу сервиса для продажи авиабилетов. Сервис для продажи авиабилетов представляет собой совокупность множества разных составляющих. Функционирует система базы данных, которая обеспечивает хранение информации об актуальных рейсах, моделях самолетов, количестве мест для пассажиров, а также данные зарегистрированных на сервисе пользователей.

При открытии сервиса для продажи авиабилетов пользователю доступен стартовый экран, на котором изображен логотип сервиса. Также, на стартовом экране присутствует кнопка “Далее”, при нажатии на которую пользователь программа загрузит следующую страницу и переместит пользователя на неё. Стартовый экран представлен на рисунке 1.1.



Рисунок 1.1- стартовая страница сервиса для продажи авиабилетов

Когда система загружает главную страницу сервиса, пользователю становятся доступны следующие функции: параметры поиска, поиск билетов, просмотр интересных направлений, корзина покупок, личный кабинет. На данной странице пользователь может ввести необходимые данные (например, место вылета и прибытия, дату полёта, количество пассажиров и тип класса) для дальнейшего выбора рейса. После ввода всех необходимых данных система обрабатывает их и находит рейсы (в случае, если данные введены не все и введены некорректно, система выдаст уведомление об ошибке). Чтобы просмотреть доступные рейсы по заданным параметрам, клиенту необходимо нажать на кнопку “Найти рейсы”, после чего система закроет главную страницу и откроет страницу с подобранными рейсами. Также на главной странице пользователь может ознакомиться с наиболее интересными направлениями полётов. Если пользователю нужно вернуться на главную страницу сервиса, он может сделать это, нажав на логотип сервиса, находящийся в левом верхнем углу страницы. Также, если пользователь захочет пройти авторизацию или авторизоваться в сервисе, он может нажать на значок пользователя, который находится в правом верхнем углу страницы. Если пользователь захочет выйти из сервиса для продажи авиабилетов, ему необходимо будет нажать кнопку, которой изображено облако с крестиком и тогда программа завершит работу. Главная страница представлена на рисунке 1.2.

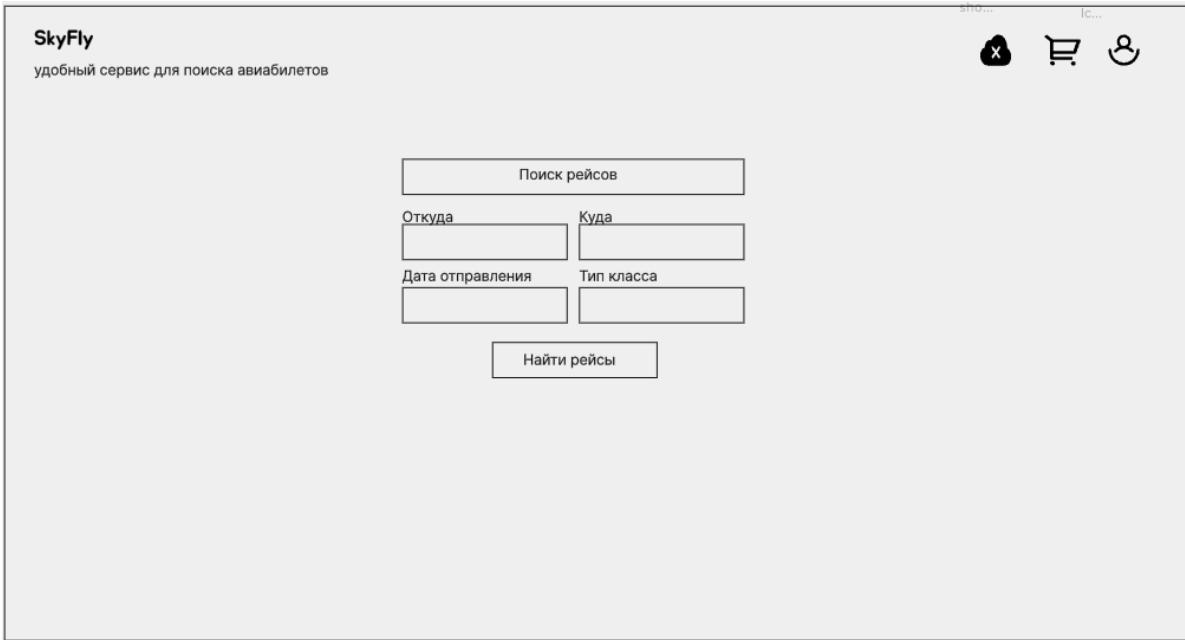


Рисунок 1.2 - главная страница сервиса для продажи авиабилетов

При переходе на страницу с подобранными рейсами пользователь может ознакомиться с предложенными актуальными рейсами. Чтобы выбрать нужный билет, клиенту необходимо нажать на кнопку “Добавить”, находящуюся рядом с каждым билетом. Система добавляет выбранный пользователем рейс в корзину покупок. После того, как пользователь выбрал подходящий рейс, ему необходимо перейти в корзину покупок для проверки данных рейса. Если пользователю нужно вернуться на главную страницу сервиса, он может сделать это, нажав на логотип сервиса, находящийся в левом верхнем углу страницы. Также, если пользователь захочет пройти авторизацию или авторизоваться в сервисе, он может нажать на значок пользователя, который находится в правом верхнем углу страницы. Если пользователь захочет выйти из сервиса для продажи авиабилетов, ему необходимо будет нажать кнопку, которой изображено облако с крестиком и тогда программа завершит работу. Страница подобранных рейсов представлена на рисунке 1.3.

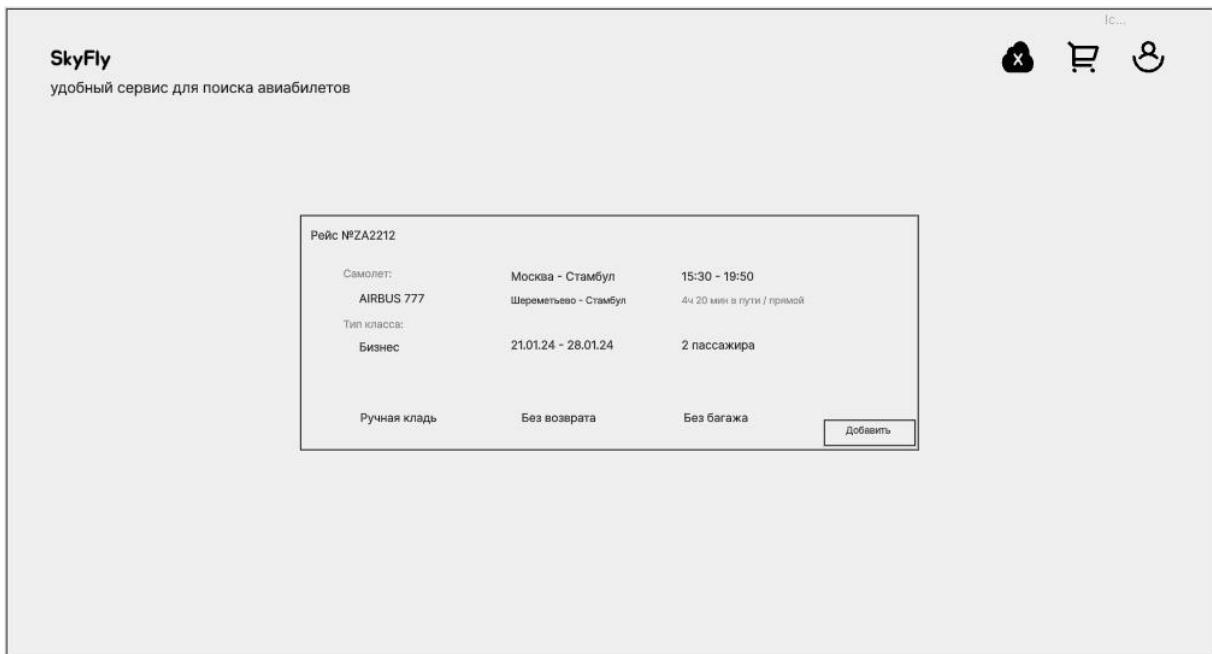


Рисунок 1.3 - страница подобранных рейсов

При нажатии пользователем кнопки “Корзина покупок” система загружает страницу с корзиной покупок, в которой находится выбранный пользователем рейс. На данной странице пользователь может проверить данные рейса, добавить багаж, сделать билет принадлежащим обмену или возврату, а также при желании выбрать место в самолёте. В случае если пользователь хочет выбрать место в самолёте, ему необходимо нажать на соответствующую кнопку. Система загрузит страницу для выбора места, на которой будет изображено расположение свободных мест в самолёте, выполняющем рейс. Пользователь выбирает желаемые места, после чего нажимает кнопку “Сохранить изменения”. Система сохранит и внесет изменения в билет и откроет пользователю страницу корзины покупок для сверки данных. В случае если пользователь не нуждается в выборе места в самолёте, система присваивает место автоматически, после чего клиент может перейти к оплате билета. Если на момент оплаты билета пользователь не авторизован, программа выдаст ошибку о переходе к оплате и попросит пользователя зарегистрироваться или войти в личный кабинет, после чего

пользователь сможет оплатить покупку и приобрести билет на выбранный рейс. Оплата билета на рейс происходит следующим образом: пользователю необходимо ввести данные карты, а также сверить номер телефона, указанный на сервисе (если программа вставила неверный номер, пользователь может его изменить), после чего клиент должен нажать кнопку “Получить код”. Если все поля заполнены верно, система вышлет проверочный код для оплаты на указанный номер. После чего пользователю необходимо будет ввести полученный код в соответствующее поле. Далее пользователю необходимо нажать кнопку “Ввести код”. Система выполнит проверку на совпадение кода. Если введённый код окажется правильным, пользователь получит уведомление об успешной оплате, иначе система уведомит пользователя о неверно введённом коде. Если пользователь захочет удалить выбранный рейс из корзины покупок, ему необходимо нажать кнопку “Удалить”, система удалит рейс из корзины и обновит страницу. Если пользователю нужно вернуться на главную страницу сервиса, он может сделать это, нажав на логотип сервиса, находящийся в левом верхнем углу страницы. Также, если пользователь захочет пройти авторизацию или авторизоваться в сервисе, он может нажать на значок пользователя, который находится в правом верхнем углу страницы. Если пользователь захочет выйти из сервиса для продажи авиабилетов, ему необходимо будет нажать кнопку, которой изображено облако с крестиком и тогда программа завершит работу. Страница корзины покупок представлена на рисунке 1.4, страница для выбора места в самолёте представлена на рисунке 1.5, страница для сверки данных и оплаты билета представлены на рисунке 1.6.



Рисунок 1.4 - страница корзины покупок

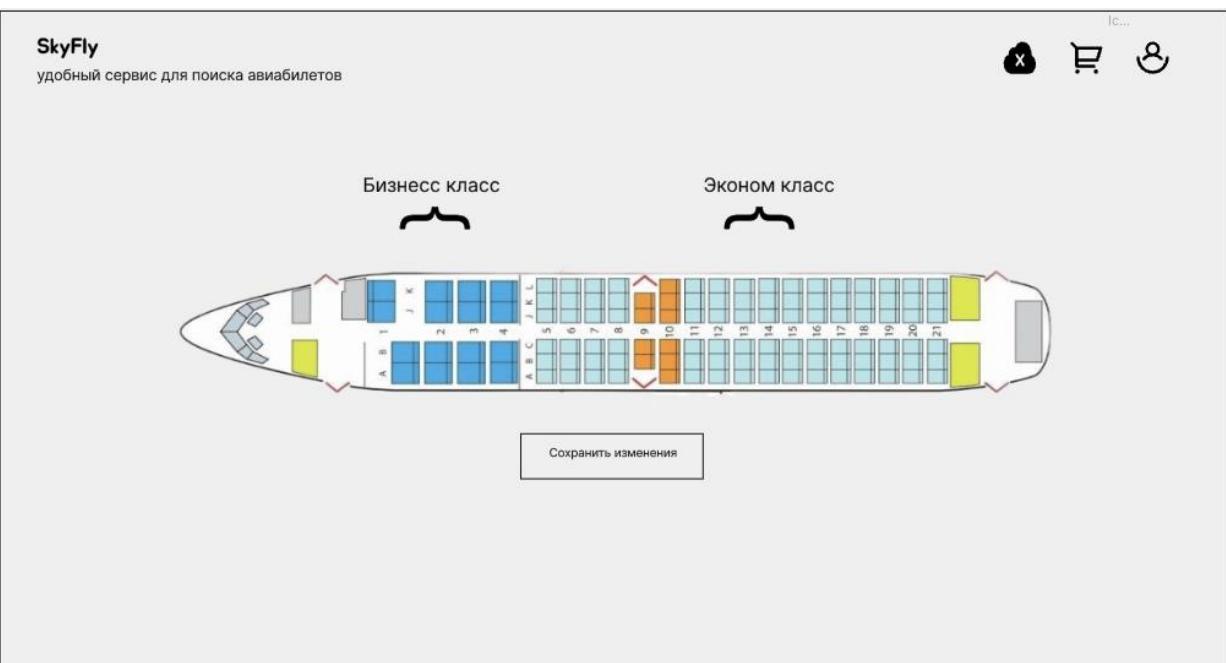


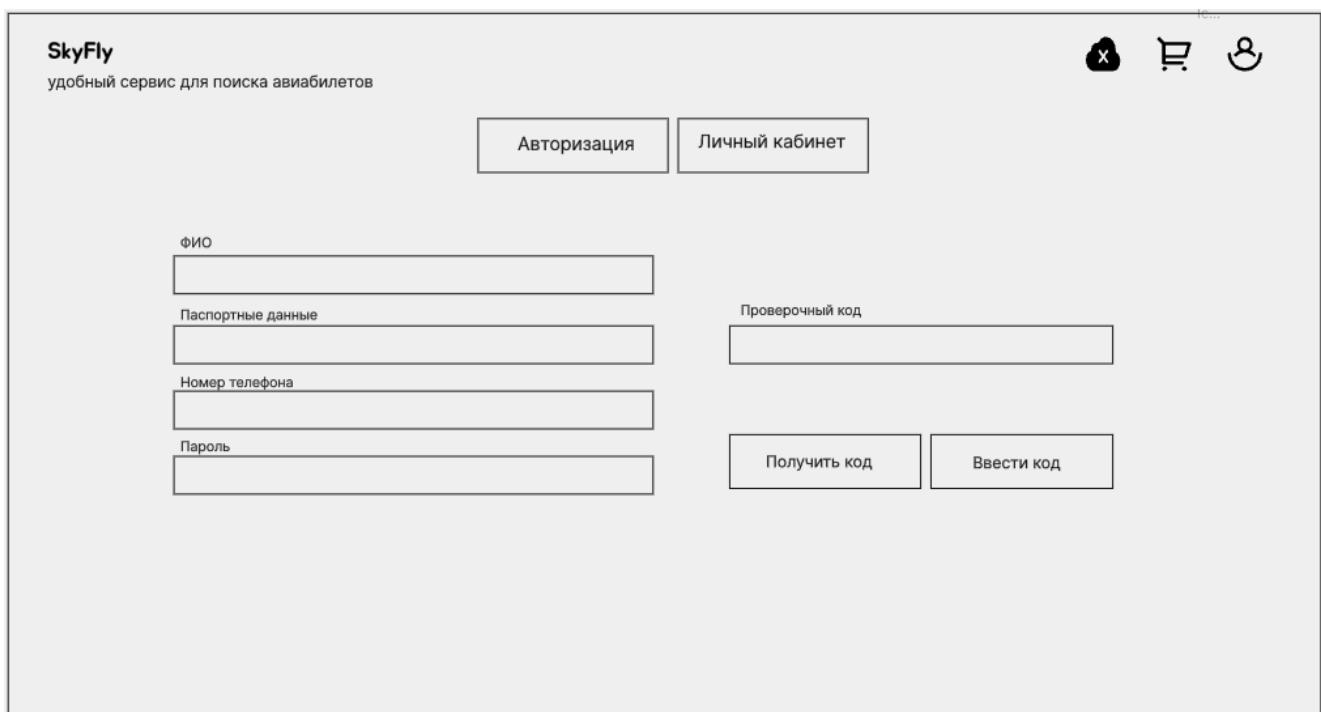
Рисунок 1.5 - страница для выбора места в самолёте

The screenshot shows a web interface for a flight booking service. At the top left is the logo 'SkyFly' with the tagline 'удобный сервис для поиска авиабилетов'. In the top right corner are icons for user profile, cart, and search. Below the header, there are two main sections: 'Рейс №ZZ2222' (Flight #ZZ2222) containing flight details like 'Самолёт: AIRBUS 737', 'Москва - Стамбул Домодедово - Стамбул', 'Имя пассажира: Иванов Иван Иванович', and travel dates '21.01.24 - 28.01.24'; and 'Оплата' (Payment) section with fields for card number, expiration date, CVC code, phone number, and a checkbox for a verification code. A 'Получить код' (Get code) button is at the bottom of the payment form.

Рисунок 1.6 – страница сверки данных и оплаты билета

Для покупки билета на рейс на сервисе для продажи авиабилетов клиенту нужно пройти регистрацию, если он ранее не был зарегистрирован или выполнить вход в личный кабинет. В верхнем правом углу находится значок личного кабинета, при нажатии на который пользователь может пройти регистрацию на сервисе, выполнить вход и перейти в личный кабинет. При регистрации система высылает клиенту проверочный код. После чего пользователю необходимо будет ввести полученный код в соответствующее поле. Система выполнит проверку на совпадение кода. Если введённый код окажется правильным, пользователь получит уведомление об успешной регистрации, иначе система уведомит пользователя о неверно введённом коде. Если клиент нажмет кнопку “Выслать новый код”, система вышлет пользователю новый код”. После чего система сохраняет все введенные данные и регистрирует пользователя. Если пользователь был зарегистрирован на сервисе ранее, он может ввести логин и пароль. Если пользователю нужно вернуться на главную страницу сервиса, он может сделать это, нажав на логотип сервиса, находящийся в левом верхнем углу

страницы. Также, если пользователь захочет пройти авторизацию или авторизоваться в сервисе, он может нажать на значок пользователя, который находится в правом верхнем углу страницы. Если пользователь захочет выйти из сервиса для продажи авиабилетов, ему необходимо будет нажать кнопку, которой изображено облако с крестиком и тогда программа завершит работу. Страница для регистрации представлена на рисунке 1.7 , страница для авторизации представлена на рисунке 1.8.



The image shows the registration page for the SkyFly service. At the top left is the logo 'SkyFly' and the tagline 'удобный сервис для поиска авиабилетов'. On the right side are icons for user profile, shopping cart, and a cloud with a cross. Below the logo are two buttons: 'Авторизация' (Authorization) and 'Личный кабинет' (Personal Cabinet). The main form area contains four input fields: 'ФИО' (Name), 'Паспортные данные' (Passport data), 'Номер телефона' (Phone number), and 'Пароль' (Password). To the right of the 'Номер телефона' field is a 'Проверочный код' (Verification code) input field with a placeholder '16...'. Below the input fields are two buttons: 'Получить код' (Get code) and 'Ввести код' (Enter code).

Рисунок 1.7 – страница для регистрации

SkyFly  
удобный сервис для поиска авиабилетов

Регистрация    Личный кабинет

ФИО

Получите код, если забыли пароль

Номер телефона

Проверочный код

Пароль

Получить код    Ввести код

Войти

Рисунок 1.8 – страница для авторизации

В случае, если пользователю необходимо изменить персональные данные, ему необходимо предварительно зарегистрироваться или авторизоваться в сервисе. Далее пользователь нажимает на кнопку “Личный кабинет”, далее откроется страница с личным кабинетом, где будут храниться данные пользователя. Пользователь может изменить данные, после чего нажать кнопку “Сохранить изменения”. Система сохранит изменения и обновит их в файле. Также пользователь может нажать кнопку “Выйти из учетной записи”. Система очистит все поля. Если пользователю нужно вернуться на главную страницу сервиса, он может сделать это, нажав на логотип сервиса, находящийся в левом верхнем углу страницы. Также, если пользователь захочет пройти авторизацию или авторизоваться в сервисе, он может нажать на значок пользователя, который находится в правом верхнем углу страницы. Если пользователь захочет выйти из сервиса для продажи авиабилетов, ему необходимо будет нажать кнопку, которой изображено

облако с крестиком и тогда программа завершит работу. Страница личного кабинета представлена на рисунке 1.9.

The screenshot shows the 'SkyFly' personal account page. At the top left is the service logo 'SkyFly' with the tagline 'удобный сервис для поиска авиабилетов'. On the right side are icons for a profile picture, a cloud, and a gear. Below the header, there are four input fields labeled 'ФИО', 'Паспортные данные', 'Номер телефона', and 'Пароль', each with a corresponding empty text box. At the bottom of the form are two buttons: 'Сохранить изменения' and 'Выйти из учетной записи'.

Рисунок 1.9 – страница личного кабинета

## 1.2 Моделирование вариантов использования

### 1.2.1 Варианты использования

Приложение должно моделировать следующие ситуации:

1. Пользователь открывает сервис для продажи авиабилетов, система загружает стартовый экран сервиса.
2. Пользователь может нажать на кнопку “Далее” и перейти на главную страницу сервиса. Программа загружает страницу, на которой будут кнопки для перехода в корзину покупок, входа в личный кабинет и завершения работы приложения, а также поля для ввода необходимых данных для поиска рейсов. Пользователь

вводит все необходимые данные и нажимает кнопку “Найти билеты”.

3. Программа загружает страницу с подобранными рейсами, согласно введённым данным клиента. Пользователь может просмотреть все рейсы и выбрать подходящие для него, нажав на кнопку “Добавить”. Система добавит выбранный рейс в корзину покупок.
4. Пользователь может перейти в корзину покупок, значок который находится в верхнем правом углу страницы и убедиться в правильности выбора. Также пользователь может выбрать место в самолёте, добавить провоз багажа и сделать билет подлежащим обмену и возврату. Клиент может удалить рейс из корзины покупок, нажав кнопку “Удалить”. Также пользователь может вернуться на главную страницу, нажав на логотип сервиса, находящийся в левом верхнем углу страницы.
5. Пользователь может перейти к сверке данных и оплате билета с помощью кнопки “Перейти к оплате”. Если пользователь авторизован или зарегистрирован, то система загрузит страницу, на которой находится выбранный пользователем билет и поля для ввода данных для оплаты.
6. Если пользователь заранее выполнил вход в личный кабинет сервиса, программа автоматически внесет персональные данные из личного кабинета. Далее пользователь сверяет данные билета, если его всё устраивает, то он вводит данные карты, после чего система проверяет правильность введенных данных. Если всё верно, пользователь нажимает на кнопку “Получить код”. Клиенту необходимо будет ввести проверочный код для оплаты, после чего нажать кнопку “Ввести код”. Система проверяет этот код, если он

является правильным, пользователь получает уведомление об успешной оплате.

7. Пользователь может войти в личный кабинет, авторизоваться или зарегистрироваться на сервисе для продажи авиабилетов. Клиент нажимает на значок личного кабинета, находящийся в правом верхнем углу страницы. Программа загружает страницу для входа или регистрации пользователя. Если клиент был зарегистрирован на сервисе ранее, он вводит номер телефона и проверочный код, отправленный программой. После авторизации или авторизации пользователь может нажать на кнопку “Личный кабинет”, после чего программа откроет страницу с личным кабинетом пользователя.

### **1.2.2 Диаграмма прецедентов**

В системе должен быть представлен один действующий пользователь.

В программе должны быть реализованы следующие прецеденты:

1. Прецедент “Поиск рейсов”. Позволяет пользователю установить параметры для поиска нужного рейса.
2. Прецедент “Найти билеты”. Открывает страницу с подобранными рейсами по индивидуальным параметрам пользователя.
3. Прецедент “Добавить билет”. Позволяет пользователю добавить выбранный рейс в корзину покупок.
4. Прецедент “Ввод дополнительных параметров”. Позволяет пользователю добавить в стоимость билета перевозку багажа, сделать билет принадлежащим обмену и возврату и выбрать место в самолёте.
5. Прецедент “Перейти к оплате”. Позволяет пользователю оплатить выбранный рейс.

6. Прецедент “Авторизация в личном кабинете”. Позволяет пользователю авторизоваться в личном кабинете.
7. Прецедент “Регистрация на сервисе”. Позволяет пользователю ввести необходимые данные и зарегистрироваться на сервисе.
8. Прецедент “Просмотр содержимого корзины”. Позволяет пользователю перейти в корзину покупок для просмотра выбранных рейсов.
9. Прецедент “Просмотр личного кабинета”. Позволяет пользователю перейти в личный кабинет для просмотра или изменения персональных данных.

На рисунке 2 представлена диаграмма прецедентов для программы.

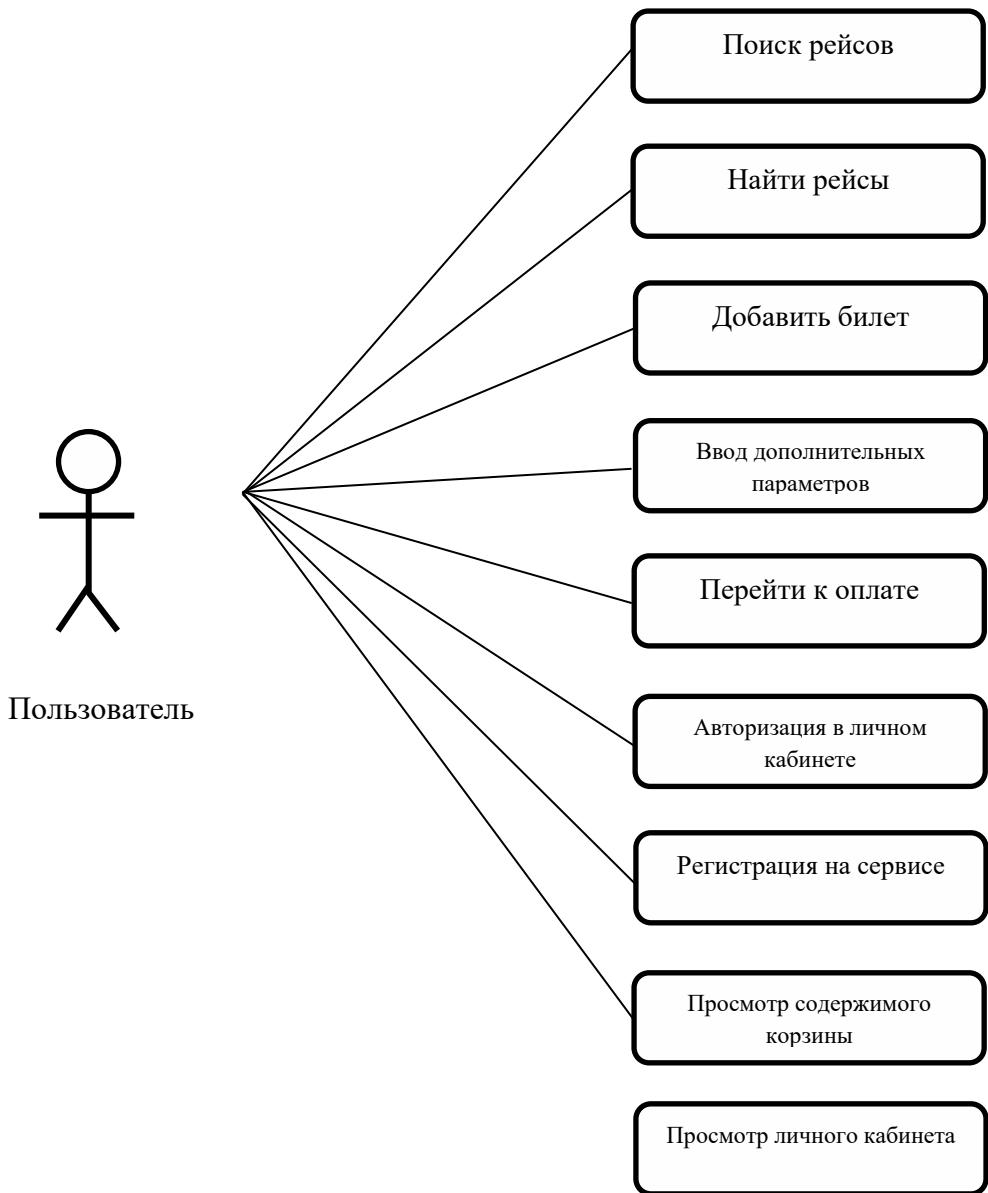


Рисунок 2 - Диаграмма прецедентов

### 1.2.3 Сценарии прецедентов программного изделия

#### **Сценарий для прецедента «Поиск рейсов»**

Основной исполнитель: пользователь.

Заинтересованные лица и их требования: пользователю необходимо задать параметры для поиска рейса.

Предусловие: перед началом работы пользователь должен открыть главную страницу сервиса.

Основной успешный сценарий:

1. Пользователь открыл главную страницу сервиса.
2. Пользователь задал нужные параметры.

### **Сценарий для прецедента «Найти билеты»**

Основной исполнитель: пользователь.

Заинтересованные лица и их требования: пользователю необходимо найти рейсы по заданным параметрам.

Предусловие: перед началом работы пользователь должен открыть главную страницу сервиса и нажать на кнопку “Найти билеты”.

Основной успешный сценарий:

1. Пользователь нажал кнопку “Найти билеты”.
2. Система обработала данные.
3. Система открыла страницу с подобранными рейсами.
4. Пользователь выбирает билет.

### **Сценарий для прецедента «Добавить билет»**

Основной исполнитель: пользователь.

Заинтересованные лица и их требования: пользователю необходимо добавить билет в корзину покупок.

Предусловие: пользователь должен открыть страницу с подобранными рейсами.

Основной успешный сценарий:

1. Пользователь открыл страницу с подобранными рейсами.

2. Пользователь выбрал билет.
3. Пользователь нажал кнопку “Добавить билет”.
4. Система добавила билет в корзину покупок.

### **Сценарий для прецедента «Ввод дополнительных параметров»**

Основной исполнитель: пользователь.

Заинтересованные лица и их требования: пользователю необходимо выбрать дополнительные параметры.

Предусловие: пользователь должен открыть корзину покупок и выбрать дополнительные параметры билета.

Основной успешный сценарий:

1. Пользователь открыл корзину покупок.
2. Пользователь выбрал дополнительный параметр, который хочет изменить.
3. Пользователь поставил галочку рядом с выбранным параметром.

### **Сценарий для прецедента «Перейти к оплате»**

Основной исполнитель: пользователь.

Заинтересованные лица и их требования: пользователю необходимо оплатить билет.

Предусловие: пользователь должен нажать кнопку “Перейти к оплате”.

Основной успешный сценарий:

1. Пользователь открыл корзину покупок.
2. Пользователь нажал кнопку “Перейти к оплате”.
3. Пользователь проверил данные рейса.
4. Пользователь ввёл необходимые данные для оплаты.

5. Пользователь нажал кнопку “Получить код”.
6. Система выслала код пользователю.
7. Пользователь ввёл код.
8. Система проверила код и провела оплату.

### **Сценарий для прецедента «Авторизация в личном кабинете»**

Основной исполнитель: пользователь.

Заинтересованные лица и их требования: пользователю необходимо перейти в личный кабинет

Предусловие: пользователь должен нажать на значок личного кабинета.

Основной успешный сценарий:

1. Пользователь нажал на значок личного кабинета.
2. Пользователь был зарегистрирован ранее и нажал на кнопку “Войти”.
3. Пользователь ввёл номер телефона.
4. Пользователь ввёл пароль от личного кабинета.
5. Система проверила пароль и выполнила вход в личный кабинет.

### **Сценарий для прецедента «Регистрация на сервисе»**

Основной исполнитель: пользователь.

Заинтересованные лица и их требования: пользователю необходимо зарегистрироваться на сервисе

Предусловие: пользователь должен нажать на значок личного кабинета.

Основной успешный сценарий:

1. Пользователь нажал на значок личного кабинета.
2. Пользователь выбрал регистрация на сервисе.

3. Пользователь ввёл данные, запрашиваемые системой.
4. Система отправила пользователю проверочный код для регистрации
5. Пользователь ввёл проверочный код.
6. Система проверила введённый код и зарегистрировала пользователя.

### **Сценарий для прецедента «Просмотр содержимого корзины»**

Основной исполнитель: пользователь.

Заинтересованные лица и их требования: пользователю необходимо перейти в корзину покупок.

Предусловие: пользователь должен нажать на значок корзины покупок.

Основной успешный сценарий:

1. Пользователь нажал на значок корзины покупок.
2. Система открыла корзину покупок, в которой находятся добавленные пользователем рейсы.
3. Пользователь выбрал нужный рейс и перешел к оплате.

### **Сценарий для прецедента «Просмотр личного кабинета»**

Основной исполнитель: пользователь.

Заинтересованные лица и их требования: пользователю необходимо перейти в личный кабинет.

1. Пользователь нажал на кнопку “Личный кабинет”.
2. Система открыла личный кабинет, в котором находятся персональные данные пользователя.
3. Пользователь внес изменения в персональные данные и сохранил их, вышел из личного кабинета или открыл другую страницу.

4. Программа сохранила изменения данных и обновила их в файле, выполнила выход из личного кабинета или открыла другую страницу, выбранную пользователем.

#### **1.2.4 Словарь предметной области приложения**

На основе анализа описания предметной области были определены основные понятия и разработан словарь предметной области, представленный в таблице 1.

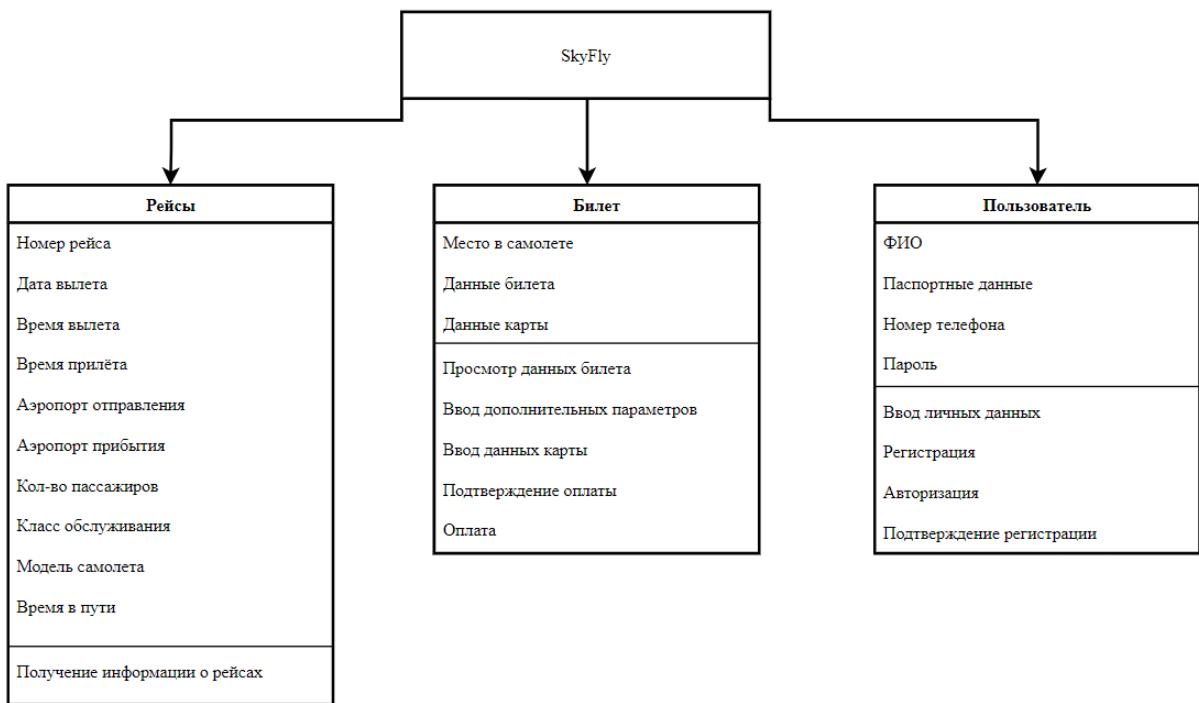
Таблица 1. словарь предметной области

Рейс	Плановый полет самолета между двумя аэропортами с указанием даты и времени вылета и прилета.
Авиакомпания	Организация, предоставляющая авиаперевозки и продающая билеты на свои рейсы.
Класс обслуживания	Категория места в самолете (эконом, бизнес, первый класс).
Багаж	Вещи, которые пассажир берет с собой в полет.
Дополнительные параметры	Услуги, которые пассажир может заказать дополнительно к авиабилету (выбор места, возврат или обмен билета).

Оплата	Процесс оплаты авиабилета через платежные системы.
База данных	Система хранения информации о рейсах, ценах, доступности мест и других данных.
Сигнал	Материальное воплощение сообщения для использования при передаче, переработки и хранения информации.
Авторизация	Процесс проверки пользователя и предоставления ему доступа сервису.
Регистрация	Процесс создания нового личного кабинета пользователя и задание пароля.
Интересные направления	Места, которые пользуются большим спросом среди туристов, отличаются своей природой и культурой.
Проверочный код	Код, который система генерирует и высылает для обеспечения безопасности данных пользователя.

### **1.2.5 Моделирование предметной области с помощью диаграммы классов**

На основе анализа словаря предметной области приложения и описания вариантов использования были определены классы, атрибуты и методы классов и отношения классов, представленные на рисунке 3 в виде диаграммы концептуальных классов.



### Рисунок 3 – Диаграмма концептуальных классов предметной области приложения

## **2 Техническое задание**

### **2.1. Основание для разработки**

Основанием для разработки программного продукта служит задание по курсовой работе по дисциплине «Языки объектно-ориентированного программирования».

### **2.2 Назначение разработки**

Программный продукт предназначен для улучшения понимания работы сервиса для продажи авиабилетов.

### **2.3 Требования к программе или программному изделию**

#### **2.3.1 Требования к функциональным характеристикам**

В программном продукте должен быть представлен один пользователь.

В разрабатываемом программном продукте для пользователя должны быть реализованы следующие функциональные требования.

1. Регистрация пользователя в сервисе.
2. Авторизация пользователя в сервисе.
3. Ввод параметров рейса.
4. Выбор рейса пользователем.
5. Добавление билета в корзину покупок.
6. Удаление билет из корзины покупок.
7. Ввод дополнительных параметров рейса.
8. Оплата билета.

9. Просмотр личного кабинета.
10. Изменение данных в личном кабинете.
11. Выход из личного кабинета.
12. Просмотр купленного билета.

### **2.3.2 Требования к входным и выходным данным**

Формирование входных данных происходит через нажатие определенных клавиш на клавиатуре и компьютерной мышки.

### **2.3.3 Требования пользователя к интерфейсу**

Для начала работы пользователя с программой был разработан интерфейс, со всеми необходимыми страницами для пользователя. Макет интерфейса представлен на рисунках 4-12.

В центральной части стартовой страницы приложения расположен логотип сервиса для продажи авиабилетов. В правом нижнем углу страницы расположена кнопка «Далее», предназначенная для перехода пользователя к главной странице сервиса.



Рисунок 4 - Макет интерфейса стартовой страницы программы

В верхнем левом углу главной страницы приложения находится кнопка с названием сервиса для продажи авиабилетов «SkyFly» (данная кнопка предназначены для возвращения пользователя на главную страницу сервиса). Также в верхней части страницы под кнопкой «SkyFly» находится текстовое поле с кратким описанием сервиса. В правом верхнем углу расположены три кнопки с графическими обозначениями (данные кнопки предназначены для завершения работы программы, перехода пользователя в корзину покупок и для перехода на страницу для авторизации или регистрации). В центральной части страницы расположены подписанные поля ввода данных рейса («Откуда», «Куда», «Дата вылета» и «Тип класса»), а также кнопка «Найти рейсы» (данная кнопка предназначена для поиска рейсов по введённым параметрам).

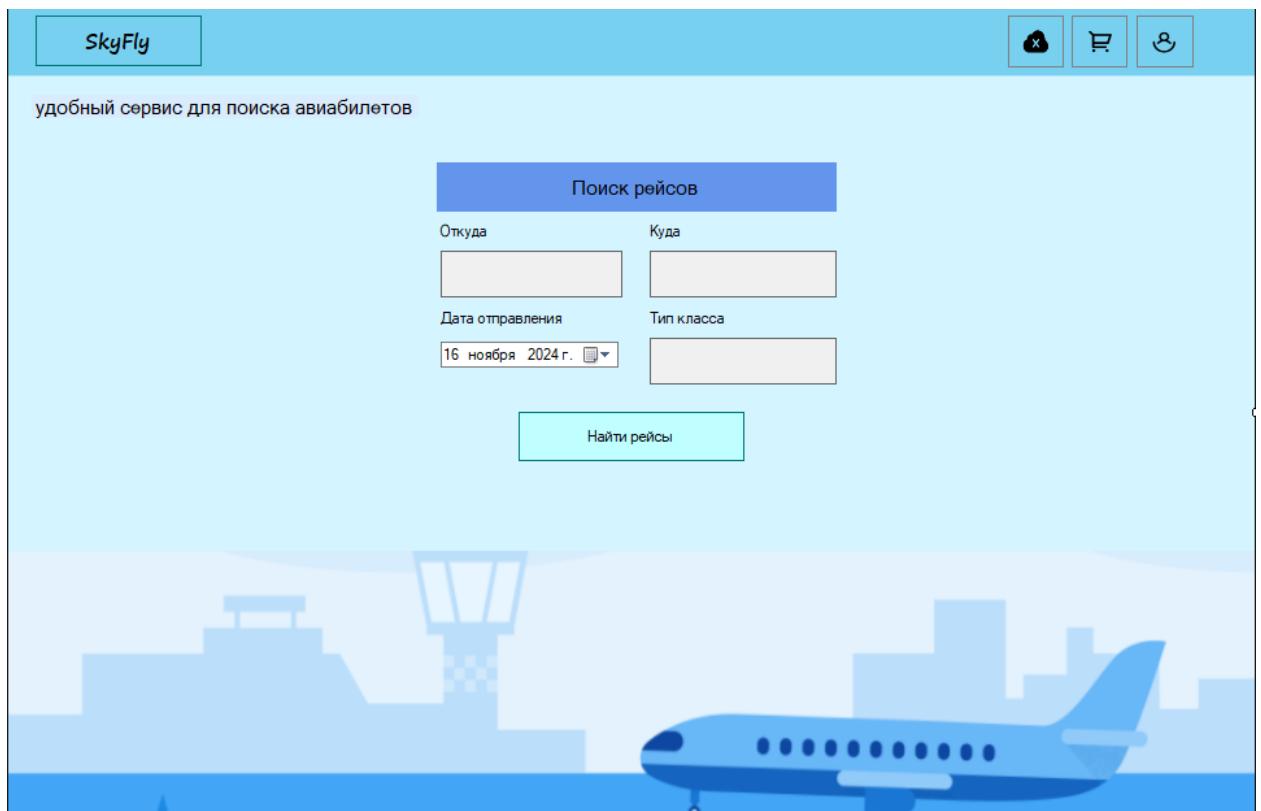


Рисунок 5 - Макет интерфейса главной страницы

В верхнем левом углу страницы подобранных рейсов находится кнопка с названием сервиса для продажи авиабилетов «SkyFly» (данная кнопка предназначены для возвращения пользователя на главную страницу сервиса). Также в верхней части страницы под кнопкой «SkyFly» находится текстовое поле с кратким описанием сервиса. В правом верхнем углу расположены три кнопки с графическими обозначениями (данные кнопки предназначены для завершения работы программы, перехода пользователя в корзину покупок и для перехода на страницу для авторизации или регистрации). В центральной части экрана расположен шаблон для подобранных рейсов. В шаблоне имеются: поле «Рейс; поле «Самолет»; поле «Тип класса»; поле с маршрутом рейса; поле с названиями аэропортов; поле с датой рейса; поле с временем отправления и прибытия; поле с длительностью маршрута; поля с дополнительными параметрами (ручная кладь, без возврата, без багажа); поле с ценой рейса; кнопка «Добавить» (позволяет добавить билет в корзину покупок). Все поля система заполняет автоматически, основываясь на введенных данных пользователем.

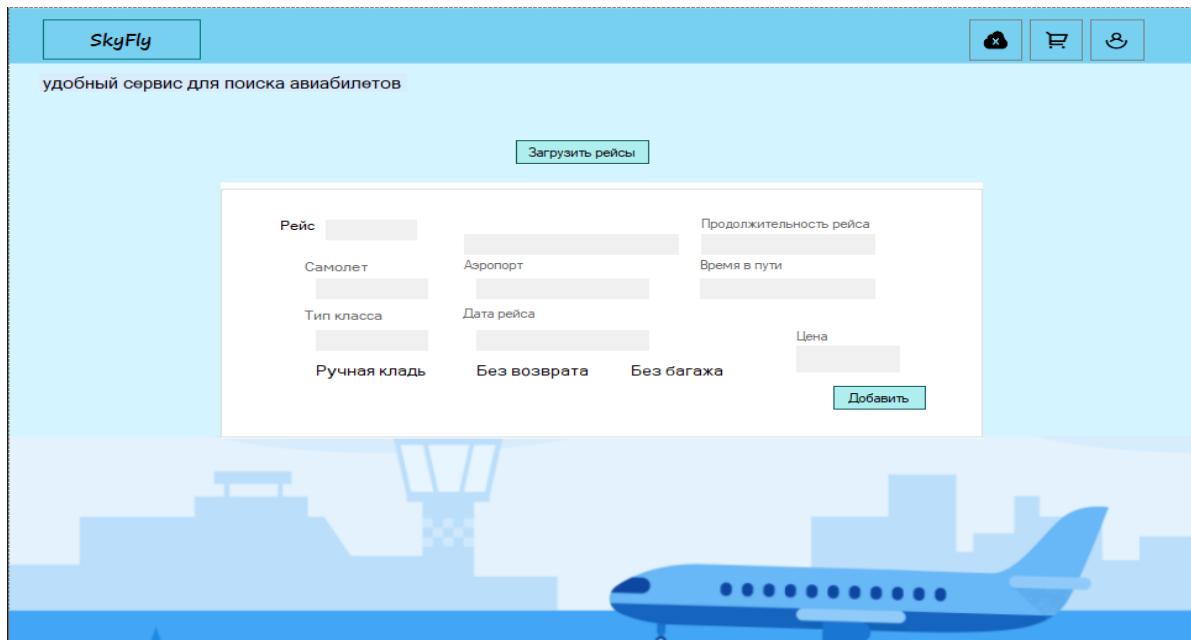


Рисунок 6 - Макет интерфейса страницы подобранных рейсов

В верхнем левом углу страницы корзины покупок находится кнопка с названием сервиса для продажи авиабилетов «SkyFly» (данная кнопка предназначены для возвращения пользователя на главную страницу сервиса). Также в верхней части страницы под кнопкой «SkyFly» находится текстовое поле с кратким описанием сервиса. В правом верхнем углу расположены три кнопки с графическими обозначениями (данные кнопки предназначены для завершения работы программы, перехода пользователя в корзину покупок и для перехода на страницу для авторизации или регистрации). В центральной части страницы расположен шаблон для билета. В шаблоне имеются: поле «Рейс; поле «Самолет»; поле «Тип класса»; поле с маршрутом рейса; поле с названиями аэропортов; поле с датой рейса; поле с временем отправления и прибытия; поле с длительностью маршрута; поля с дополнительными параметрами (ручная кладь, без возврата, без багажа); поле с ценой рейса; поле с местом в самолете (генерируется автоматически или выбирается пользователем), поле для добавления багажа, поле для добавления билету функции обмен/возврат, кнопка «Добавить» (позволяет добавить билет в корзину покупок).

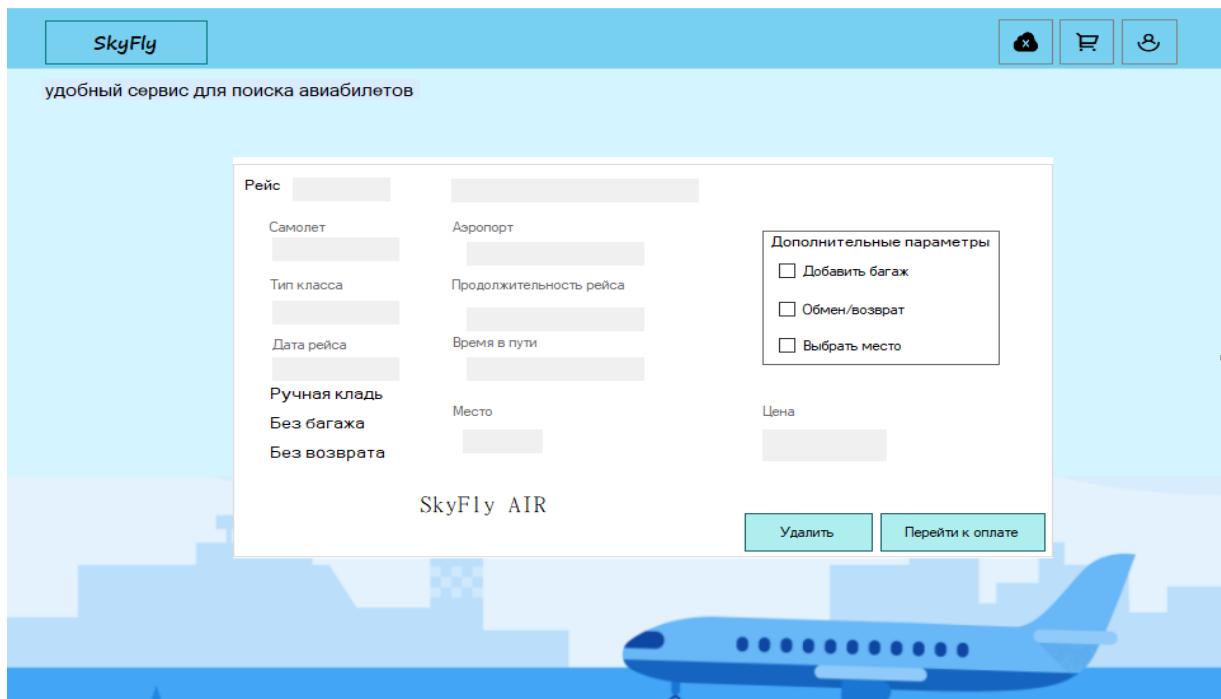


Рисунок 7 - Макет интерфейса страницы корзины покупок

В верхнем левом углу страницы выбора места в самолете находится кнопка с названием сервиса для продажи авиабилетов «SkyFly» (данная кнопка предназначены для возвращения пользователя на главную страницу сервиса). Также в верхней части страницы под кнопкой «SkyFly» находится текстовое поле с кратким описанием сервиса. В правом верхнем углу расположены три кнопки с графическими обозначениями (данные кнопки предназначены для завершения работы программы, перехода пользователя в корзину покупок и для перехода на страницу для авторизации или регистрации). Ниже находится изображение с расположением мест в самолете. При выборе места программа проверяет соответствие выбранного места с выбранным типом класса, если они совпадают, то программа сохраняет и обновляет данные.

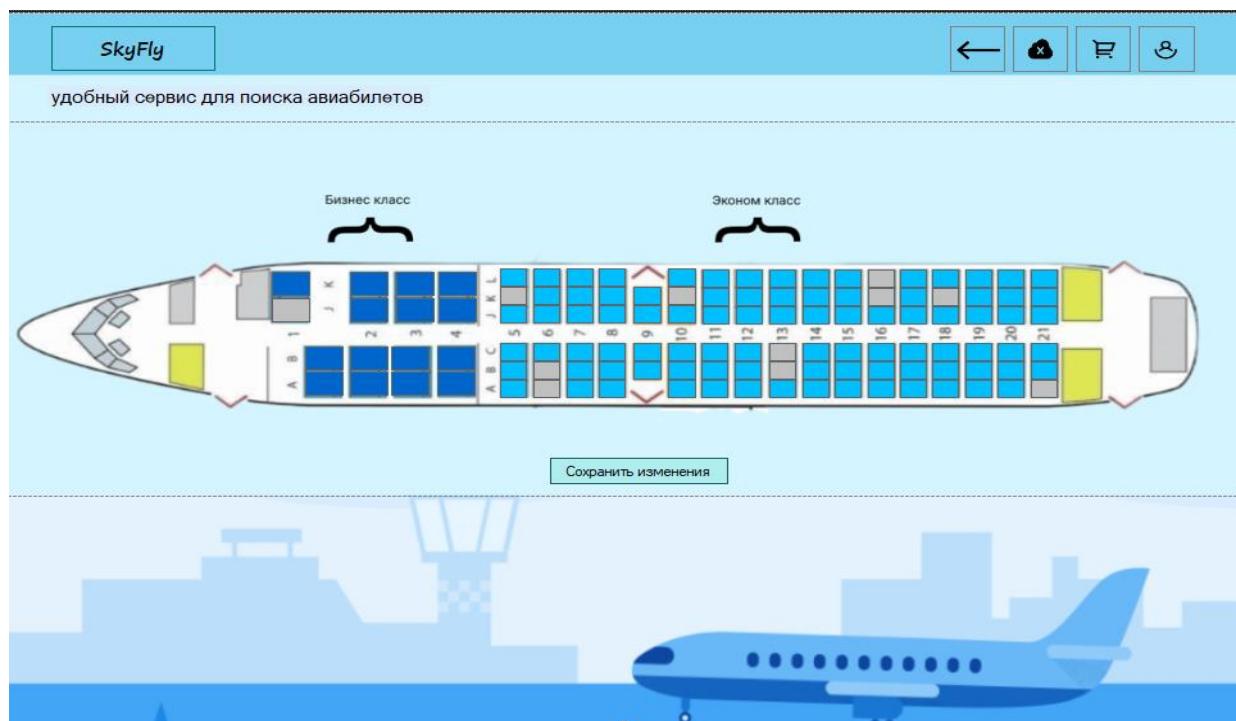


Рисунок 8 - Макет интерфейса страницы выбора места в самолете

В верхнем левом углу страницы для регистрации находится кнопка с названием сервиса для продажи авиабилетов «SkyFly» (данная кнопка предназначены для возвращения пользователя на главную страницу сервиса). В верхней части страницы под кнопкой «SkyFly» находится текстовое поле с кратким описанием сервиса. В правом верхнем углу расположены три кнопки с графическими обозначениями (данные кнопки предназначены для завершения работы программы, перехода пользователя в корзину покупок и для перехода на страницу для авторизации или регистрации). Ниже находятся кнопки «Личный кабинет» и «Авторизация» (позволяют пользователю переходить на страницу для авторизации или на страницу личного кабинета). В левой части страницы находятся поля «ФИО», «Паспортные данные», «Номер телефона», «Пароль». В правой части экрана находится поле для ввода проверочного кода, кнопка «Получить код» (при нажатии на нее система проверяет заполнение всех полей и правильность их

заполнения и отправляет пользователю код) и кнопка «Ввести код» (при нажатии на нее программа проверяет совпадение кодов, в случае совпадения регистрирует пользователя и сохраняет введенные данные).

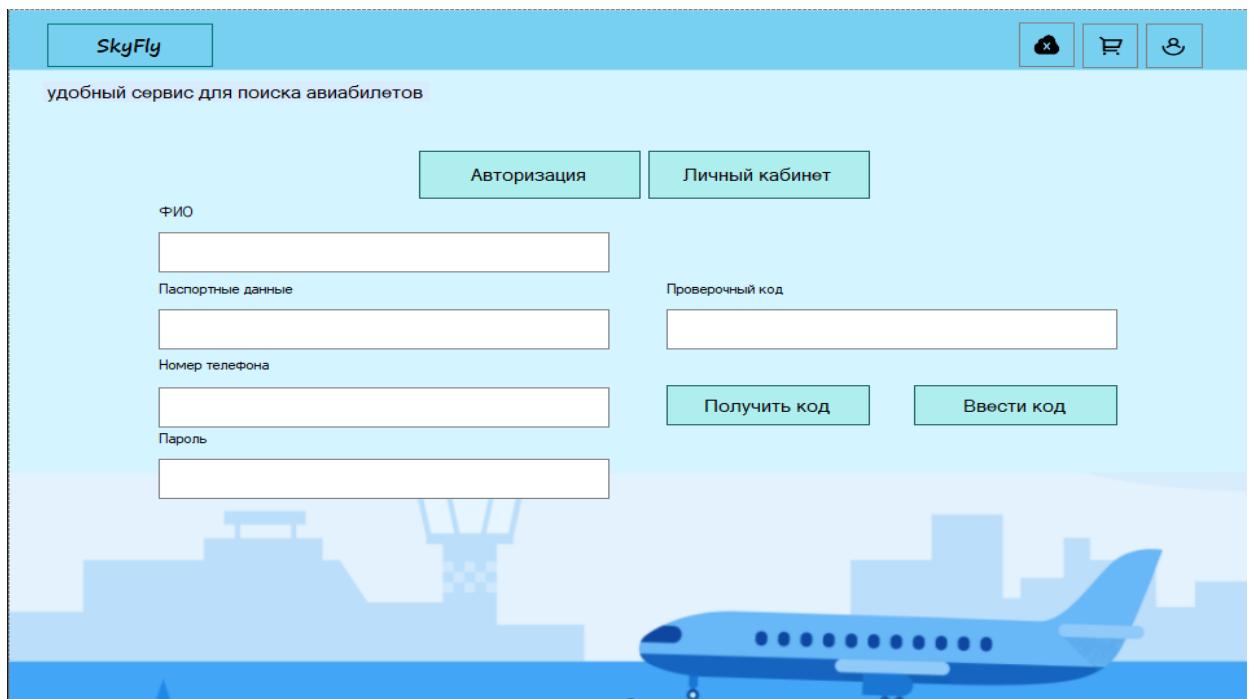


Рисунок 9 - Макет интерфейса страницы для регистрации в сервисе

В верхнем левом углу страницы для авторизации находится кнопка с названием сервиса для продажи авиабилетов «SkyFly» (данная кнопка предназначены для возвращения пользователя на главную страницу сервиса). Также в верхней части страницы под кнопкой «SkyFly» находится текстовое поле с кратким описанием сервиса. В правом верхнем углу расположены три кнопки с графическими обозначениями (данные кнопки предназначены для завершения работы программы, перехода пользователя в корзину покупок и для перехода на страницу для авторизации или регистрации). Ниже находятся кнопки «Регистрация» и «Личный кабинет» (позволяют пользователю переходить на страницу для регистрации или на страницу личного кабинета). В левой части страницы находятся поля «ФИО», «Номер

телефона», «Пароль». В правой части страницы находится поле для ввода проверочного кода (в случае, если пользователь забыл пароль), кнопка «Получить код» (при нажатии на нее система проверяет заполнение всех полей и правильность их заполнения и отправляет пользователю код) и кнопка «Ввести код» (при нажатии на нее программа проверяет совпадение кодов, в случае совпадения авторизует пользователя).

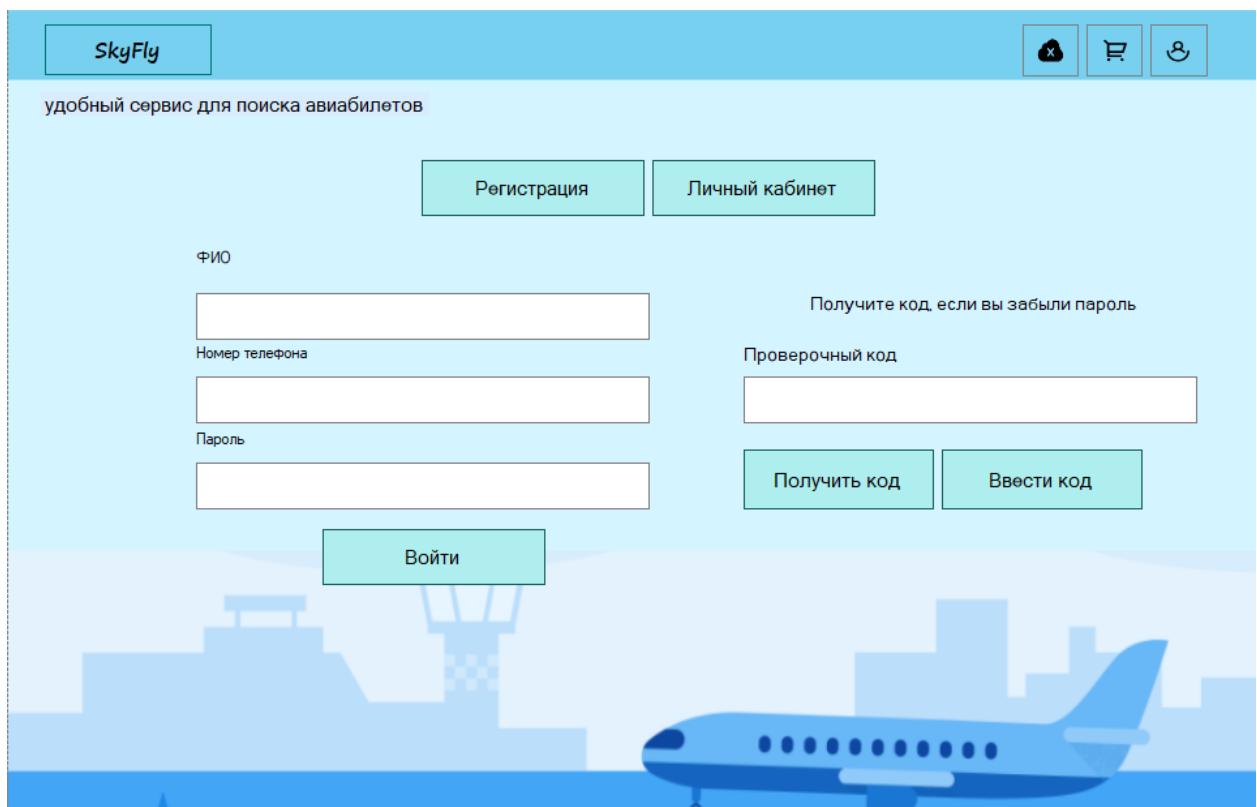


Рисунок 10 - Макет интерфейса страницы для авторизации в сервисе

В верхнем левом углу страницы личного кабинета находится кнопка с названием сервиса для продажи авиабилетов «SkyFly» (данная кнопка предназначены для возвращения пользователя на главную страницу сервиса). Также в верхней части страницы под кнопкой «SkyFly» находится текстовое поле с кратким описанием сервиса. Ниже находятся поля «ФИО», «Паспортные данные», «Номер телефона», «Пароль». В нижней части

страницы находятся кнопки «Сохранить изменения» (при нажатии программа сохраняет изменения и обновляет данные пользователя в файле) и «Выйти из учетной записи» (очищает все поля и выходит из учетной записи).

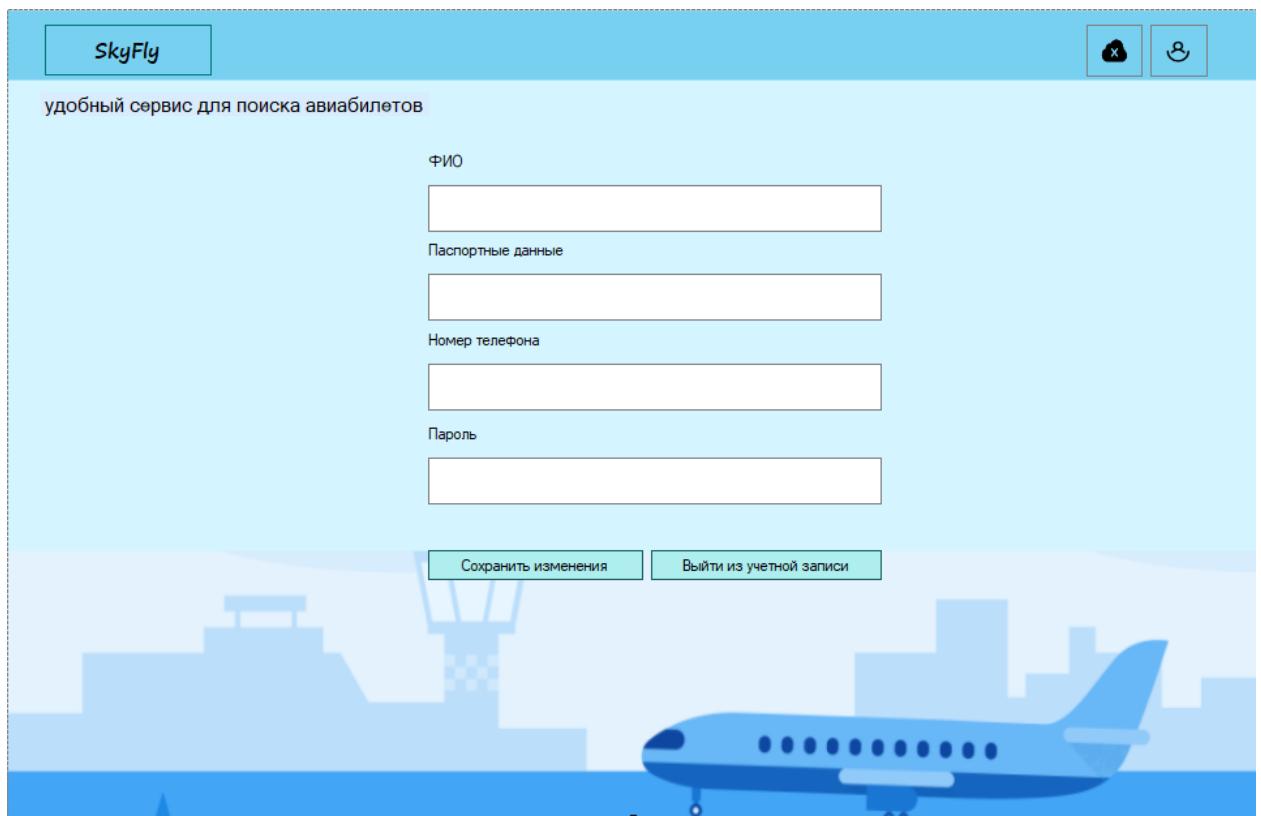


Рисунок 11 - Макет интерфейса страницы личного кабинета

В верхнем левом углу страницы для оплаты билета находится кнопка с названием сервиса для продажи авиабилетов «SkyFly» (данная кнопка предназначены для возвращения пользователя на главную страницу сервиса). Также в верхней части страницы под кнопкой «SkyFly» находится текстовое поле с кратким описанием сервиса. В правом верхнем углу расположены три кнопки с графическими обозначениями (данные кнопки предназначены для завершения работы программы, перехода пользователя в корзину покупок и для перехода на страницу для авторизации или регистрации). В левой части страницы расположен шаблон для билета. В шаблоне имеются: поле «Рейс;

поле «Самолет»; поле «Тип класса»; поле с маршрутом рейса; поле с названиями аэропортов; поле с датой рейса; поле с временем отправления и прибытия; поле с длительностью маршрута; поле с ценой рейса; поле с местом в самолете (генерируется автоматически или выбирается пользователем). В правой части страницы расположен шаблон для оплаты. В шаблоне находятся: поле «ФИО», поле «Паспортные данные», поле «Номер карты», поле «Срок действия», поле «CVC код», поле «Номер телефона», поле «Проверочный код», кнопка «Получить код» (при нажатии на нее система проверяет заполнение всех полей и правильность их заполнения и отправляет пользователю код) и кнопка «Ввести код» (при нажатии на нее программа проверяет совпадение кодов, в случае совпадения авторизует пользователя).

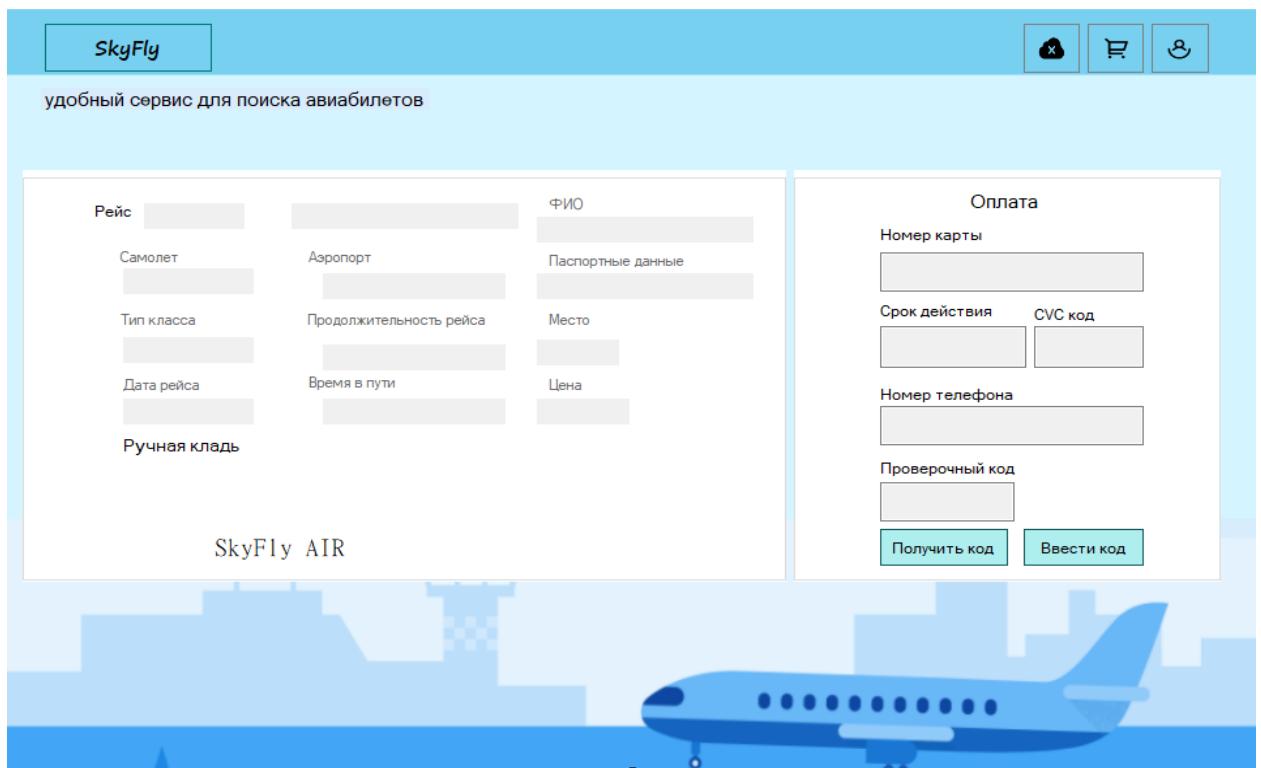


Рисунок 12 - Макет интерфейса страницы для оплаты билета

### **2.3.4 Требования к надежности**

Приложение должно функционировать на всех разработанных тестах. Тесты требуется разработать на этапе рабочего проекта.

### **2.3.5 Условия эксплуатации**

Программное изделие будет эксплуатироваться на персональном компьютере пользователя под управлением операционной системы Windows.

### **2.3.6 Требования к составу и параметрам технических средств**

Для работы клиентского приложения необходимо дисковое пространство не менее 256 Мб, свободная оперативная память в размере не менее 512 Мб, видеокарта с не менее 256 Мб видеопамяти, разрешение экрана не менее 1920\*1080, операционной системы не ниже Windows 7, клавиатура, мышь.

### **2.3.7 Требования к информационной и программной совместимости**

Для разработки программного продукта необходимы использовать:

- язык C# – для разработки игры.
- Microsoft Visual Studio не ниже версии 2022 – для написания кода игры.

Программное изделие должно работать в операционных системах Windows с установленным .NET Framework версии не ниже 4.6.1. Для переноса программы не должны требоваться специальные аппаратные и программные средства.

## **2.4 Требования к программной документации**

Разработка программной документации и программного изделия должна производиться согласно СТУ 02.030–2023 «Курсовые работы (проекты). Выпускные квалификационные работы. Общие требования к структуре и оформлению» ЮЗГУ, ГОСТ 19.001-77, ГОСТ Р 7.0.100–2018.

## **2.5 Стадии и этапы разработки**

Выполнение разработки должно включать три стадии:

- техническое задание;
- технический проект;
- рабочий проект.

На стадии «Техническое задание» проводится постановка задачи, анализ данной предметной области, разработка требований к программному изделию, изучение литературы по задаче и оформление документа «Техническое задание».

На стадии «Технический проект» проводится проектирование программной системы. В заключение данного этапа оформляется документ "Технический проект".

На стадии «Рабочий проект» проводится реализация программной системы. В заключение данного этапа оформляется документ «Рабочий проект».

## **2.6 Порядок контроля и приемки**

Приемка программного изделия осуществляется при сдаче документально оформленных этапов разработки и проведении испытаний на

основе установленных тестов. Тесты должны быть предоставлены поставщиком и согласованы с заказчиком.

### 3 Технический проект

#### 3.1 Моделирование последовательности действий

Диаграмма активности для программы представлена на рисунке 9.

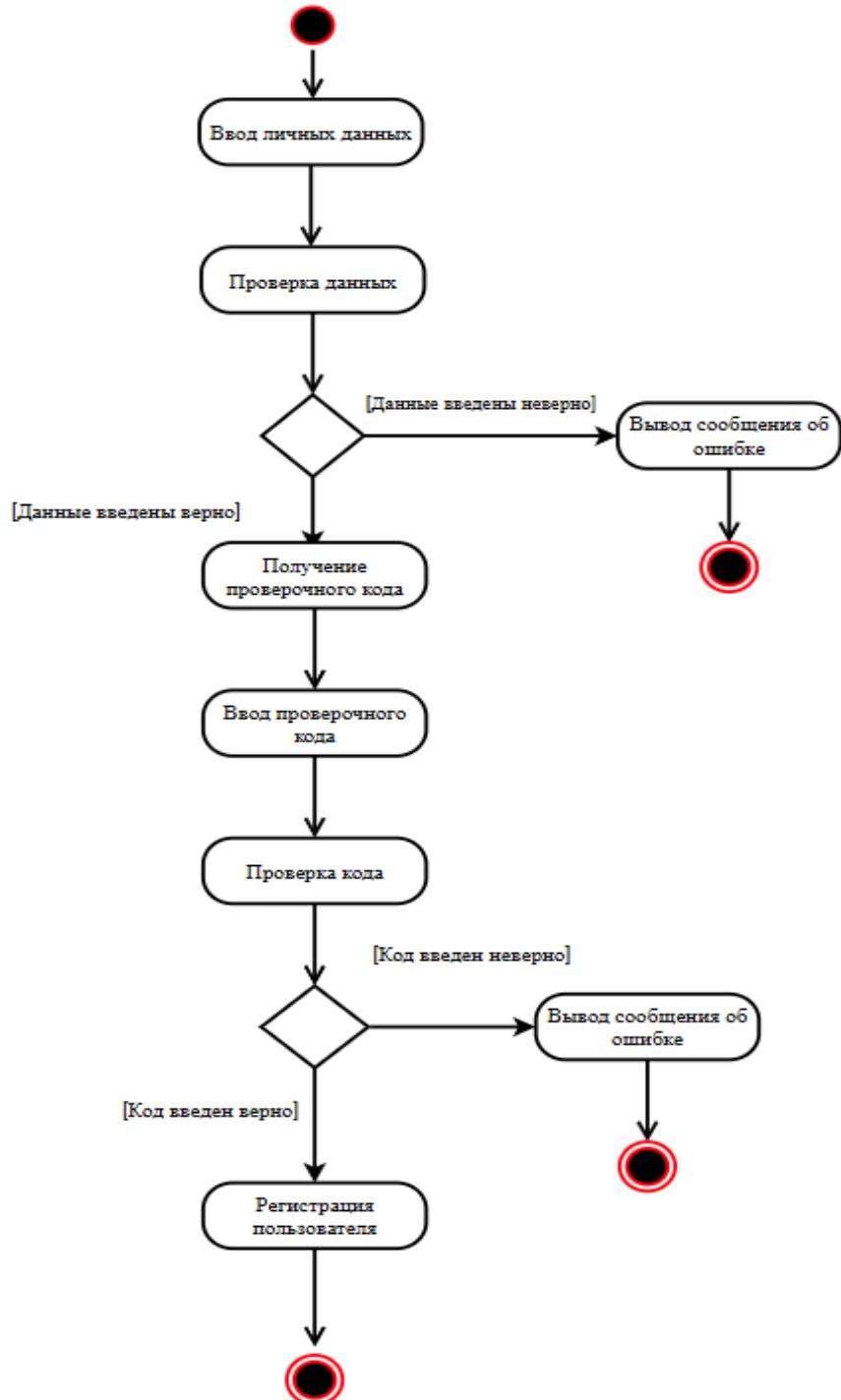


Рисунок 13 - Диаграмма активности

### 3.2 Проектирование архитектуры программной системы

На рисунке 10 представлена диаграмма компонентов разрабатываемой программы.

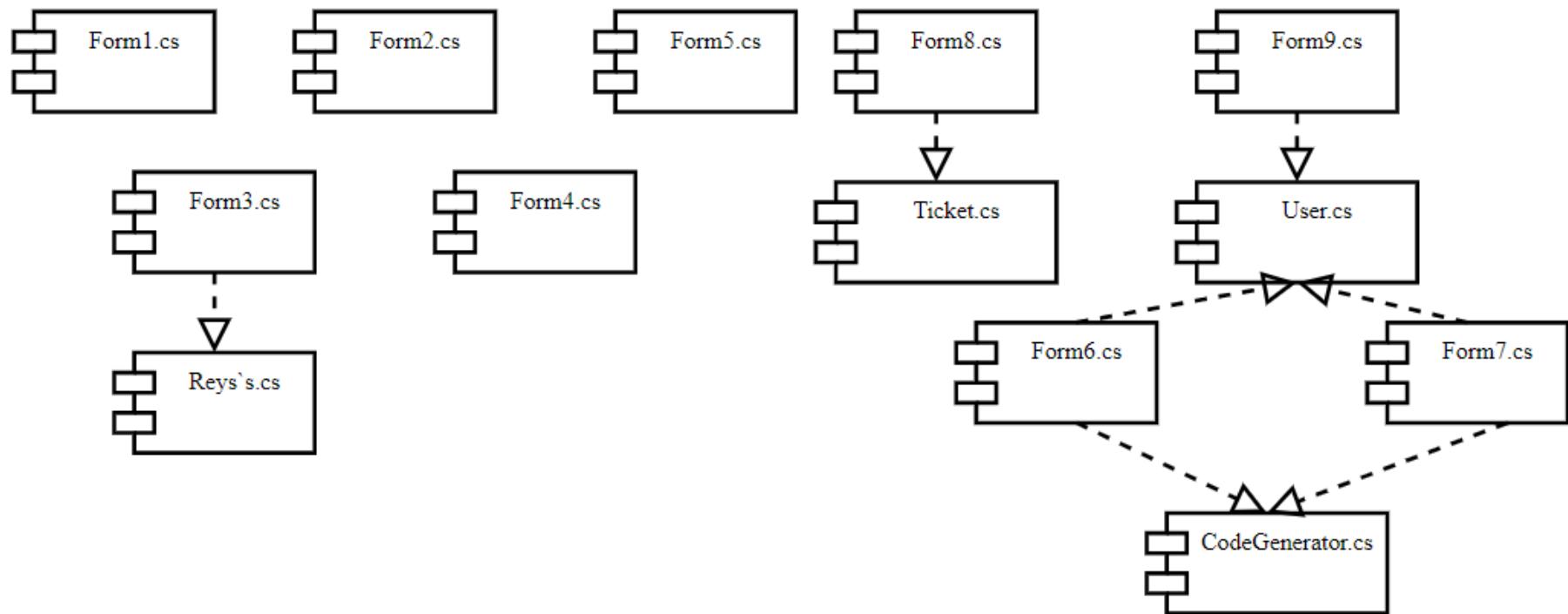


Рисунок 14 – Диаграмма компонентов разрабатываемой программой

### **3.3 Описание структур и форматов данных**

Программа основана на взаимодействии программных классов с данными, сохраненными на локальном диске пользователя – фото, звуковыми файлами. В программе используются следующие форматы файлов для:

- текстовых файлов - .txt

### **3.4 Схемы алгоритмов**

В ПРИЛОЖЕНИИ Б представлены блок-схема алгоритма выполнения функции GenerateNomEr1 и ValidateInput.

## **4.Рабочий проект**

### **4.1 Спецификация компонентов и классов программ**

#### **4.1.1 Модули и объекты интерфейса пользователя**

Пользовательский интерфейс включает в себя следующие компоненты:

- Стартовая страница приложения;
- Главная страница приложения;
- Страница подобранных рейсов;
- Страница корзины покупок;
- Страница выбора места в самолёте;
- Страница регистрации;
- Страница авторизации;
- Страница оплаты билета;
- Страница личного кабинета;
- Страница купленного билета.

#### **4.1.2 Описание объектов интерфейса программы**

На основе макета интерфейса в разделе 1.3.5 технического задания, с помощью языка программирования C#, IDE Visual Studio 2022 была создана программа для моделирования сервиса для продажи авиабилетов. На рисунке 15 представлен интерфейс взаимодействия пользователя со стартовой страницей программы. Числами на рисунке обозначены номера объектов интерфейса.



Рисунок 15 – Интерфейс взаимодействия пользователя со стартовой страницей программы

В таблице 2 представлено описание объектов интерфейса взаимодействия пользователя со стартовой страницей программы.

Таблица 2 - Описание объектов интерфейса взаимодействия пользователя с главным меню программы

№	Тип объекта	Имя объекта	Действие/описание объекта
1	Button	button1	Кнопка, при нажатии которой программа открывает главную страницу.
2	PictureBox	pictureBox1	Элемент, который содержит изображение фона программы.

На рисунке 16 представлен интерфейс взаимодействия пользователя с главной страницей. Числами на рисунке обозначены номера объектов интерфейса.

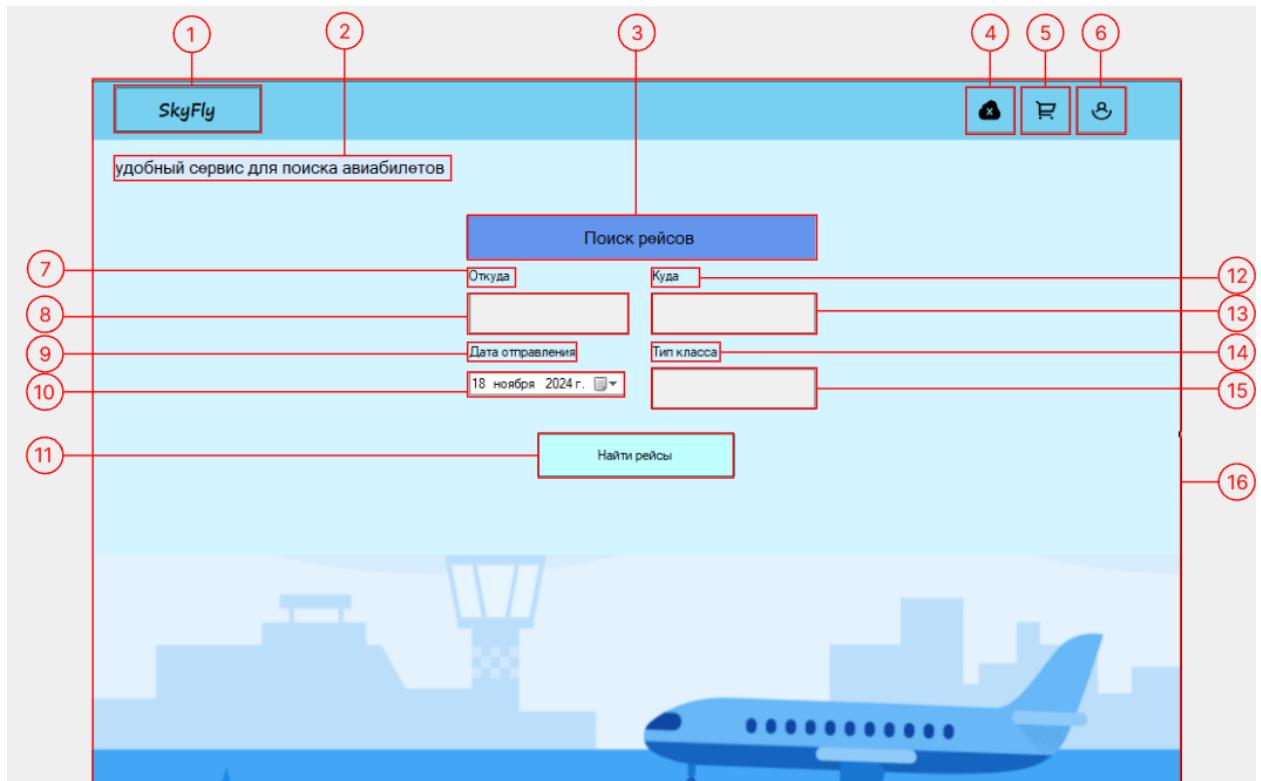


Рисунок 16 - Интерфейс взаимодействия пользователя с главной страницей

В таблице 3 представлено описание объектов взаимодействия пользователя с главной страницей.

Таблица 3 - Описание объектов интерфейса взаимодействия пользователя с главной страницей.

№	Тип объекта	Имя объекта	Действие/описание объекта
1	Button	button1	Кнопка «SkyFly», при нажатии которой открывается главная страница программы.

1	2	3	4
2	Label	label1	Поле, в котором хранится описание сервиса.
3	Label	label2	Поле, в котором хранится название страницы.
4	Button	button8	Кнопка, при нажатии которой завершается работа программы.
5	Button	button2	Кнопка, при нажатии которой программа открывает страницу корзины покупок.
6	Button	button3	Кнопка, при нажатии которой программа открывает страницу для регистрации.
7	Label	label3	Название поля.
8	Label	label7	Поле, при нажатии на которое открывается окно, содержащее список городов.
9	Label	label5	Название поля.
10	DateTimePicker	dateTimePicker1	Календарь, в котором пользователь выбирает дату рейса.
11	Button	button4	Кнопка, при нажатии которой программа сохраняет введенные данные и открывает страницу с подобранными рейсами.
12	Label	label4	Название поля.
13	Label	label8	Поле, при нажатии на которое открывается окно, содержащее список городов.
14	Label	label6	Название поля.
15	Label	label9	Поле, при нажатии на которое открывается окно, содержащее список типов класса в самолете.

1	2	3	4
16	PictureBox	pictureBox1	Элемент, который содержит изображение фона программы.

На рисунке 17 представлен интерфейс взаимодействия пользователя со страницей подобранных рейсов. Числами на рисунке обозначены номера объектов интерфейса.

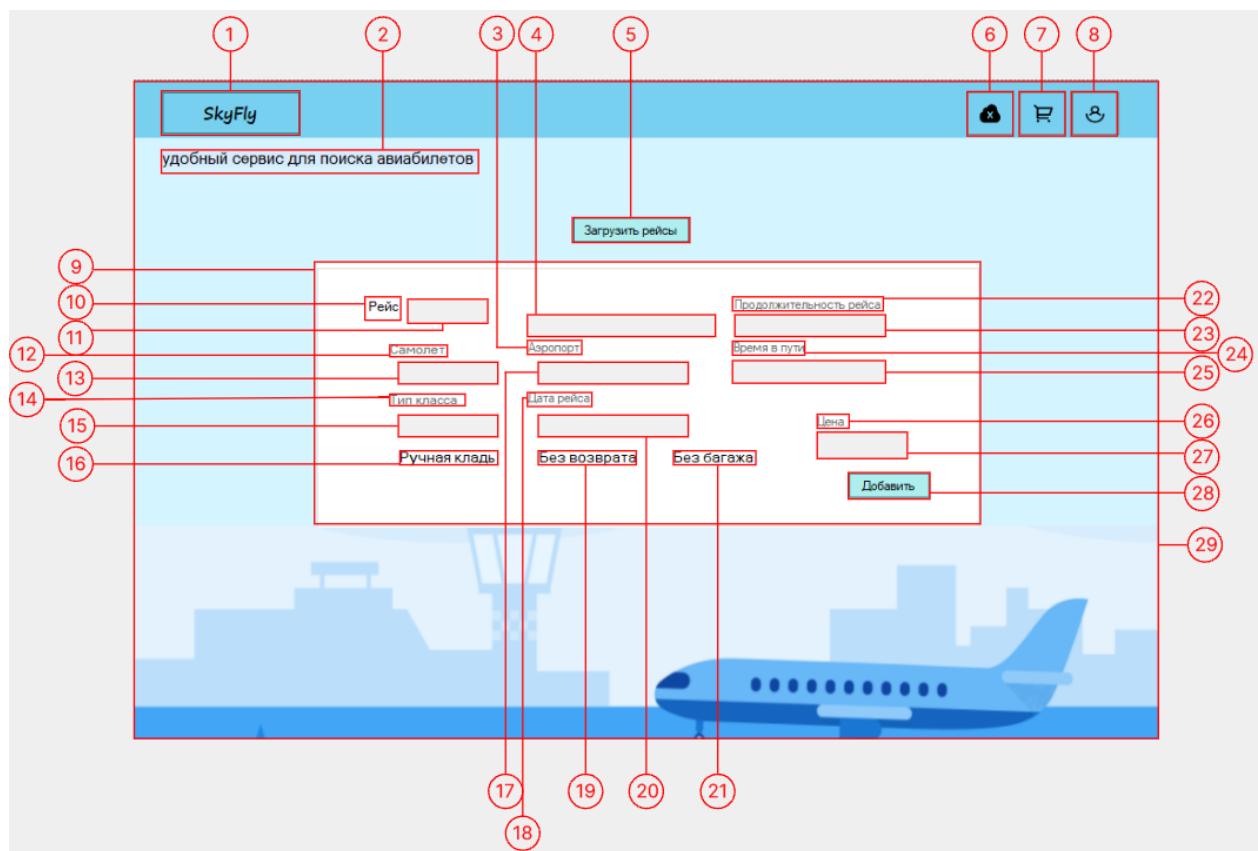


Рисунок 17 - Интерфейс взаимодействия пользователя со страницей подобранных рейсов

В таблице 4 представлено описание объектов интерфейса взаимодействия пользователя со страницей подобранных рейсов.

Таблица 4 - Описание объектов интерфейса взаимодействия пользователя со страницей подобранных рейсов.

№	Тип объекта	Имя объекта	Действие/описание объекта
1	Button	button1	Кнопка «SkyFly», при нажатии которой открывается главная страница программы.
2	Label	label1	Поле, в котором хранится описание сервиса.
3	Label	label37	Название поля.
4	TextBox	textBox4	Поле, в которое программа вносит наименование рейса (город отправления – город прибытия).
5	Button	button6	Кнопка «Загрузить рейсы», при нажатии которой программа считывает данные из файла, а также генерирует сама и вносит их в определенные textbox.
6	Button	button8	Кнопка, при нажатии которой завершается работа программы.
7	Button	button2	Кнопка, при нажатии которой программа открывает страницу корзины покупок.
8	Button	button3	Кнопка, при нажатии которой программа открывает страницу для регистрации.
9	GroupBox	groupBox1	Контейнер, содержащий в себе несколько полей, в которых содержится информация о рейсе.
10	Label	label2	Название поля.

1	2	3	4
11	TextBox	textBox1	Поле, в которое программа вносит сгенерированный номер рейса.
12	Label	label5	Название поля.
13	TextBox	textBox2	Поле, в которое программа вносит название самолёта, выполняющего рейс.
14	Label	label6	Название поля.
15	TextBox	textBox3	Поле, в которое программа вносит тип класса (хранится в файле), который указал пользователь.
16	Label	label7	Поле, содержащее информацию о типе билета (разрешен ли провоз ручной клади).
17	TextBox	textBox5	Поле, в которое программа вносит названия аэропортов, согласно городу отправления и городу прибытия.
18	Label	label38	Название поля.
19	Label	label8	Поле, содержащее информацию о типе билета (является ли билет возвратным).
20	TextBox	textBox6	Поле, в которое программа вносит указанную пользователем дату рейса (хранится в файле).
21	Label	label9	Поле, содержащее информацию о типе билета (включен ли в стоимость билета провоз багажа).
22	Label	label17	Название поля.

1	2	3	4
23	TextBox	textBox7	Поле, в которое программа вносит сгенерированную продолжительность рейса (время отправления - время прибытия).
24	Label	label34	Название поля.
25	TextBox	textBox8	Поле, в которое программа вносит сгенерированное время в пути (рассчитывается программой исходя из времени отправления и времени прибытия).
26	Label	label40	Название поля.
27	TextBox	textBox9	Поле, в которое программа вносит сгенерированную цену билета.
28	Button	button4	Кнопка, при нажатии которой программа копирует информацию из всех textbox и вносит ее в определенные textbox, находящиеся на странице корзины покупок.
29	PictureBox	pictureBox1	Элемент, который содержит изображение фона программы.

На рисунке 18 представлен интерфейс взаимодействия пользователя со страницей корзины покупок. Числами на рисунке обозначены номера объектов интерфейса.

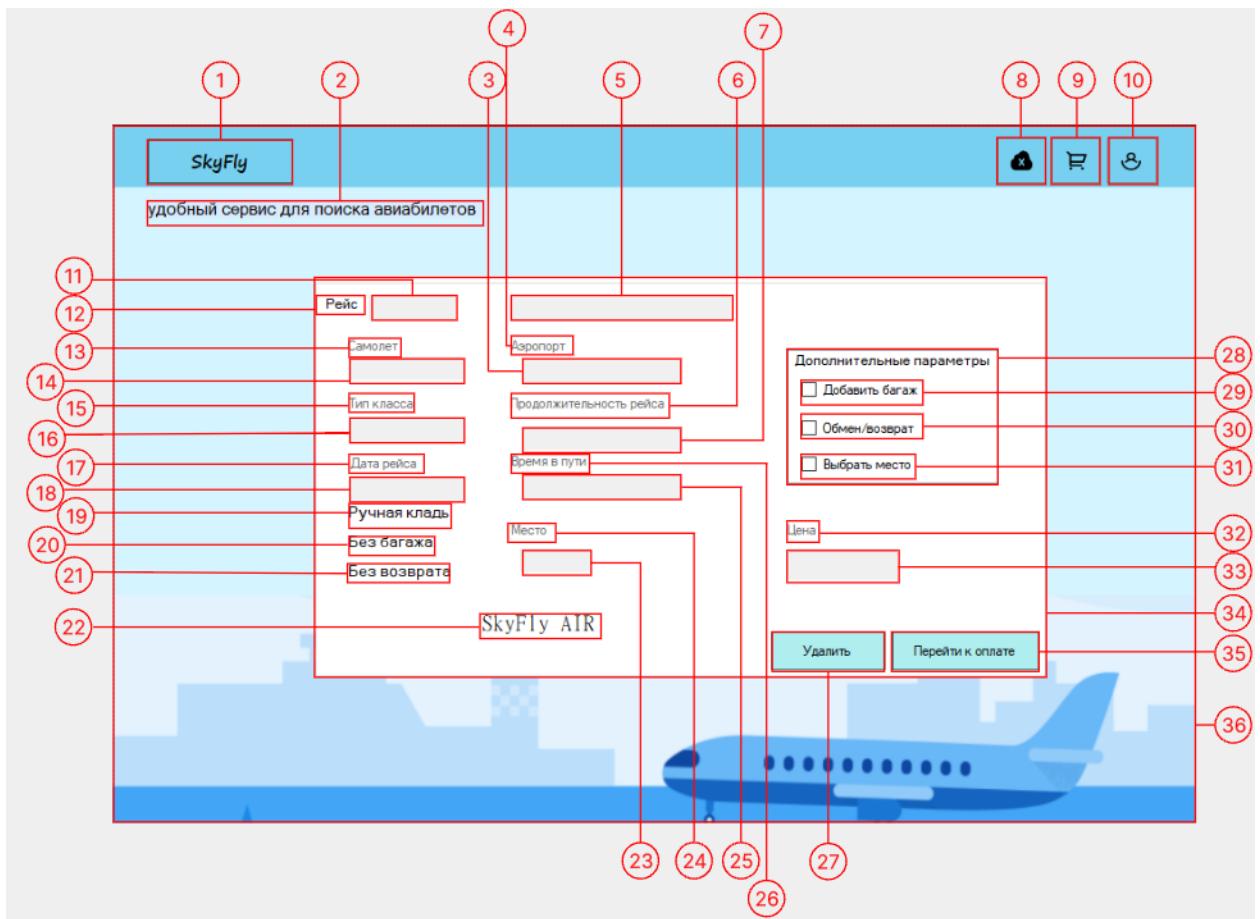


Рисунок 18 - Интерфейс взаимодействия пользователя со страницей корзины покупок

В таблице 5 представлено описание объектов интерфейса взаимодействия пользователя со страницей корзины покупок.

Таблица 5 - Описание объектов интерфейса взаимодействия пользователя со страницей корзины покупок.

№	Тип объекта	Имя объекта	Действие/описание объекта
1	Button	button1	Кнопка «SkyFly», при нажатии которой открывается главная страница программы.

1	2	3	4
2	Label	label1	Поле, в котором хранится описание сервиса.
3	TextBox	textBox6	Поле, в которое программа вносит названия аэропортов.
4	Label	label36	Название поля.
5	TextBox	textBox5	Поле, в которое программа вносит наименование рейса (город отправления – город прибытия).
6	Label	label19	Название поля.
7	TextBox	textBox7	Поле, в которое программа вносит продолжительность рейса (время отправления - время прибытия).
8	Button	button8	Кнопка, при нажатии которой завершается работа программы.
9	Button	button2	Кнопка, при нажатии которой программа открывает страницу корзины покупок.
10	Button	button3	Кнопка, при нажатии которой программа открывает страницу для регистрации.
11	TextBox	textBox1	Поле, в которое программа вносит номер рейса.
12	Label	label33	Название поля.
13	Label	label31	Название поля.
14	TextBox	textBox2	Поле, в которое программа вносит название самолёта, выполняющего рейс.
15	Label	label30	Название поля.
16	TextBox	textBox3	Поле, в которое программа вносит тип класса.
17	Label	label8	Название поля.
18	TextBox	textBox4	Поле, в которое программа вносит дату рейса.

1	2	3	4
19	Label	label29	Поле, содержащее информацию о типе билета (разрешен ли провоз ручной клади).
20	Label	label27	Поле, содержащее информацию о типе билета (включен ли в стоимость билета провоз багажа).
21	Label	label28	Поле, содержащее информацию о типе билета (является ли билет возвратным).
22	Label	label5	Поле, в котором хранится название авиакомпании.
23	Label	label4	Поле, в которое программа вносит сгенерированное или выбранное пользователем место в самолете.
24	Label	label2	Название поля.
25	TextBox	textBox8	Поле, в которое программа вносит значение времени в пути (рассчитывается программой исходя из времени отправления и времени прибытия).
26	Label	label35	Название поля.
27	Button	button4	Кнопка для очистки всех текстовых полей.
28	Label	label6	Поле для обозначения дополнительных параметров билета.
29	CheckBox	checkBox1	Поле для добавления багажа.
30	CheckBox	checkBox2	Поле для обмена\возврата билета.
31	CheckBox	checkBox3	Поле для выбора места в самолете.
32	Label	label41	Название поля.
33	TextBox	textBox9	Поле, в которое программа вносит сгенерированную цену билета.

1	2	3	4
34	GroupBox	groupBox1	Контейнер, содержащий в себе несколько полей, в которых содержится информация о рейсе.
35	Button	button5	Кнопка, при нажатии которой программа проверяет авторизацию пользователя в сервисе, если пользователь авторизован, программа копирует данные из label4, после чего открывает страницу для оплаты билета и вносит скопированные данные в label4.
36	PictureBox	pictureBox1	Элемент, который содержит изображение фона программы.

На рисунке 19 представлен интерфейс взаимодействия пользователя со страницей для выбора места в самолёте. Числами на рисунке обозначены номера объектов интерфейса.

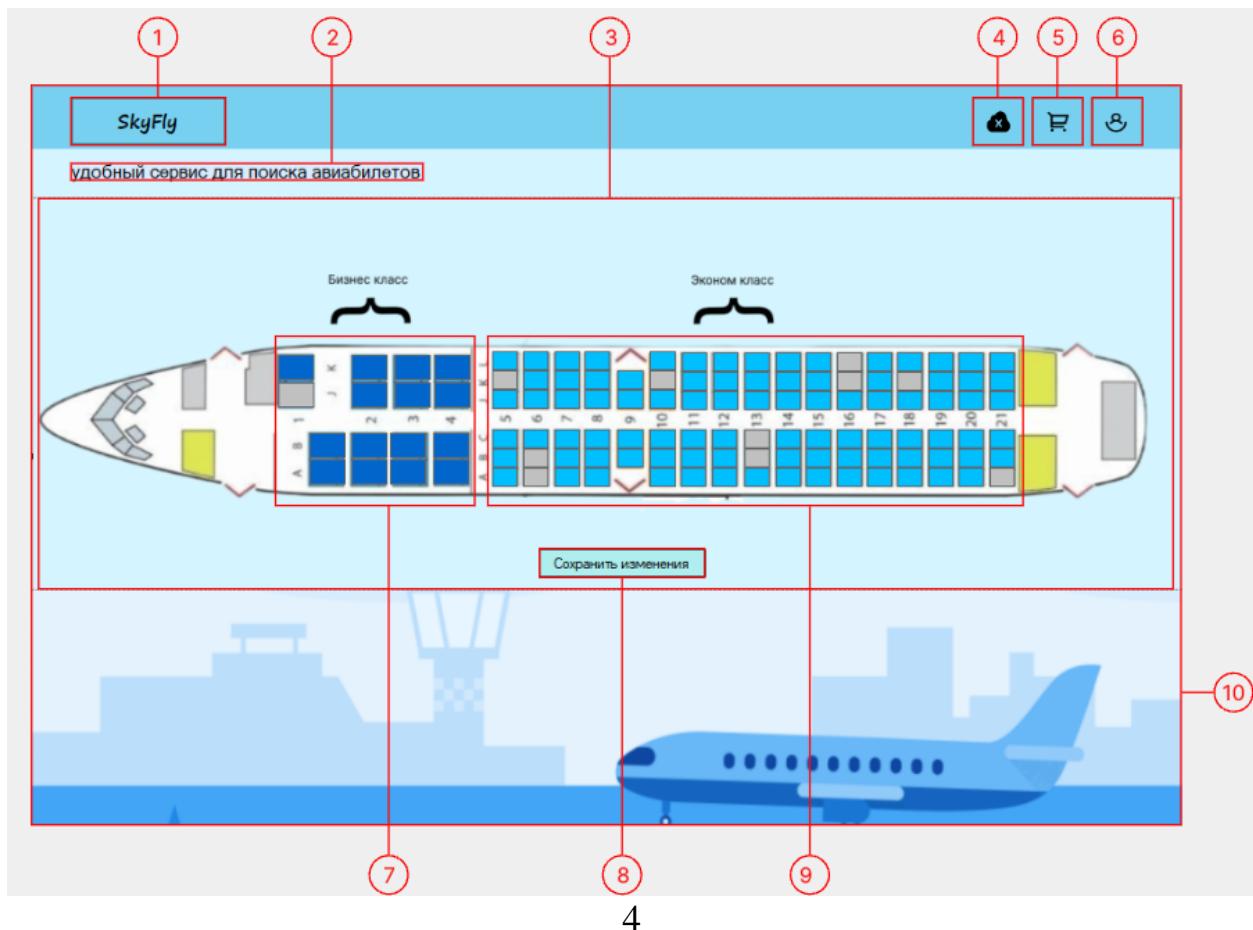


Рисунок 19 - Интерфейс взаимодействия пользователя со страницей для выбора места в самолёте

В таблице 6 представлено описание объектов интерфейса взаимодействия пользователя со страницей для выбора места в самолёте.

Таблица 6 - Описание объектов интерфейса взаимодействия пользователя со страницей для выбора места в самолёте.

№	Тип объекта	Имя объекта	Действие/описание объекта
1	Button	button1	Кнопка «SkyFly», при нажатии которой открывается главная страница программы.

1	2	3	4
2	Label	label1	Поле, в котором хранится описание сервиса.
3	PictureBox	pictureBox1	Элемент, который содержит изображение расположения мест в самолете.
4	Button	button8	Кнопка, при нажатии которой завершается работа программы.
5	Button	button2	Кнопка, при нажатии которой программа открывает страницу корзины покупок.
6	Button	button3	Кнопка, при нажатии которой программа открывает страницу для регистрации.
7	Label	label2-17	Поля, относящиеся к бизнес классу, при нажатии на которые программа считывает значение выбранного поля.
8	Button	button6	Кнопка, при нажатии которой система считывает значение выбранного поля, после чего открывает страницу корзины покупок и вносит в label4 место, которое выбрал пользователь.
9	Label	Label18-117	Поля, относящиеся к эконом классу, при нажатии на которые программа считывает значение выбранного поля.
10	PictureBox	pictureBox2	Элемент, который содержит изображение фона программы.

На рисунке 20 представлен интерфейс взаимодействия пользователя со страницей для регистрации. Числами на рисунке обозначены номера объектов интерфейса.

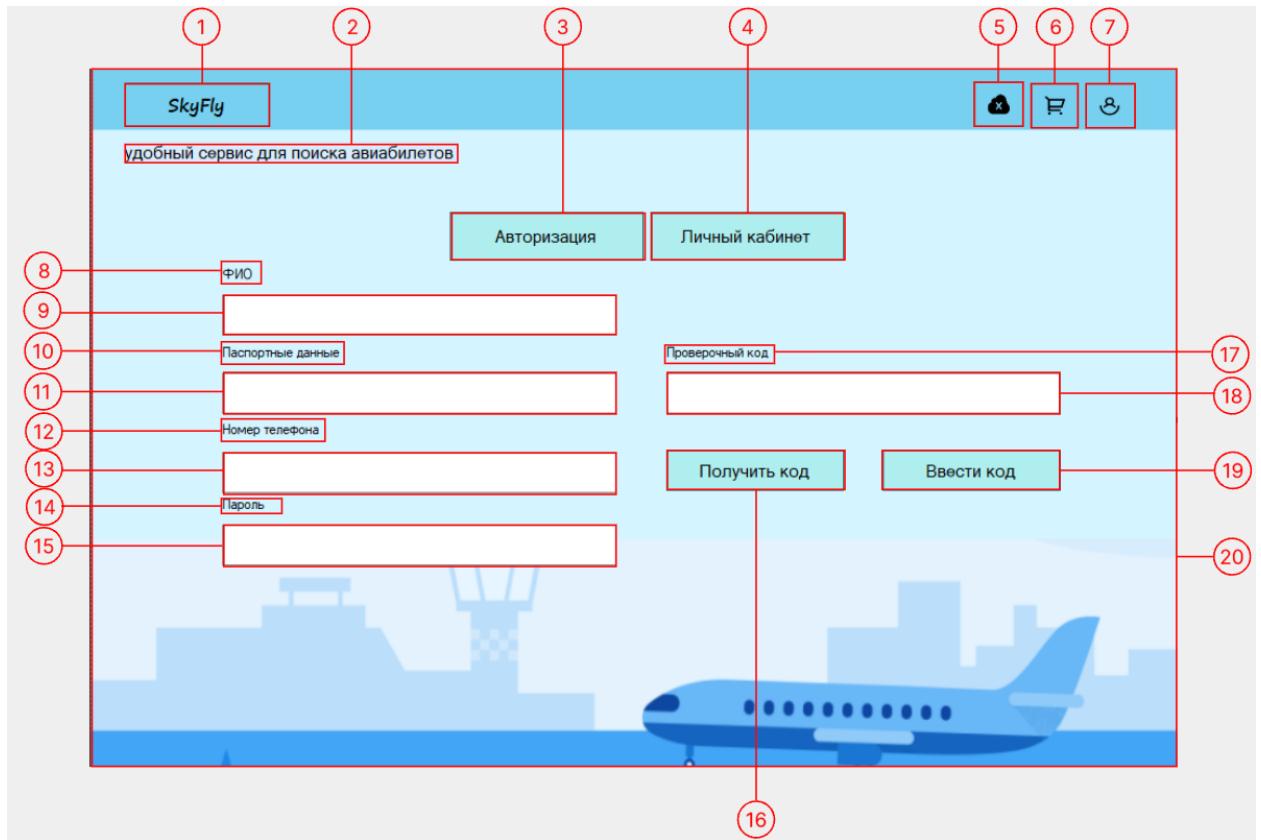


Рисунок 20 - Интерфейс взаимодействия пользователя со страницей для регистрации

В таблице 7 представлено описание объектов интерфейса взаимодействия пользователя со страницей для регистрации.

Таблица 7 - Описание объектов интерфейса взаимодействия пользователя со страницей для регистрации.

№	Тип объекта	Имя объекта	Действие/описание объекта
1	Button	button1	Кнопка «SkyFly», при нажатии которой открывается главная страница программы.
2	Label	label1	Поле, в котором хранится описание сервиса.
3	Button	button5	Кнопка, при нажатии которой программа открывает страницу для авторизации.
4	Button	button2	Кнопка, при нажатии которой программа открывает страницу личного кабинета пользователя.
5	Button	button8	Кнопка, при нажатии которой завершается работа программы.
6	Button	button9	Кнопка, при нажатии которой программа открывает страницу корзины покупок.
7	Button	button3	Кнопка, при нажатии которой программа открывает страницу для регистрации.
8	Label	label2	Название поля.
9	TextBox	textBox1	Поле, в которое пользователь вводит ФИО.
10	Label	label3	Название поля.
11	TextBox	textBox2	Поле, в которое пользователь вводит паспортные данные.
12	Label	label4	Название поля.

1	2	3	4
13	TextBox	textBox3	Поле, в которое пользователь вводит номер телефона.
14	Label	label5	Название поля.
15	TextBox	textBox4	Поле, в которое пользователь вводит пароль.
16	Button	button6	Кнопка, при нажатии которой программа проверяет наличие введенных данных и их корректность, после чего отправляет проверочный код или выводит ошибку о неправильно введенных данных.
17	Label	label6	Название поля.
18	TextBox	textBox5	Поле, в которое пользователь вводит проверочный код.
19	Button	button7	Кнопка, при нажатии на которую программа проверяет введенный код с тем, который был отправлен пользователю и, в случае их совпадения, регистрирует пользователя. Иначе отправляет пользователю уведомление о неправильном коде.
20	PictureBox	pictureBox1	Элемент, который содержит изображение фона программы.

На рисунке 21 представлен интерфейс взаимодействия пользователя со страницей для авторизации. Числами на рисунке обозначены номера объектов интерфейса.

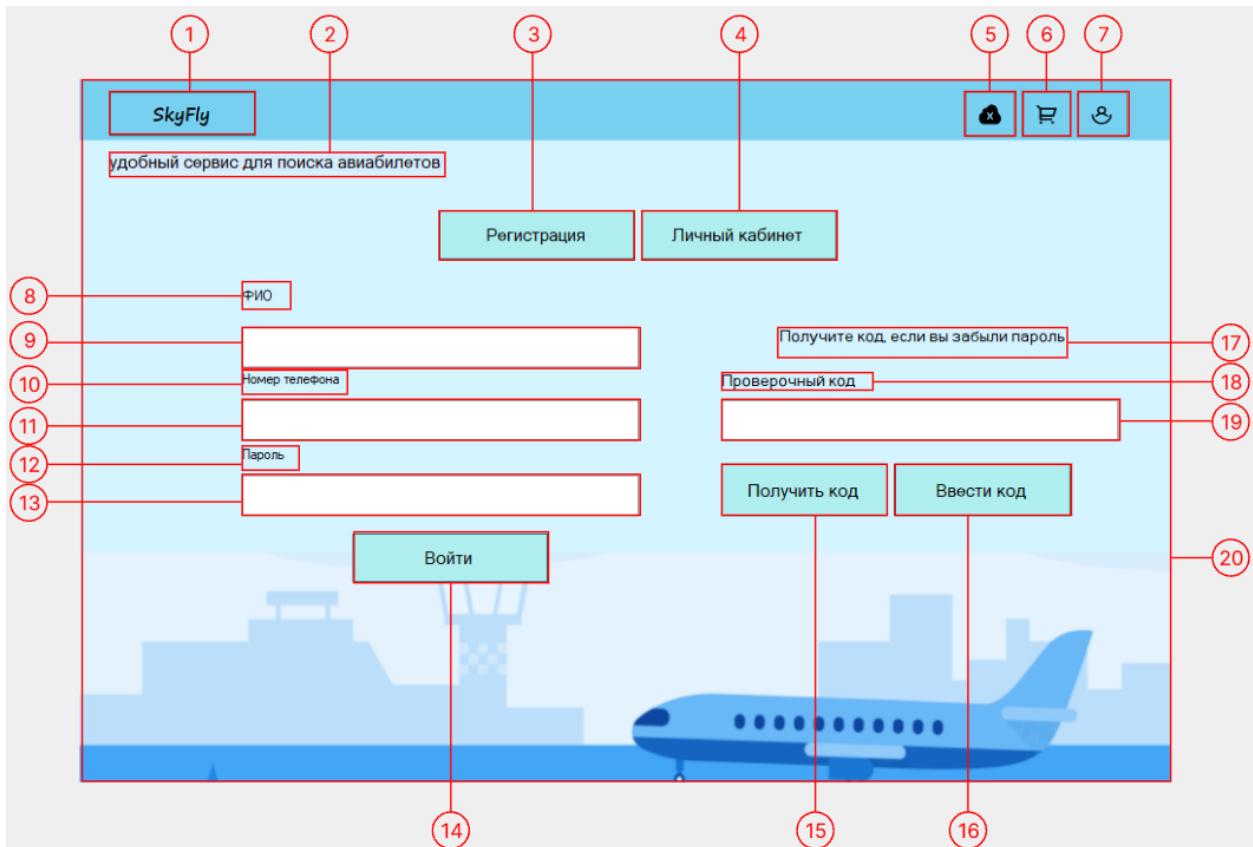


Рисунок 21 - Интерфейс взаимодействия пользователя со страницей для авторизации

В таблице 8 представлено описание объектов интерфейса взаимодействия пользователя со страницей для авторизации.

Таблица 8 - Описание объектов интерфейса взаимодействия пользователя со страницей для авторизации.

№	Тип объекта	Имя объекта	Действие/описание объекта
1	Button	button1	Кнопка «SkyFly», при нажатии которой открывается главная страница программы.
2	Label	label1	Поле, в котором хранится описание сервиса.

1	2	3	4
3	Button	button4	Кнопка, при нажатии которой программа открывает страницу для авторизации.
4	Button	button2	Кнопка, при нажатии которой программа открывает страницу личного кабинета пользователя.
5	Button	button8	Кнопка, при нажатии которой завершается работа программы.
6	Button	button10	Кнопка, при нажатии которой программа открывает страницу корзины покупок.
7	Button	button3	Кнопка, при нажатии которой программа открывает страницу для регистрации.
8	Label	label4	Название поля.
9	TextBox	textBox4	Поле, в которое пользователь вводит ФИО.
10	Label	label2	Название поля.
11	TextBox	textBox1	Поле, в которое пользователь вводит номер телефона.
12	Label	label3	Название поля.
13	TextBox	textBox2	Поле, в которое пользователь вводит пароль.
14	Button	button6	Кнопка, при нажатии которой программа проверяет наличие введенных данных и их корректность, если все верно, программа авторизует пользователя. Иначе отправляет уведомление «пользователь не найден»

1	2	3	4
15	Button	button9	Кнопка, при нажатии которой программа проверяет наличие введенных данных и их корректность, после чего отправляет проверочный код или выводит ошибку о неправильном введении данных.
16	Button	button7	Кнопка, при нажатии на которую программа проверяет введенный код с тем, который был отправлен пользователю и, в случае их совпадения, регистрирует пользователя. Иначе отправляет пользователю уведомление о неправильном коде.
17	Label	label6	Поле, в котором хранится подсказка на случай, если пользователь забыл пароль.
18	Label	label5	Название поля.
19	TextBox	textBox3	Поле, в которое пользователь вводит проверочный код.
20	PictureBox	pictureBox1	Элемент, который содержит изображение фона программы.

На рисунке 22 представлен интерфейс взаимодействия пользователя со страницей для оплаты билета. Числами на рисунке обозначены номера объектов интерфейса.

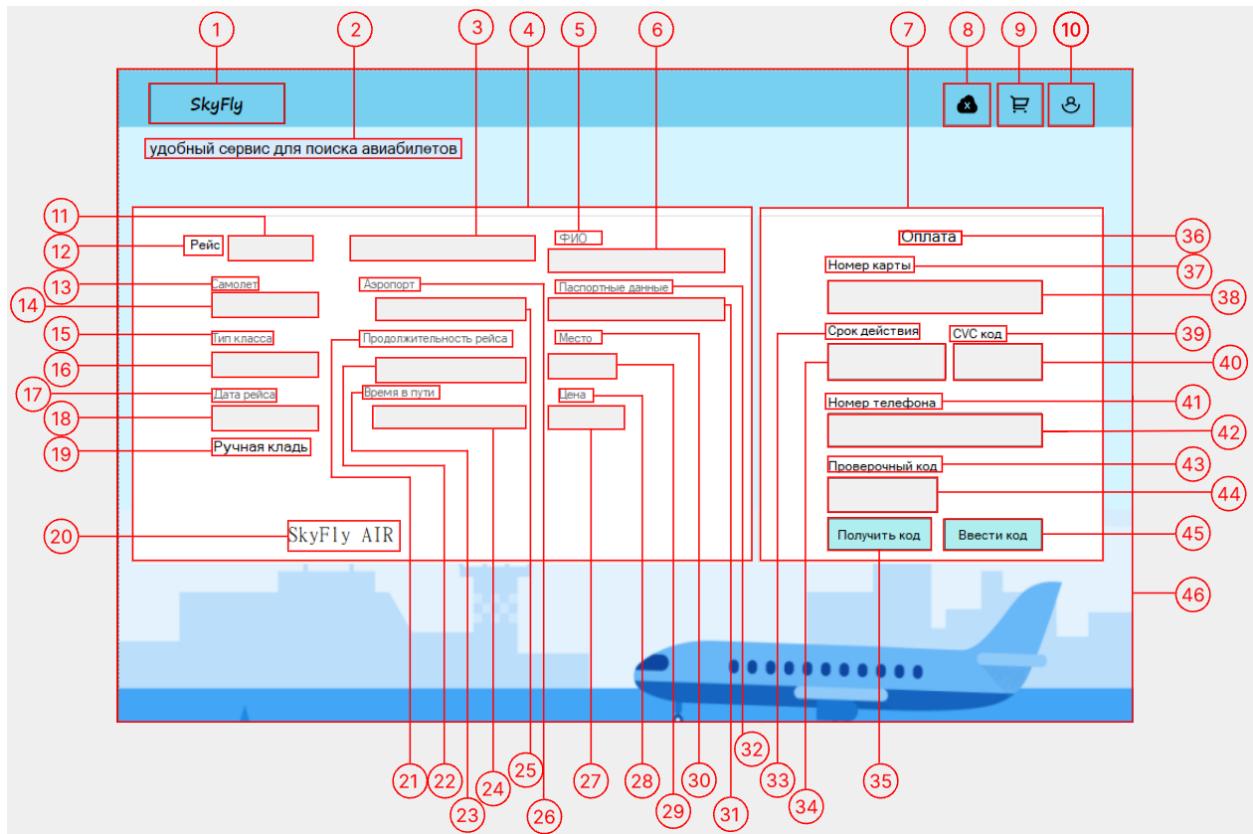


Рисунок 22 - Интерфейс взаимодействия пользователя со страницей для оплаты билета

В таблице 9 представлено описание объектов интерфейса взаимодействия пользователя со страницей для оплаты билета.

Таблица 9 - Описание объектов интерфейса взаимодействия пользователя со страницей для оплаты билета.

№	Тип объекта	Имя объекта	Действие/описание объекта
1	Button	button1	Кнопка «SkyFly», при нажатии которой открывается главная страница программы.
2	Label	label1	Поле, в котором хранится описание сервиса.

1	2	3	4
3	TextBox	textBox10	Поле, в которое программа вносит наименование рейса (город отправления – город прибытия).
4	GroupBox	groupBox1	Контейнер, содержащий в себе несколько полей, в которых содержится информация о рейсе.
5	Label	label3	Название поля.
6	TextBox	textBox17	Поле, в которое программа вносит ФИО.
7	GroupBox	groupBox2	Контейнер, содержащий в себе несколько полей для оплаты билета.
8	Button	button8	Кнопка, при нажатии которой завершается работа программы.
9	Button	button2	Кнопка, при нажатии которой программа открывает страницу корзины покупок.
10	Button	button3	Кнопка, при нажатии которой программа открывает страницу для регистрации.
11	TextBox	textBox14	Поле, в которое программа вносит номер рейса.
12	Label	label33	Название поля.
13	Label	label31	Название поля.
14	TextBox	textBox13	Поле, в которое программа вносит название самолёта, выполняющего рейс.
15	Label	label30	Название поля.
16	TextBox	textBox12	Поле, в которое программа вносит тип класса.
17	Label	label8	Название поля.
18	TextBox	textBox11	Поле, в которое программа вносит дату рейса.

1	2	3	4
19	Label	label29	Поле, которое содержит информацию о типе билета (разрешен ли провоз ручной клади).
20	Label	label5	Поле, в котором хранится название авиакомпании.
21	Label	label19	Название поля.
22	TextBox	textBox7	Поле, в которое программа вносит продолжительность рейса (время отправления - время прибытия).
23	Label	label35	Название поля.
24	TextBox	textBox8	Поле, в которое программа вносит значение времени в пути (рассчитывается программой исходя из времени отправления и времени прибытия).
25	TextBox	textBox6	Поле, в которое программа вносит названия аэропортов.
26	Label	label36	Название поля.
27	TextBox	textBox9	Поле, в которое программа вносит сгенерированную цену билета.
28	Label	label41	Название поля.
29	Label	label4	Поле, в которое программа вносит номер места в самолете.
30	Label	label2	Название поля.
31	TextBox	textBox18	Поле, в которое программа вносит паспортные данные пользователя.
32	Label	label6	Название поля.
33	Label	label11	Поле, в которое пользователь вводит срок действия карты.
34	TextBox	textBox2	Название поля.

1	2	3	4
35	Button	button9	Кнопка, при нажатии которой программа проверяет наличие введенных данных и их корректность, после чего отправляет проверочный код или выводит ошибку о неправильном введении данных.
36	Label	label9	Поле, в котором хранится название groupBox2.
37	Label	label10	Название поля.
38	TextBox	textBox1	Поле, в которое пользователь вводит номер карты.
39	Label	label12	Название поля.
40	TextBox	textBox3	Поле, в которое пользователь вводит CVC код карты.
41	Label	label13	Название поля.
42	TextBox	textBox4	Поле, в которое программа вносит номер телефона пользователя.
43	Label	label14	Название поля.
44	TextBox	textBox5	Поле, в которое пользователь вводит проверочный код.
45	Button	button7	Кнопка, при нажатии на которую программа проверяет введенный код с тем, который был отправлен пользователю и, в случае их совпадения, проводит оплату билета. Иначе отправляет пользователю уведомление о неправильном коде.
46	PictureBox	pictureBox1	Элемент, который содержит изображение фона программы.

На рисунке 23 представлен интерфейс взаимодействия пользователя со страницей личного кабинета пользователя. Числами на рисунке обозначены номера объектов интерфейса.

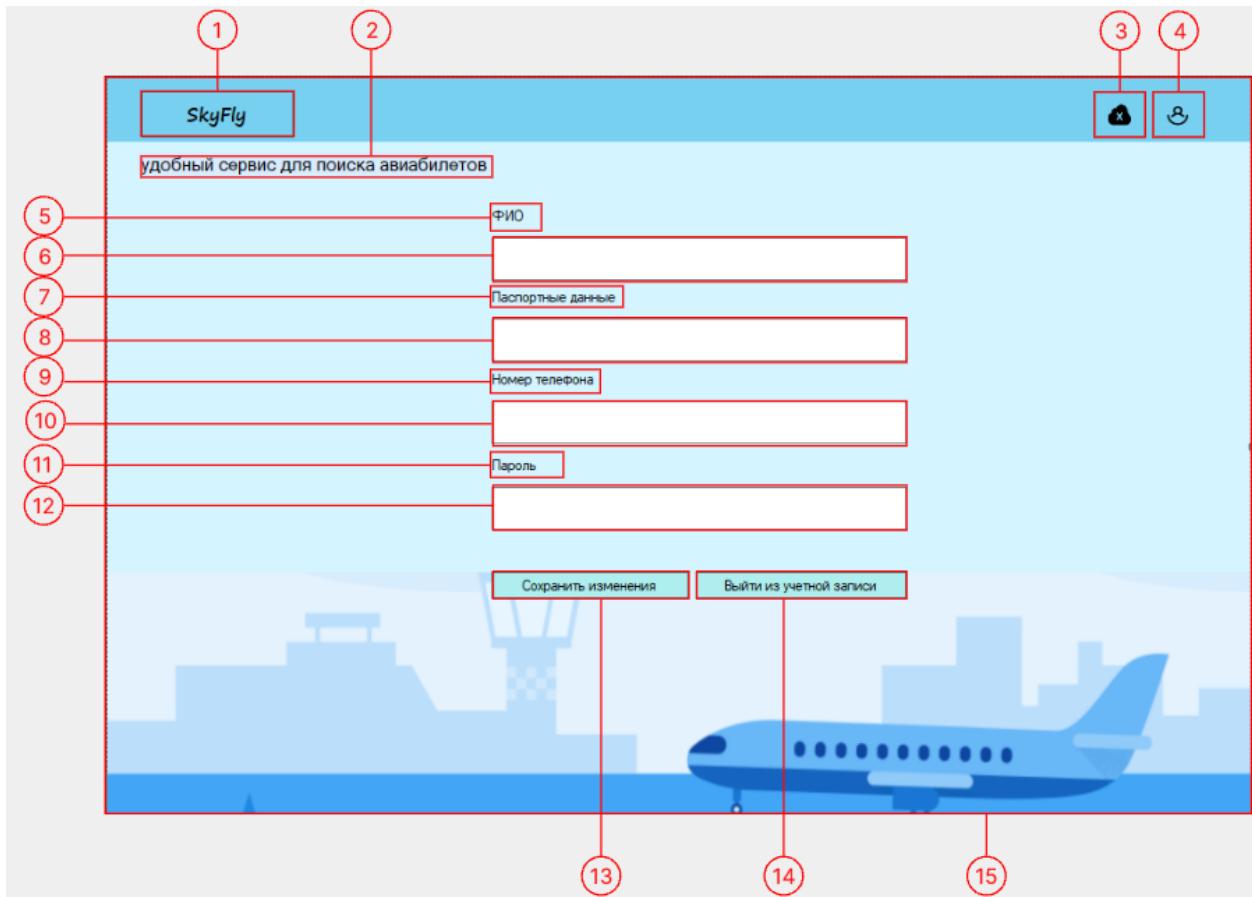


Рисунок 23 - Интерфейс взаимодействия пользователя со страницей личного кабинета пользователя

В таблице 10 представлено описание объектов интерфейса взаимодействия пользователя со страницей личного кабинета пользователя.

Таблица 10 - Описание объектов интерфейса взаимодействия пользователя со страницей личного кабинета пользователя.

№	Тип объекта	Имя объекта	Действие/описание объекта
1	Button	button1	Кнопка «SkyFly», при нажатии которой открывается главная страница программы.

1	2	3	4
2	Label	label1	Поле, в котором хранится описание сервиса.
3	Button	button8	Кнопка, при нажатии которой завершается работа программы.
4	Button	button3	Кнопка, при нажатии которой программа открывает страницу для регистрации.
5	Label	label4	Название поля.
6	TextBox	textBox4	Поле, в которое программа вносит ФИО пользователя (из файла).
7	Label	label5	Название поля.
8	TextBox	textBox1	Поле, в которое программа вносит паспортные данные пользователя (из файла).
9	Label	label2	Название поля.
10	TextBox	textBox2	Поле, в которое программа вносит номер телефона пользователя (из файла).
11	Label	label3	Название поля.
12	TextBox	textBox3	Поле, в которое программа вносит ФИО пользователя (из файла).
13	Button	button6	Кнопка, при нажатии которой программа сохраняет все изменения, внесенные пользователем и обновляет данные в файле.
14	Button	button2	Кнопка, при нажатии которой программа очищает все textBox.
15	PictureBox	pictureBox1	Элемент, который содержит изображение фона программы.

### **4.1.3 Описание классов**

#### **Описание класса Reys**

-Внутренние поля класса:

public string NomEr: хранит номер рейса в виде строки.

public string Otkuda: хранит название места отправления.

public string Kuda: хранит название места прибытия.

public string Klass: хранит названия типа класса в самолёте.

public DateTime DataOtpravleniya: хранит значение даты рейса.

public TimeSpan VremyaVyleta: хранит значение времени вылета.

public TimeSpan VremyaPrileta: хранит значение времени прилета.

public TimeSpan VremyaVPute: хранит значение времени в пути.

public string ModelSamolyota: хранит название модели самолёта.

public int Tsena: хранит значение цены рейса.

public string AeroportOtkuda: хранит название аэропорта отправления.

public string AeroportKuda: хранит название аэропорта прибытия.

**-Внутренние методы класса:**

GenerateNomEr(): Инициализирует новый объект Random. Генерирует две случайные буквы в диапазоне от 'A' до 'Z' с помощью и присваивает их переменным letter1 и letter2. Генерирует случайное число от 1000 до 9999 и присваивает его переменной number. Формирует строку с номером рейса в виде №letter1letter2number и возвращает ее.

Reys(string filePath): Считывает данные из файла. Разделяет каждую строку из файла по символу ':' и извлекает данные из второй части строки (после двоеточия). Присваивает извлеченные данные свойствам объекта Reys. Генерирует случайный номер рейса с помощью метода GenerateNomEr(). Генерирует случайные времена вылета, прилета и время в пути. Устанавливает модель самолета как "AIRBUS777". Генерирует случайную цену в диапазоне от 5000 до 10000. Определяет аэропорты

отправления и прибытия с помощью метода GetAeroport, основываясь на городах отправления и прибытия.

GetAeroport(string city): Проверяет значения города и возвращает название соответствующего аэропорта.

### **Описание класса Ticket**

#### **-Внутренние поля класса:**

public string NomEr: хранит номер рейса в виде строки.

public string Marshrut: хранит название рейса (например, "Москва - Санкт-Петербург") в виде строки.

public DateTime DepartureDate: хранит дату отправления рейса в виде объекта типа DateTime.

public string ModelSamolyota: хранит модель самолёта рейса в виде строки.

public string Klass: хранит класс рейса (например, "Эконом", "Бизнес") в виде строки.

public string Airport: хранит название аэропорта в виде строки.

public string Duration: хранит продолжительность рейса в виде строки.

public string TravelTime: хранит время в пути рейса в виде строки.

public decimal Tsena: хранит цену билета на рейс в виде десятичного числа.

public string Seat: хранит номер места пассажира в виде строки.

public string FullName: хранит полное имя пассажира в виде строки.

public string PhoneNumber: хранит номер телефона пассажира в виде строки.

public string PassportData: хранит паспортные данные пассажира в виде строки.

public string NumberCard: хранит номер пластиковой карты пассажира в виде строки.

public string SrokCard: хранит срок действия пластиковой карты пассажира в виде строки.

public string CVC: хранит CVC-код пластиковой карты пассажира в виде строки.

public string Code: хранит код подтверждения в виде строки.

#### **-Внутренние методы класса:**

ReadFromFile(string filePath): Читает данные о рейсе из файла. Перебирает каждую строку и проверяет, начинается ли она с определенного ключевого слова (например, "Номер рейса:"). Если строка начинается с ключевого слова, метод извлекает данные из строки и записывает их в соответствующие свойства объекта Ticket.

LoadUserData(string path): Читает данные пользователя из файла. Затем он разделяет каждую строку по символу ':' и извлекает данные из второй части строки (после двоеточия). Извлеченные данные записываются в свойства FullName, PhoneNumber, PassportData объекта Ticket.

ValidateInput(TextBox textBox1, TextBox textBox2, TextBox textBox3): Проверяет введенные данные карты пользователя на корректность. Выполняет проверки формата номера карты, срока действия и CVC-кода. Если формат данных неверен, метод выводит сообщение об ошибке и возвращает false.

## **Описание класса UserRegistration**

#### **-Внутренние поля класса:**

public string FullName: хранит полное имя пользователя в виде строки.

public string PassportData: хранит данные паспорта пользователя в виде строки.

`public string PhoneNumber:` хранит номер телефона пользователя в виде строки.

`public string Password:` хранит пароль пользователя в виде строки.

`public string Code:` хранит код подтверждения (если используется) в виде строки.

**-Внутренние методы класса:**

`UserRegistration(string fullName, string passportData, string phoneNumber, string password, string code):` Принимает в качестве параметров полное имя, паспортные данные, номер телефона, пароль и код подтверждения. Присваивает эти параметры соответствующим свойствам объекта `UserRegistration`.

`ValidateInput (TextBox textBox1, TextBox textBox2, TextBox textBox3, TextBox textBox4, TextBox textBox5):` Принимает в качестве параметров пять элементов управления типа `TextBox`, которые соответствуют введенным пользователем данным. Проверяет каждое поле на соответствие регулярным выражениям. Если хотя бы одно поле не соответствует требованиям, выводит сообщение об ошибке и возвращает `false`. Если все поля прошли проверку, то сохраняет данные из `TextBox` в соответствующие свойства объекта `UserRegistration` и возвращает `true`.

`SaveToFile(string fileName):` Принимает в качестве параметра имя файла. Формирует полный путь к файлу. Создает объект для записи в файл. Записывает в файл свойства объекта в заданном формате. Закрывает запись в файл.

## **Описание класса `CodeGenerator`**

**-Внутренние поля класса:**

`private Random random = new Random();` Поле класса, которое хранит объект типа Random. Этот объект используется для генерации случайных чисел.

**-Внутренние методы класса:**

`public string GenerateCode();` Генерирует случайный код в виде строки. Использует метод для генерации случайного целого числа в диапазоне от 100 до 999 (не включая 1000). Преобразует сгенерированное число в строку. Возвращает результат в виде строки.

На рисунке 24 показана диаграмма программных классов.

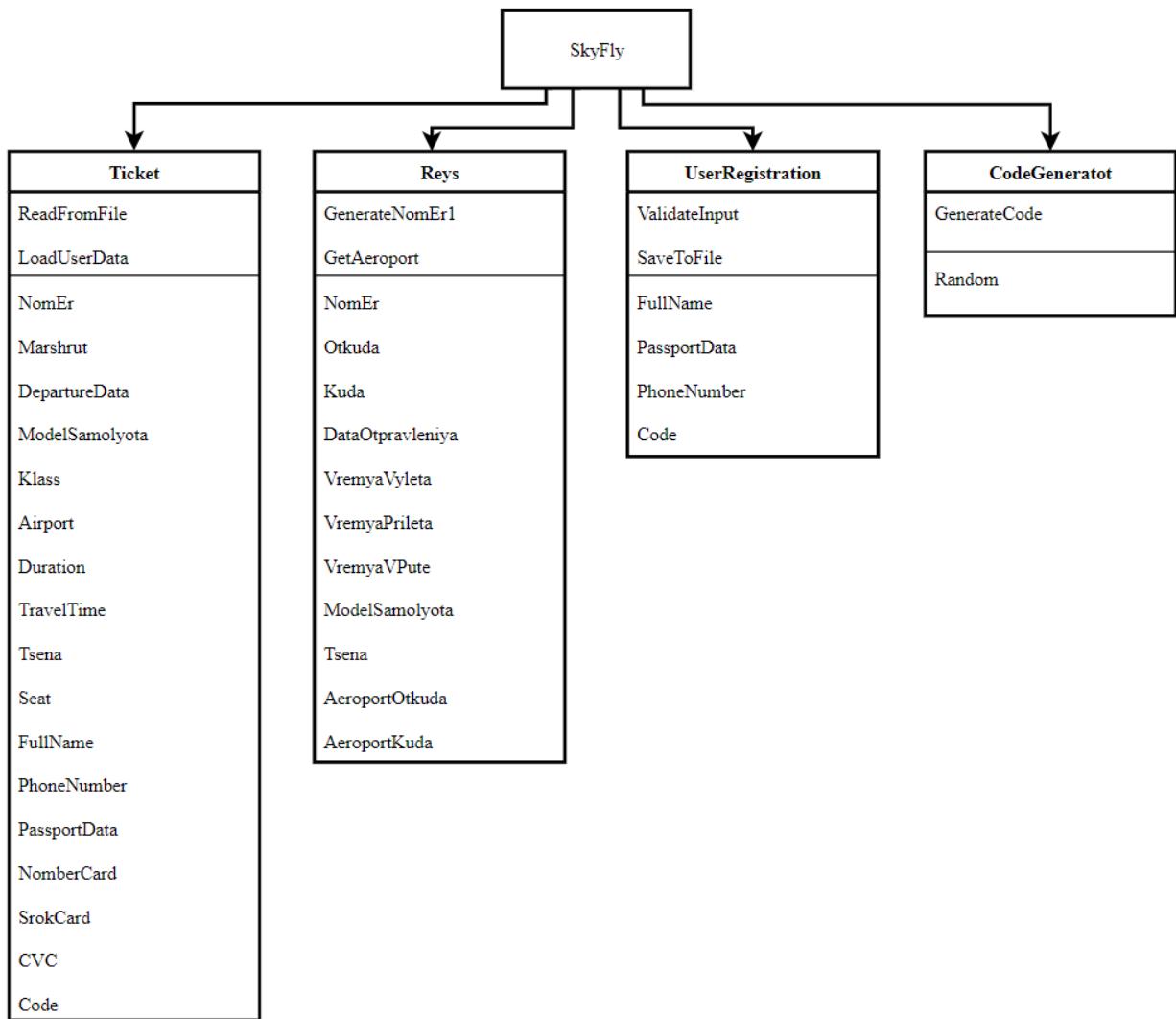


Рисунок 24 – Диаграмма программных классов

## 4.2 Тестирование программной системы

Разработка тестовых наборов для интерфейсов будет рассмотрена на примере тестирования программы для моделирования сервиса для продажи авиабилетов.

В таблице 11 представлены тестовые наборы для отладки интерфейса.

Таблица 11 – Тестовые наборы для отладки интерфейса

Имя теста	Проверяемая ситуация	Действия пользователя	Входные данные	Реакция системы
1	2	3	4	5
T1	Пользователь открыл программу и ничего не нажимает.	Пользователь запустил программу и ничего не нажимает.	Открыто стартовая страница программы.	Открывается стартовая страница программы, с логотипом.
T2	Пользователь переходит к главной странице программы.	Пользователь нажимает на кнопку “Далее”.	На стартовой странице программы нажата кнопка “Далее”.	Открывается главная страница программы.
T3	Пользователь вводит параметры рейса.	Пользователь на главной странице программы вводит параметры рейса.	На главной странице программы нажато label.	Программа записывает данные рейса.
T4	Пользователь выбирает рейсы по заданным параметрам.	Пользователь на главной странице программы нажимает кнопку “Найти рейсы”.	На главной странице программы нажата кнопка “Найти рейсы”.	Программа сохраняет данные в файл и открывает страницу подобранных рейсов.

1	2	3	4	5
T5	Пользователь выбирает рейсы.	Пользователь на странице подобранных рейсов нажимает на кнопку “Загрузить рейсы”.	Нажата кнопка «Загрузить рейсы».	Программа извлекает данные из файла и вносит их в определенные поля.
T6	Пользователь добавляет рейс в корзину покупок.	Пользователь нажимает на кнопку “Добавить”.	Нажата кнопка “Добавить”.	Программа сохраняет данные выбранного рейса в файл и вносит данные в текстовые поля на странице корзины покупок.
T7	Пользователь открывает корзину покупок.	Пользователь нажимает на кнопку с изображением корзины покупок (можно нажать на любом этапе программы).	Нажата кнопка с изображением корзины покупок.	Программа открывает страницу корзины покупок.

1	2	3	4	5
T8	Пользователь добавляет в стоимость билета провоз багажа.	Пользователь нажимает на поле CheckBox “Добавить багаж”.	Нажато поле CheckBox “Добавить багаж”.	Программа добавляет в стоимость билета провоз багажа.
T9	Пользователь удаляет выбранный рейс из корзины покупок.	Пользователь нажимает на кнопку “Удалить”.	Нажата кнопка “Удалить”.	Программа очищает все текстовые поля.
T10	Пользователь выбирает другое место в самолете.	Пользователь выбирает место в самолете и нажимает на него.	Пользователь выбрал место.	Программа проверяет совпадает ли выбранное место с введенным пользователем типом класса. Если типы классов совпадают, программа сохраняет изменения и открывает страницу корзины покупок.

1	2	3	4	5
T11	Пользователь переходит к оплате билета.	Пользователь нажимает на кнопку “Перейти к оплате”.	Нажата кнопка “Перейти к оплате”.	Программа проверяет авторизован ли пользователь. Если авторизован, программа открывает страницу для оплаты билета, иначе открывает уведомление о необходимости авторизации.
T12	Пользователь открывает страницу регистрации в сервисе.	Пользователь нажимает на кнопку с изображением личного кабинета пользователя (можно нажать на любом этапе программы).	Нажата кнопка с изображением личного кабинета пользователя.	Программа открывает страницу для регистрации.

1	2	3	4	5
T13	Пользователь вводит данные для регистрации в сервисе.	Пользователь вводит необходимые данные для регистрации и получает проверочный код.	Введены необходимые данные. Нажата кнопка “Получить код”.	Программа проверяет введенные пользователем данные, если данные введены корректно, программа открывает уведомление с проверочным кодом и сохраняет данные в файл.
T14	Пользователь вводит проверочный код для регистрации.	Пользователь ввел проверочный код и нажимает на кнопку “Ввести код”.	Введены необходимые данные. Нажата кнопка “ Ввести код”.	Программа проверяет, совпадает ли введенный код с тем, что был в уведомлении, если да, то регистрирует пользователя. Иначе открывает уведомление о неверном коде.

1	2	3	4	5
T15	Пользователь открывает страницу для авторизации.	Пользователь нажимает кнопку “Авторизация”.	Нажата кнопка “Авторизация”.	Программа открывает страницу для авторизации.
T16	Пользователь авторизуется в сервисе.	Пользователь вводит персональные данные и нажимает на кнопку “Войти”.	Введены необходимые данные. Нажата кнопка “Войти”.	Программа проверяет введенные данные пользователя. Если данные введены корректно и такой пользователь существует (есть данные в файле), программа авторизует пользователя в сервисе.

1	2	3	4	5
T17	Пользователь забыл пароль от личного кабинета.	Пользователь ввел персональные данные и нажимает на кнопку “Получить код”.	Введены необходимые данные. Нажата кнопка “Получить код”.	Программа проверяет введенные пользователем данные, если данные введены корректно, программа открывает уведомление с проверочным кодом
T18	Пользователь получает код для авторизации.	Пользователь ввел проверочный код и нажимает на кнопку “ Ввести код ”;	Введен проверочный код. Нажата кнопка “Ввести код ”.	Программа проверяет, совпадает ли введенный код с тем, что был в уведомлении, если да, то авторизует пользователя. Иначе открывает уведомление о неверном коде.

1	2	3	4	5
T19	Пользователь меняет данные в личном кабинете.	Пользователь меняет персональные данные и нажимает на кнопку “Сохранить изменения”.	Персональные данные изменены. Нажата кнопка “Сохранить изменения”.	Программа проверяет корректно ли внесены изменения. Если данные корректны, программа записывает новые данные пользователя в файл. Иначе выводит уведомление о неправильно введенных данных.
T20	Пользователь выходит из учетной записи.	Пользователь нажимает на кнопку “Выйти из учетной записи”.	Нажата кнопка “Выйти из учетной записи”.	Программа очищает все текстовые поля.

1	2	3	4	5
T21	Пользователь получает проверочный код для оплаты билета.	Пользователь вводит все необходимые данные и нажимает кнопку “Получить код”;	Введены необходимые данные. Нажата кнопка “Получить код”.	Программа проверяет введенные пользователем данные, если данные введены корректно, программа открывает уведомление с проверочным кодом.
T22	Пользователь оплачивает билет.	Пользователь вводит проверочный код и нажимает кнопку “Ввести код”;	Введен проверочный код. Нажата кнопка “Ввести код”.	Программа проверяет, совпадает ли введенный код с тем, что был в уведомлении, если да, то проводит оплату билета пользователя. Иначе открывает уведомление о неверном коде.

1	2	3	4	5
T23	Пользователь закрывает приложение и завершает работу программы.	Пользователь нажимает на кнопку с изображением облака с крестиком.	Нажата кнопка с изображением облака с крестиком.	После нажатия на кнопку с изображением облака с крестиком программа завершает работу (можно нажать на любом этапе программы).

## **Тестовые наборы для отладки работы программы.**

Для отладки работы программы разработаны следующие тестовые наборы:

### **1. Случай использования: пользователь открыл программу и ничего не нажимает.**

Предусловие: программа запущена, открыта стартовая страница программы;

Тестовый случай: пользователь запустил программу и ничего не нажимает;

Ожидаемый результат: Программа открывает стартовую страницу программы с логотипом сервиса.

Результат представлен на рисунке 25.



Рисунок 25 - Результат запуска программы

## **2. Случай использования: пользователь переходит к главной странице программы.**

Предусловие: программа запущена, открыта стартовая страница программы;

Тестовый случай: пользователь на стартовой странице программы нажимает на кнопку “Далее”;

Ожидаемый результат: программа открывает главную страницу.

## **3. Случай использования: Пользователь вводит параметры рейса.**

Предусловие: программа запущена, открыта главная страница программы;

Тестовый случай: На главной странице программы пользователь вводит параметры рейса. Пользователь нажал на текстовое поле Label, открыто окно выбора типа класса;

Ожидаемый результат: программа записывает введенные пользователем параметры в текстовые поля.

Результат представлен на рисунке 26.

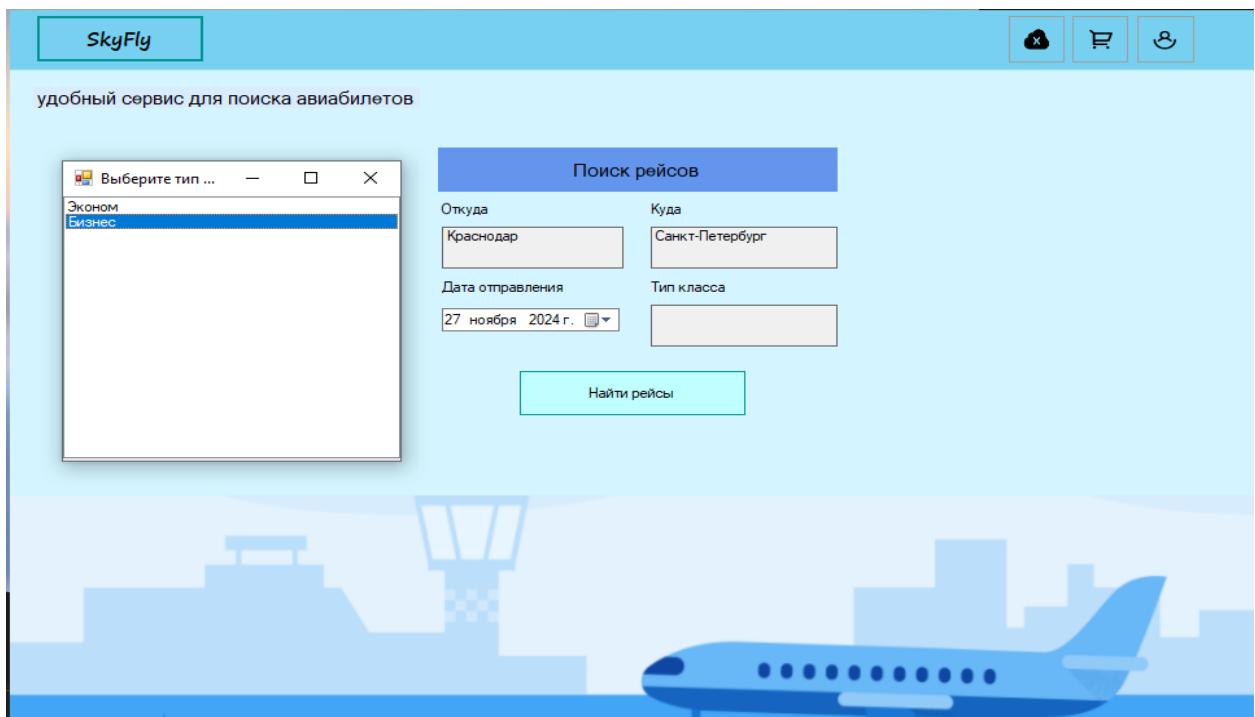


Рисунок 26 - Результат нажатия на текстовое поле

**4. Случай использования: пользователь ищет рейсы по введенным параметрам.**

Предусловие: программа запущена, открыта главная страница программы, введены все параметры рейса;

Тестовый случай: На главной странице программы пользователь нажимает на кнопку “Найти рейсы”;

Ожидаемый результат: Программа сохраняет введенные параметры в файл, открывает страницу с подобранными рейсами.

**5. Случай использования: пользователь выбирает рейсы.**

Предусловие: программа запущена, открыта страница подобранных рейсов программы, введены все параметры рейса;

Тестовый случай: На странице подобранных рейсов пользователь нажимает на кнопку “Загрузить рейсы”;

Ожидаемый результат: После нажатия на кнопку “Загрузить рейсы” программа извлекает данные из файла и вносит их в определенные поля.

Результат представлен на рисунке 27-28.

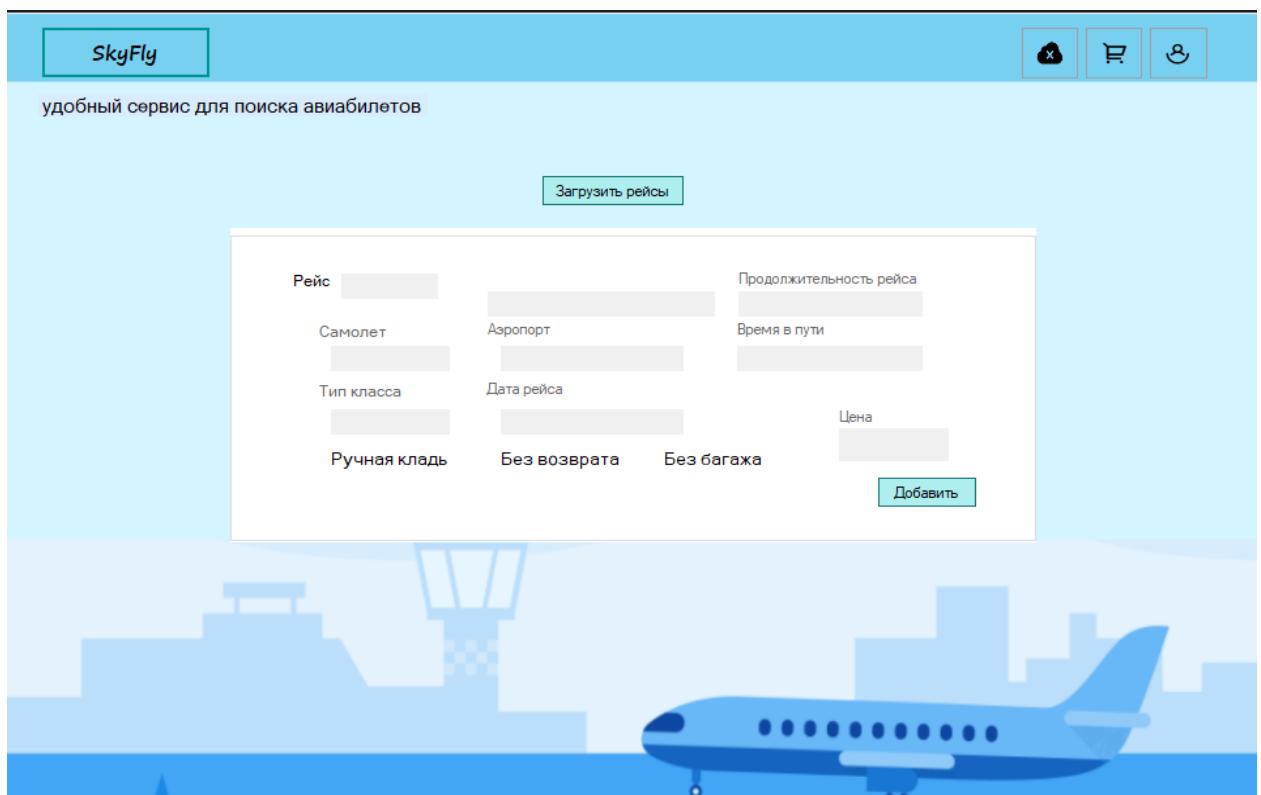


Рисунок 27 - Результат нажатия на кнопку “Найти рейсы”

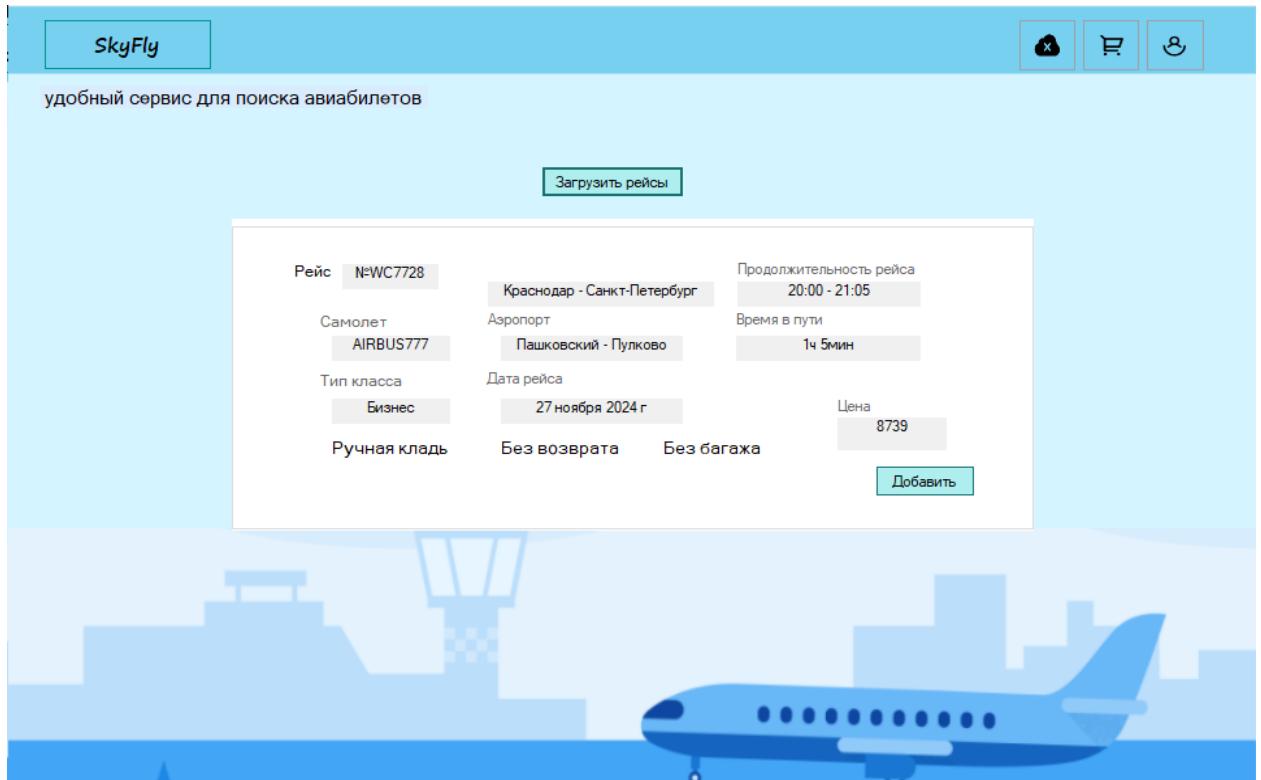


Рисунок 28 - Результат нажатия на кнопку “Загрузить рейсы”

**6. Случай использования: Пользователь добавляет рейс в корзину покупок.**

Предусловие: программа запущена, открыта страница подобранных рейсов;

Тестовый случай: Пользователь нажимает на кнопку “Добавить”;

Ожидаемый результат: Программа сохраняет данные выбранного рейса в файл и вносит данные в текстовые поля на странице корзины покупок.

**7. Случай использования: Пользователь открывает корзину покупок.**

Предусловие: программа запущена, открыта страница подобранных рейсов;

Тестовый случай: Пользователь нажимает на кнопку с изображением корзины покупок (можно нажать на любом этапе программы);

Ожидаемый результат: Программа открывает страницу корзины покупок.

Результат представлен на рисунке 29.

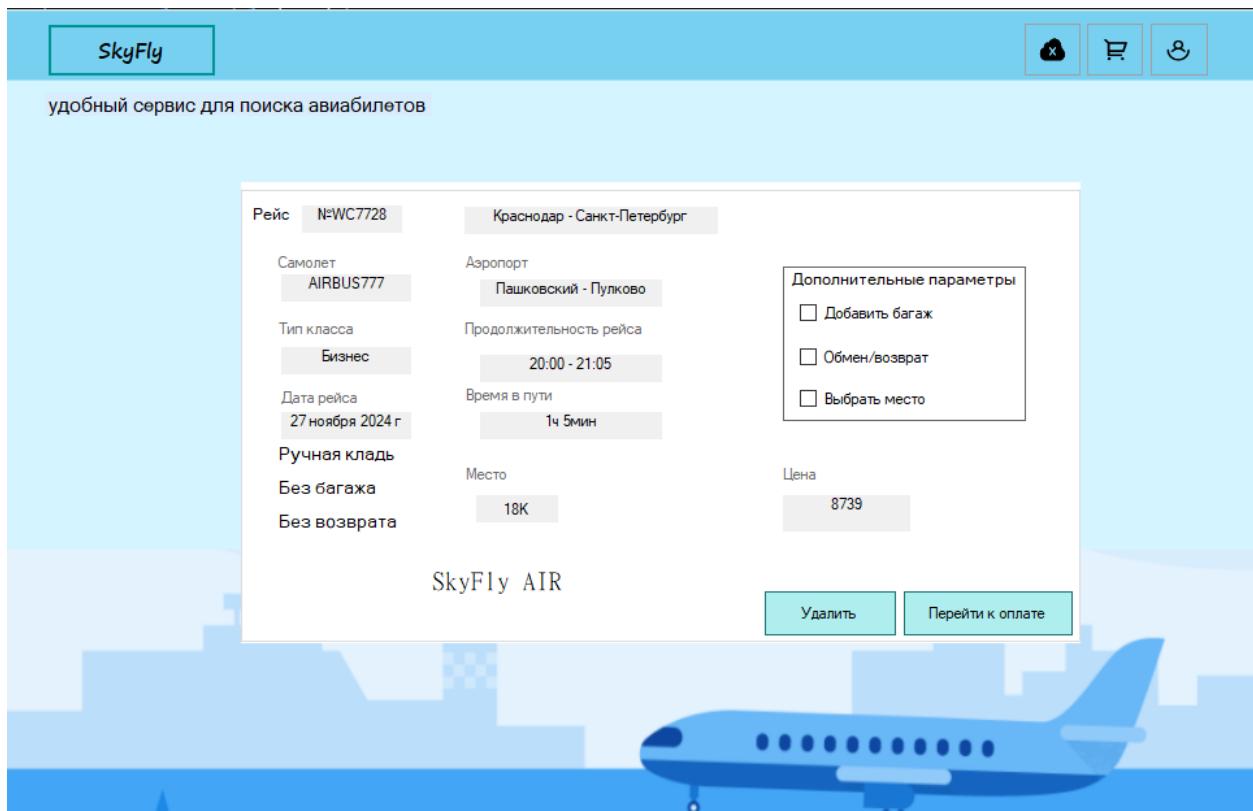


Рисунок 29 - Результат нажатия на кнопку с графическим изображением корзины покупок

## 8. Случай использования: Пользователь добавляет в стоимость билета провоз багажа.

Предусловие: программа запущена, открыта страница корзины покупок;

Тестовый случай: Пользователь нажимает на CheckBox “Добавить багаж”;

Ожидаемый результат: Программа обновляет состояние CheckBox и обновляет цену (аналогично с другими CheckBox) .

Результат представлен на рисунке 30.

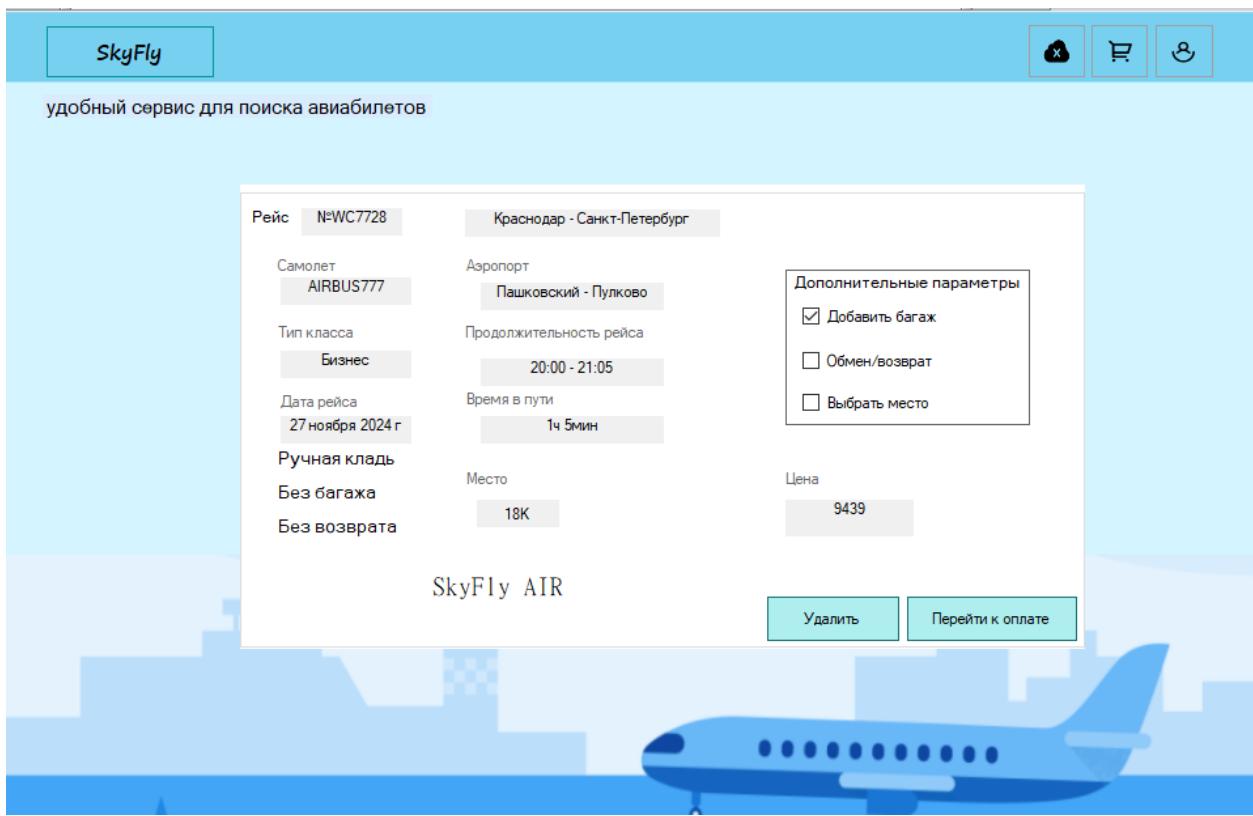


Рисунок 30 - Результат нажатия на CheckBox “Добавить багаж”

## 9. Случай использования: Пользователь удаляет выбранный рейс из корзины покупок.

Предусловие: программа запущена, открыта страница корзины покупок;

Тестовый случай: Пользователь нажимает кнопку “Удалить”;

Ожидаемый результат: Программа очищает все текстовые поля.

Результат представлен на рисунке 31.

## 10. Случай использования: Пользователь выбирает другое место в самолёте.

Предусловие: программа запущена, открыта страница корзины покупок;

Тестовый случай: Пользователь выбирает место в самолете и нажимает на него;

Ожидаемый результат: При нажатии на место в самолёте программа проверяет, совпадает ли выбранное место с введенным пользователем типом

класса. Если типы классов совпадают, программа сохраняет изменения и открывает страницу корзины покупок.

Результат представлен на рисунке 32-34.

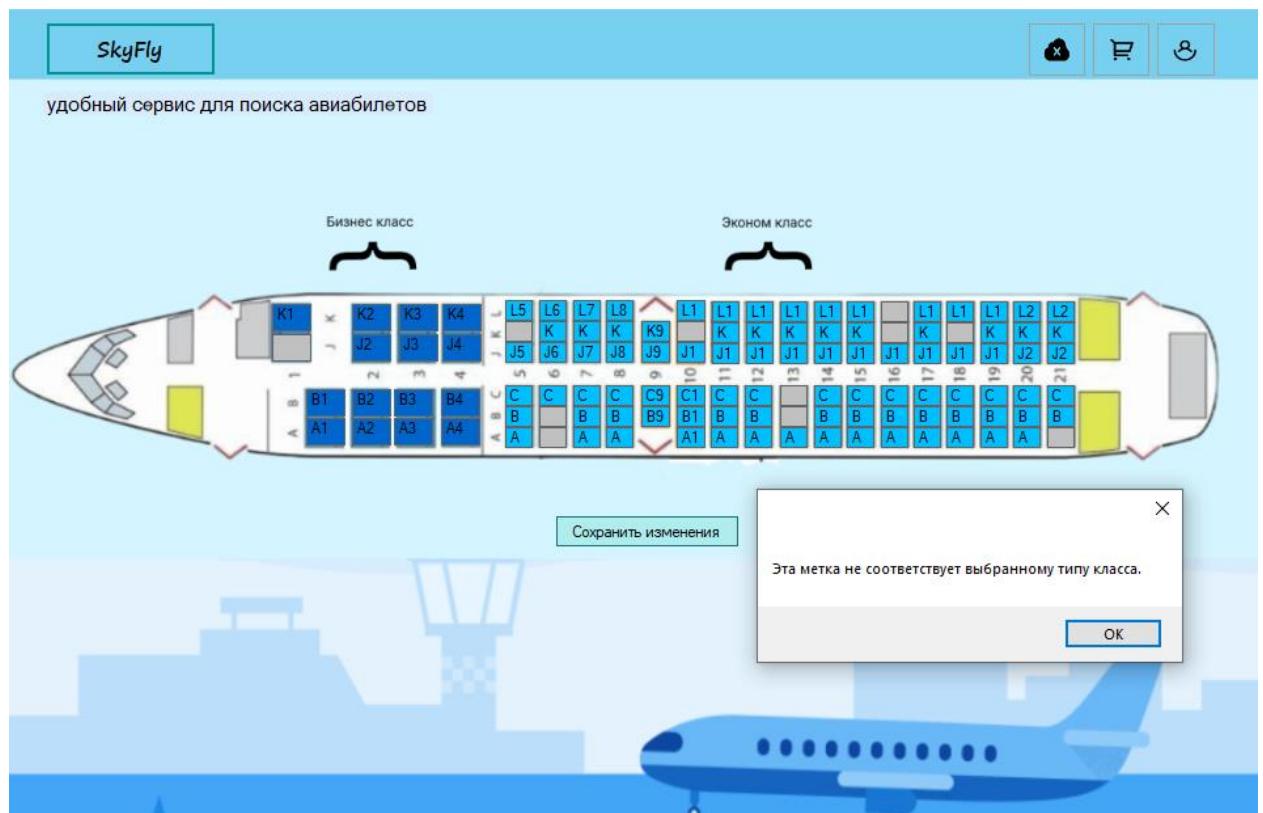


Рисунок 32 - Результат выбора места, несоответствующего введенному типу класса.

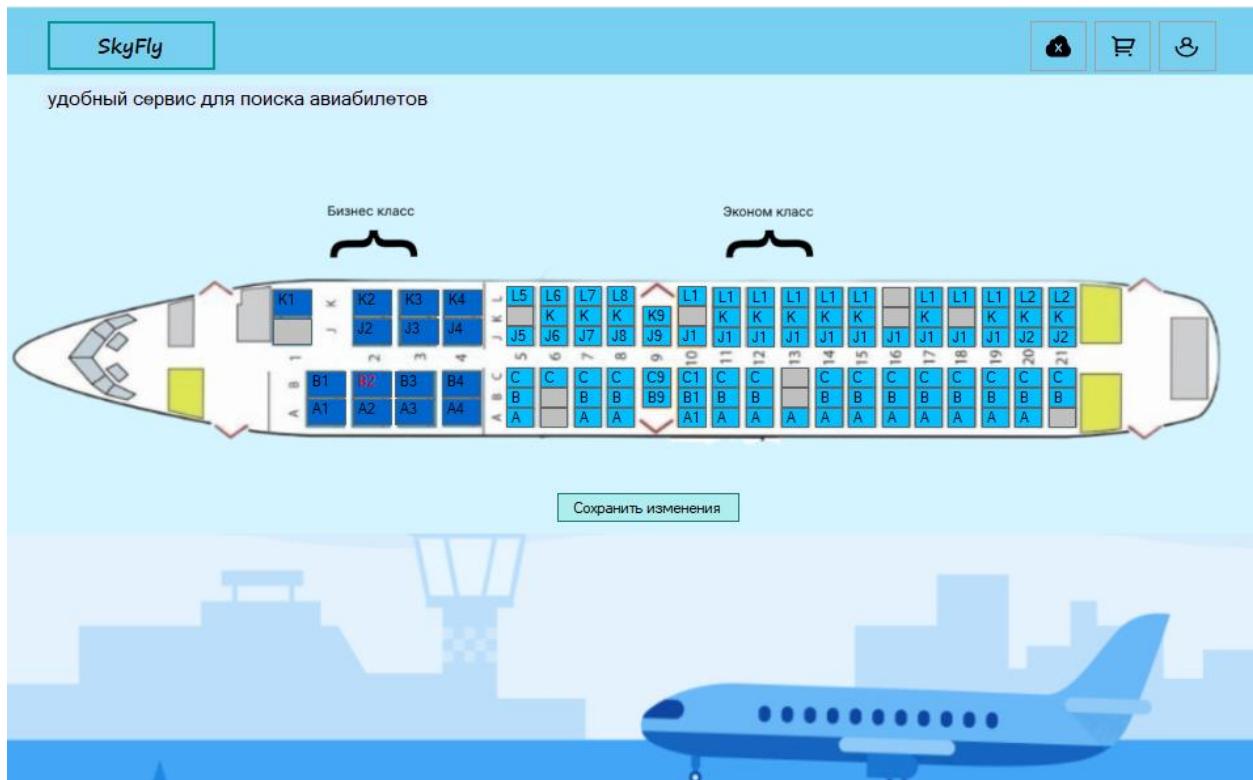


Рисунок 33 - Результат выбора места, соответствующего введенному типу класса

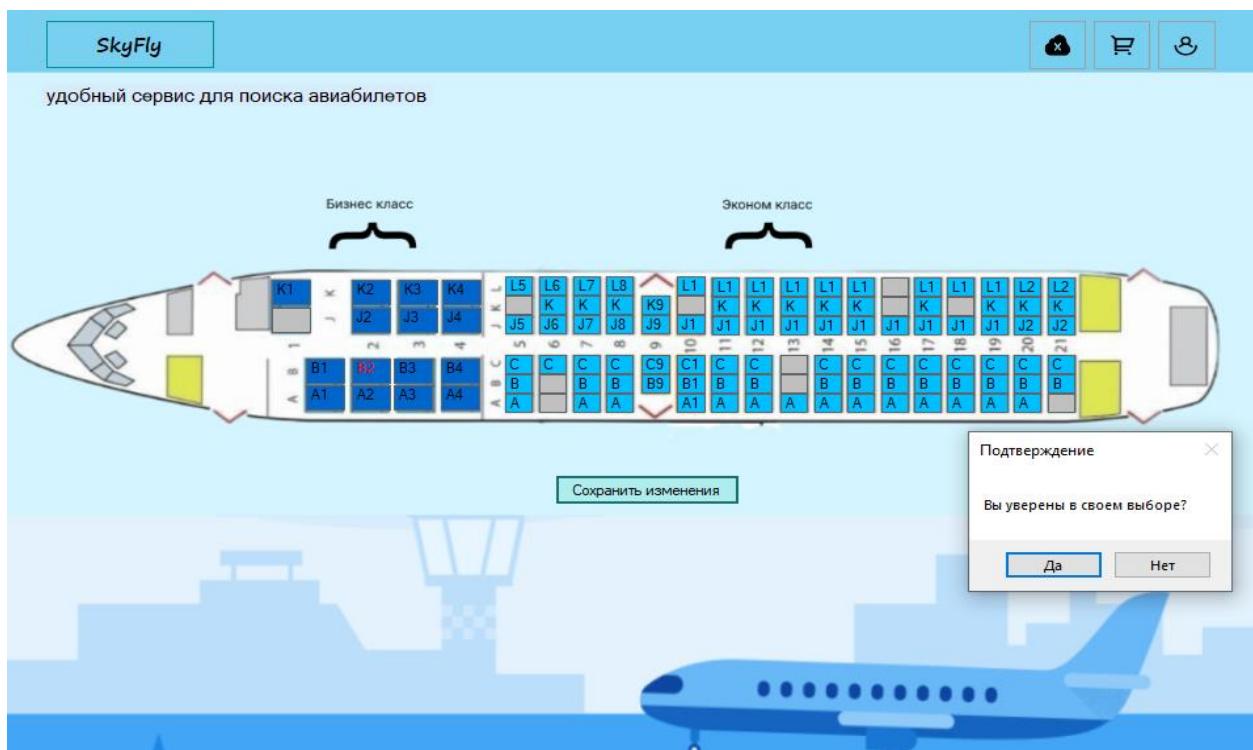


Рисунок 34 - Результат нажатия на кнопку “Сохранить изменения”

## 11. Случай использования: Пользователь переходит к оплате билета.

Предусловие: программа запущена, открыта страница корзины покупок;

Тестовый случай: Пользователь нажимает кнопку “Перейти к оплате”;

Ожидаемый результат: Программа проверяет, авторизован ли пользователь. Если авторизован, программа открывает страницу для оплаты билета, иначе открывает уведомление о необходимости авторизации.

Результат представлен на рисунке 35-36.

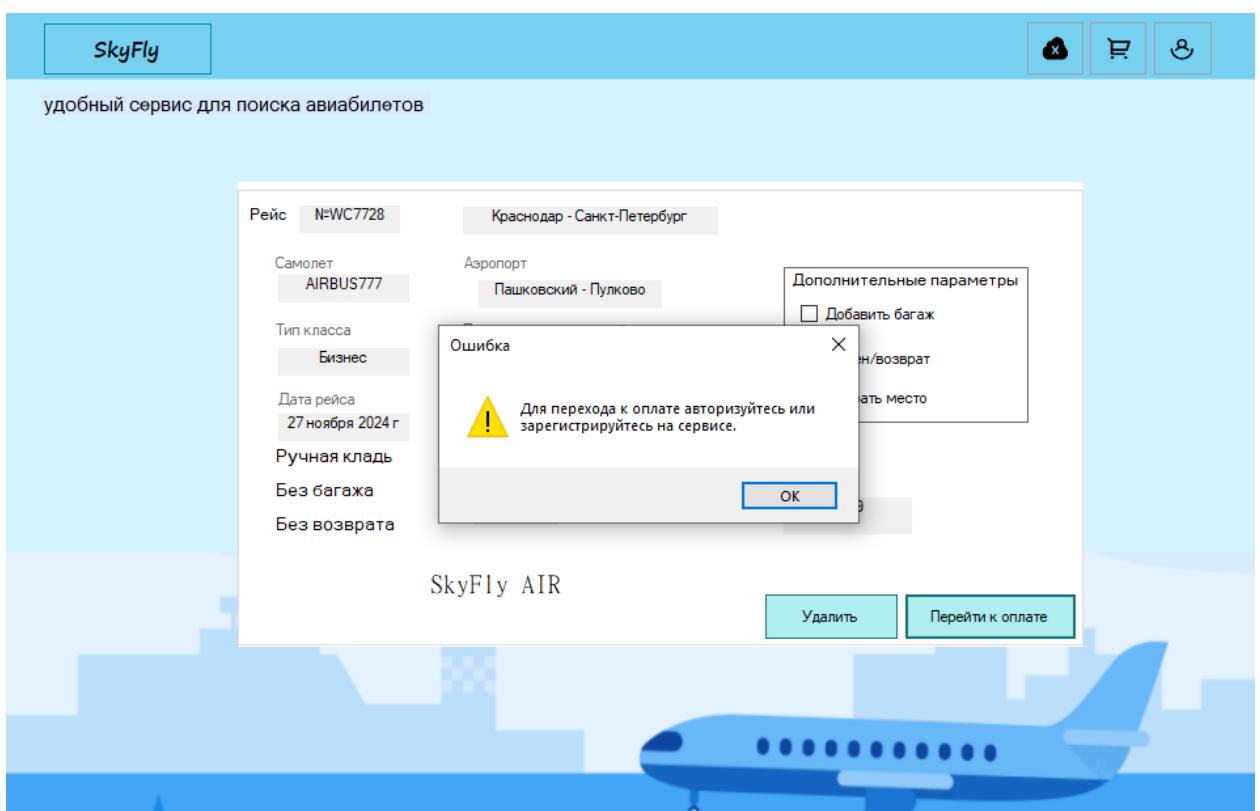


Рисунок 35 - Результат нажатия на кнопку “Перейти к оплате”

## **12. Случай использования: Пользователь открывает страницу регистрации в сервисе.**

Предусловие: программа запущена, открыта страница корзины покупок;

Тестовый случай: Пользователь нажимает на кнопку с изображением личного кабинета пользователя (можно нажать на любом этапе программы). После пользователь вводит необходимые данные, получает и вводит проверочный код;

Ожидаемый результат: Программа открывает страницу для регистрации.

Результат представлен на рисунке 37.

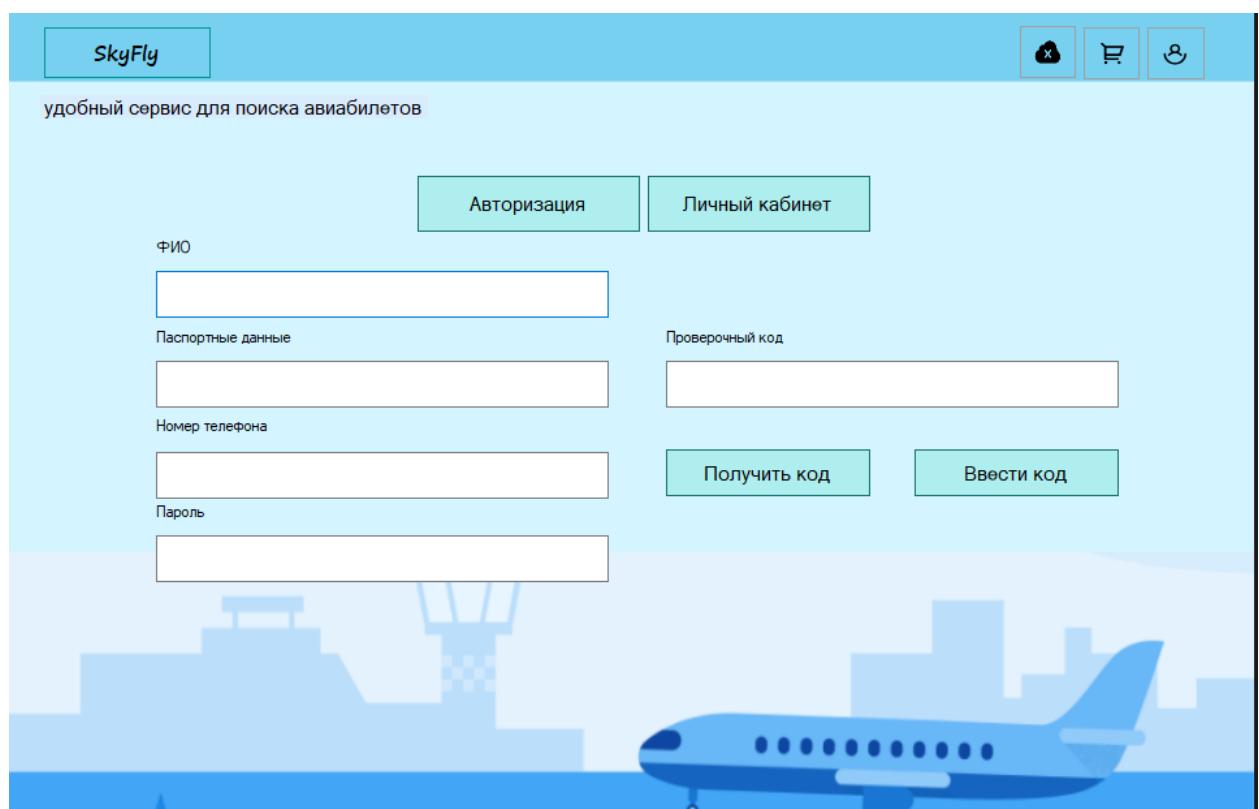


Рисунок 35 - Результат открытия страницы для регистрации

## **13. Случай использования: Пользователь проходит регистрацию в сервисе.**

Предусловие: программа запущена, открыта страница регистрации;

Тестовый случай: Пользователь вводит необходимые данные для регистрации и получает проверочный код;

Ожидаемый результат: Программа проверяет введенные пользователем данные, если данные введены корректно, программа открывает уведомление с проверочным кодом и сохраняет данные в файл.

Результат представлен на рисунке 38.

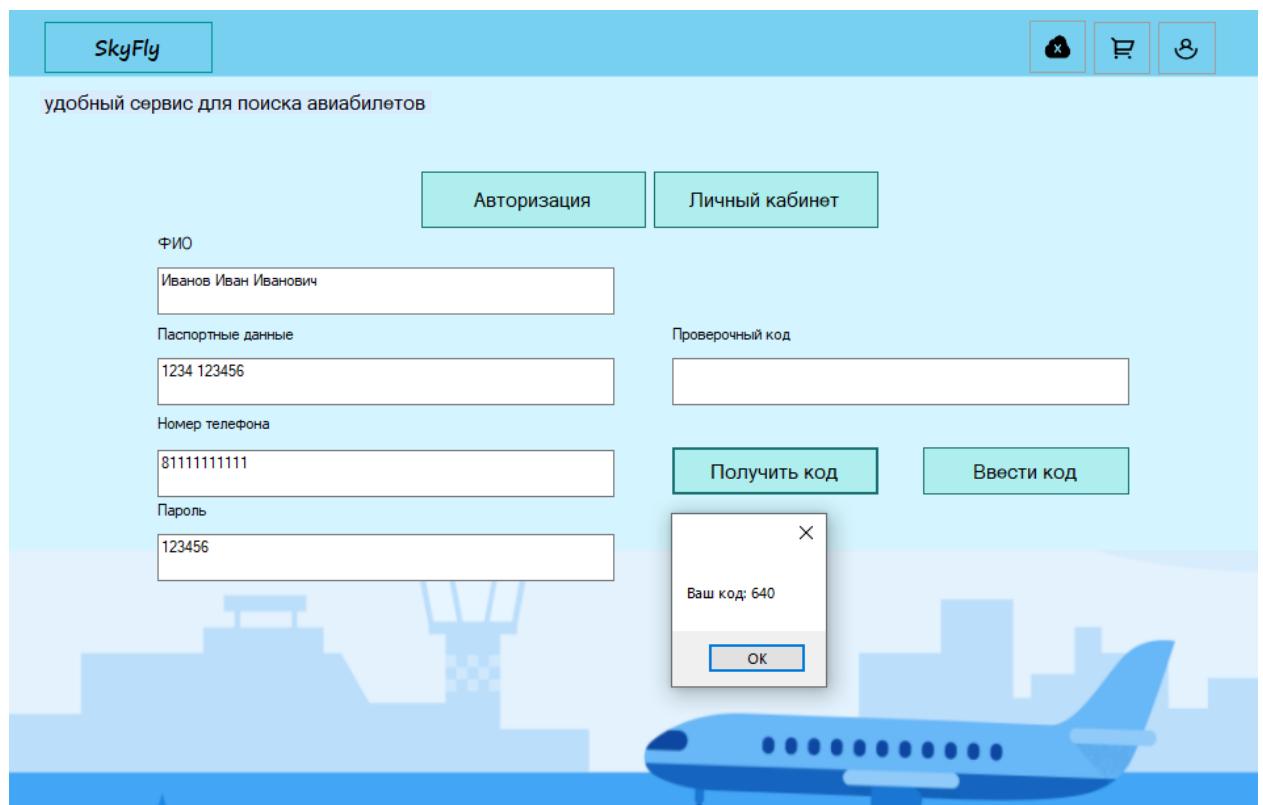


Рисунок 38 - Результат нажатия на кнопку “Получить код”

#### 14. Случай использования: Пользователь вводит проверочный код для регистрации.

Предусловие: программа запущена, открыта страница корзины покупок;

Тестовый случай: Пользователь ввел проверочный код и нажимает на кнопку “Ввести код”.

Ожидаемый результат: Программа проверяет, совпадает ли введенный код с тем, что был в уведомлении, если да, то регистрирует пользователя. Иначе открывает уведомление о неверном коде.

Результат представлен на рисунке 39.

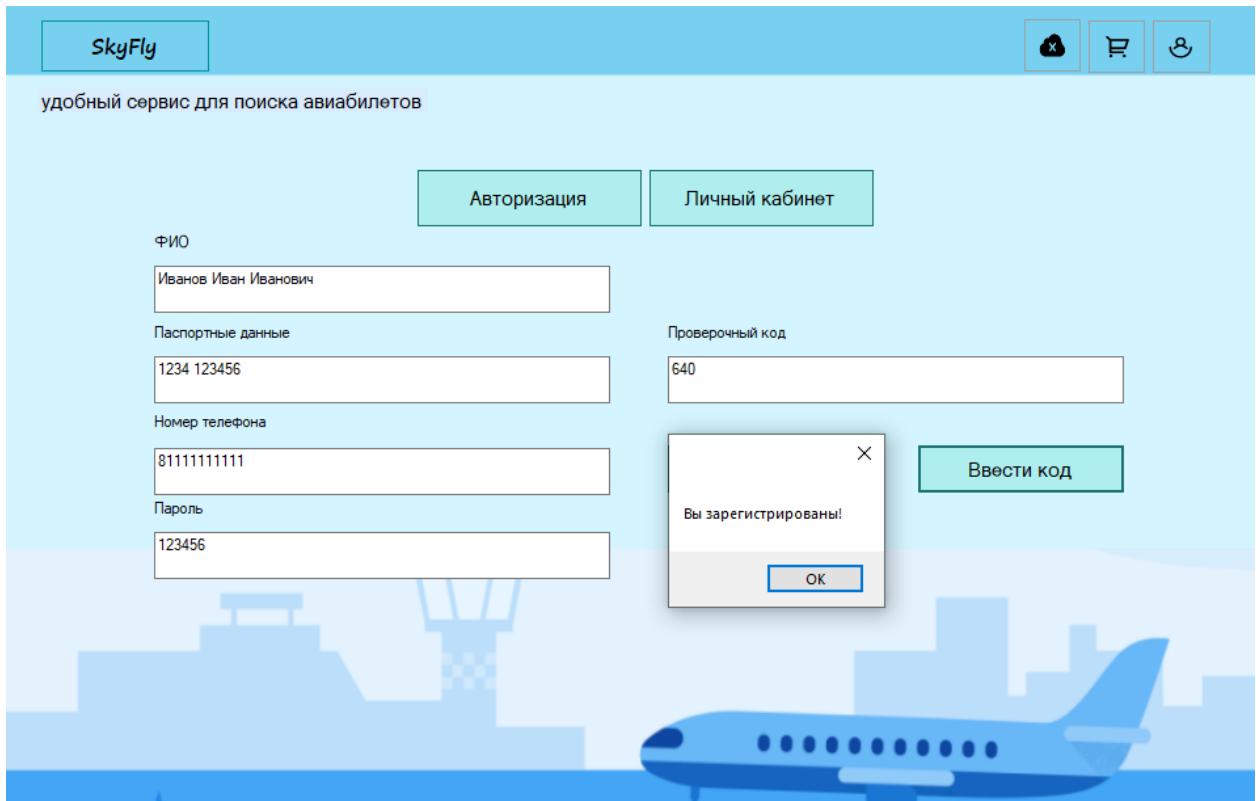


Рисунок 39 - Результат успешной регистрации

## 15. Случай использования: Пользователь открывает страницу для авторизации.

Предусловие: программа запущена, открыта страница для регистрации;

Тестовый случай: Пользователь нажимает кнопку “Авторизация”;

Ожидаемый результат: Программа открывает страницу для авторизации.

Результат представлен на рисунке 40.

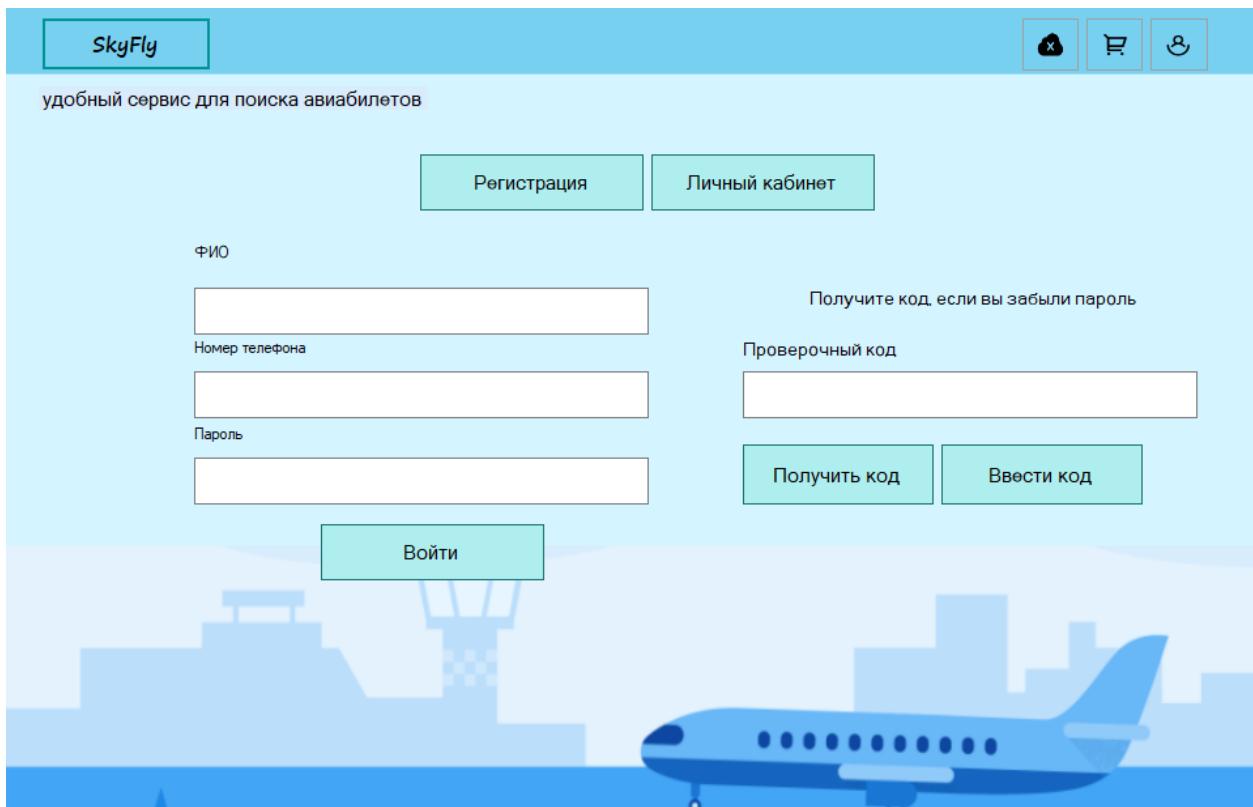


Рисунок 40 - Результат открытия страницы для авторизации

## 16. Случай использования: Пользователь авторизуется в сервисе.

Предусловие: программа запущена, открыта страница для авторизации;

Тестовый случай: Пользователь вводит персональные данные и нажимает на кнопку “Войти”;

Ожидаемый результат: Программа проверяет введенные данные пользователя. Если данные введены корректно и такой пользователь существует (есть данные в файле), программа авторизует пользователя в сервисе.

Результат представлен на рисунке 41.

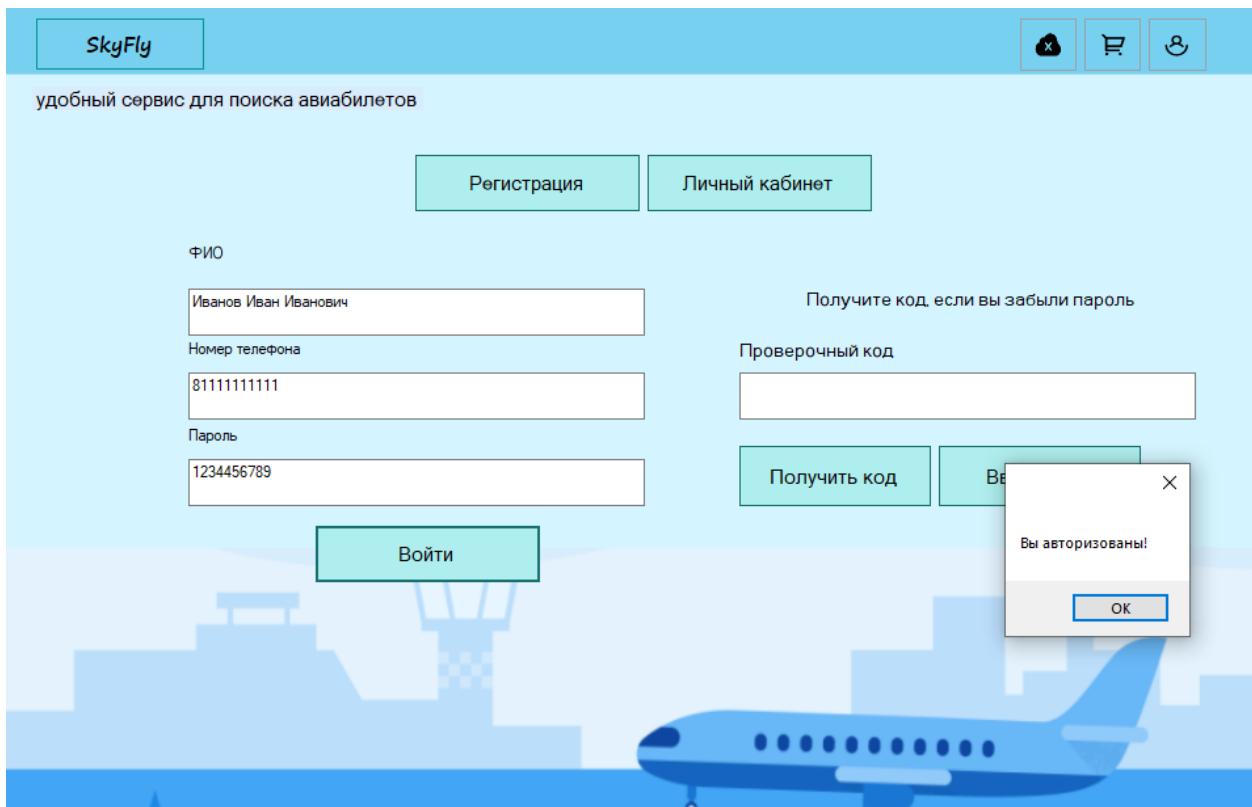


Рисунок 41 - Результат успешной авторизации в сервисе

## 17. Случай использования: Пользователь забыл пароль от личного кабинета.

Предусловие: программа запущена, открыта страница для авторизации;

Тестовый случай: Пользователь ввел персональные данные и нажимает на кнопку “Получить код”;

Ожидаемый результат: Программа проверяет введенные пользователем данные, если данные введены корректно, программа открывает уведомление с проверочным кодом.

Результат представлен на рисунке 42.

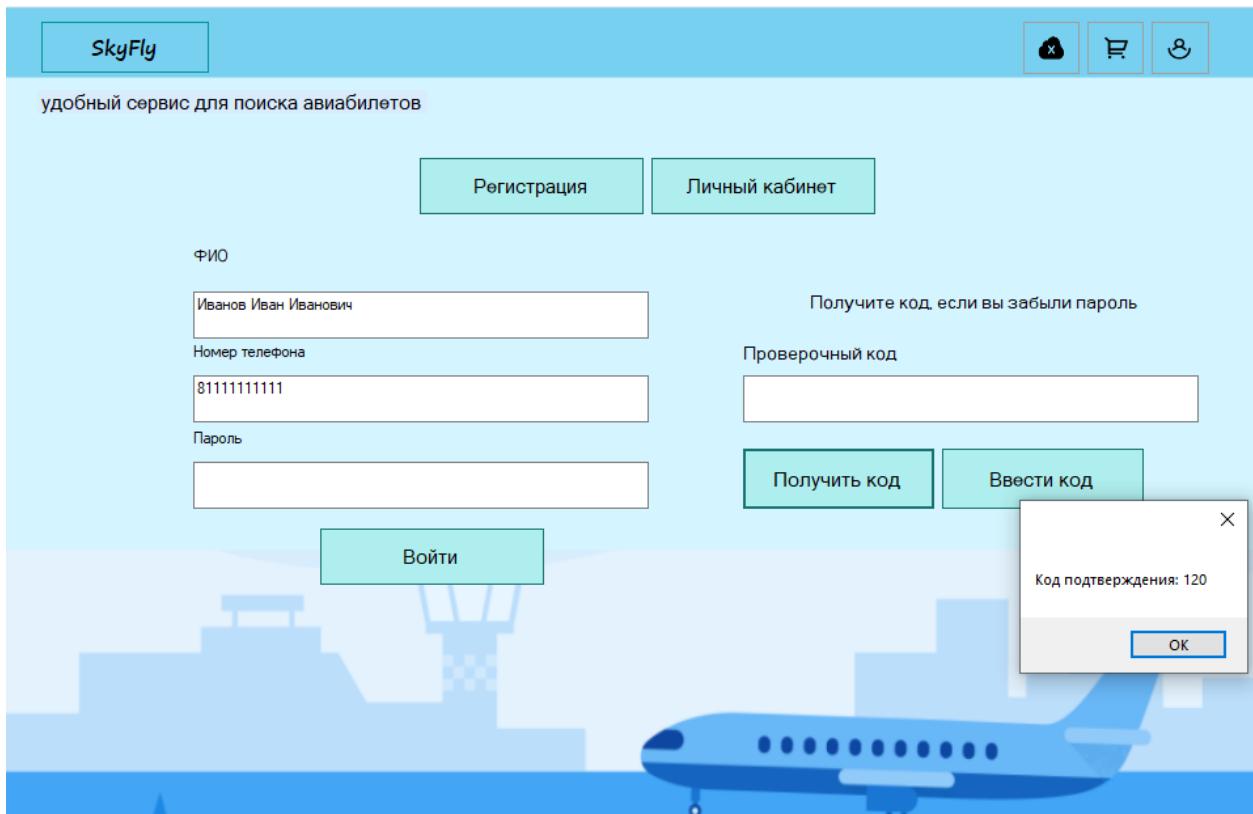


Рисунок 42 - Результат нажатия на кнопку “Получить код”

## 18. Случай использования: Пользователь получает код для авторизации.

Предусловие: программа запущена, открыта страница для авторизации;

Тестовый случай: Пользователь ввел проверочный код и нажимает на кнопку “ Ввести код ”;

Ожидаемый результат: Программа проверяет, совпадает ли введенный код с тем, что был в уведомлении, если да, то авторизует пользователя. Иначе открывает уведомление о неверном коде.

Результат представлен на рисунке 43-44.

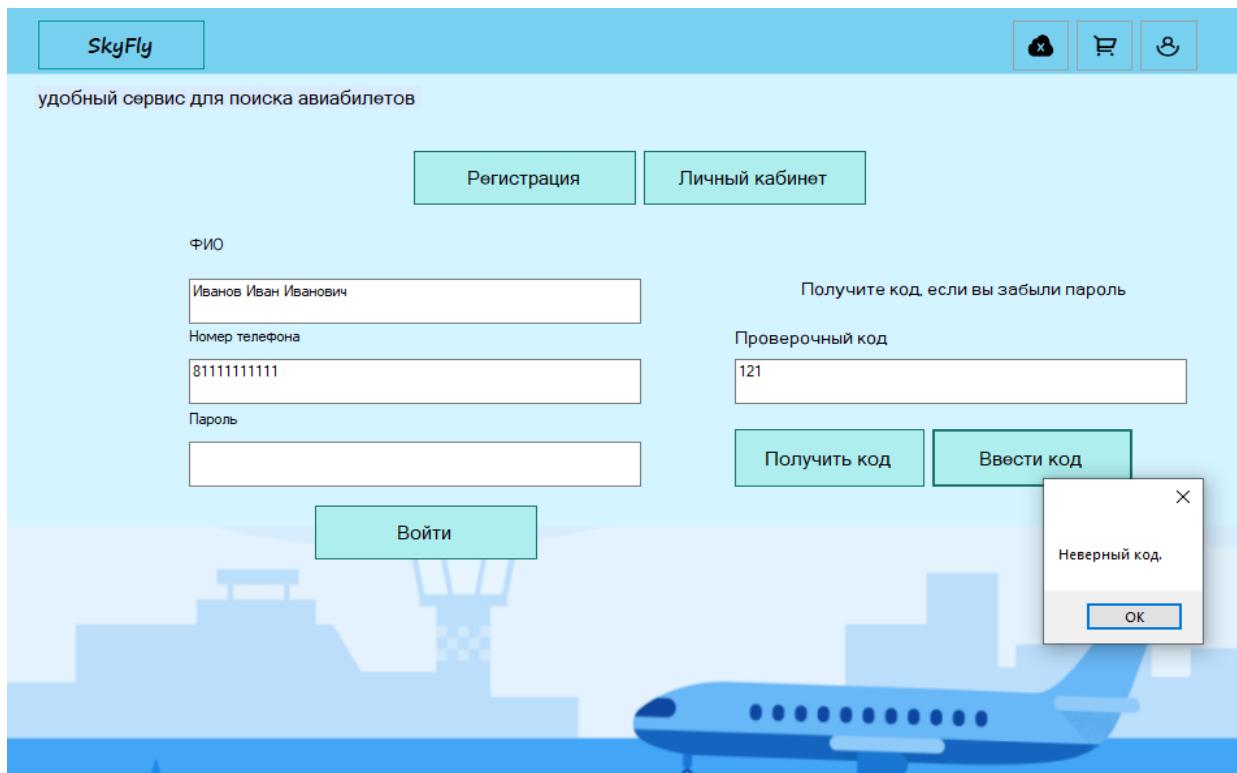


Рисунок 43 - Результат ввода неверного кода

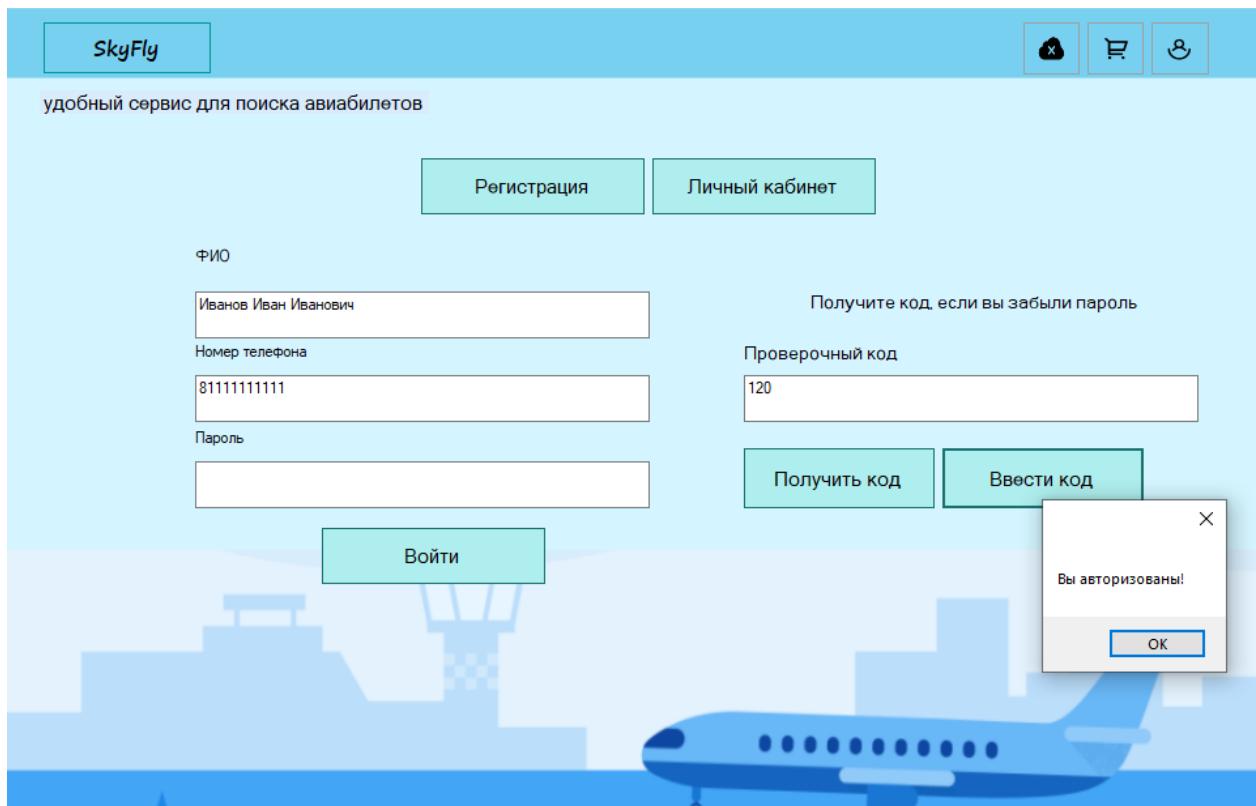


Рисунок 44 - Результат нажатия на кнопку “Ввести код”

## **19. Случай использования: Пользователь меняет данные в личном кабинете.**

Предусловие: программа запущена, открыта страница личного кабинета;

Тестовый случай: Пользователь меняет персональные данные и нажимает на кнопку “Сохранить изменения”;

Ожидаемый результат: Программа проверяет корректно ли внесены изменения. Если данные корректны, программа записывает новые данные пользователя в файл. Иначе выводит уведомление о неправильно введенных данных.

Результат представлен на рисунке 45-46.

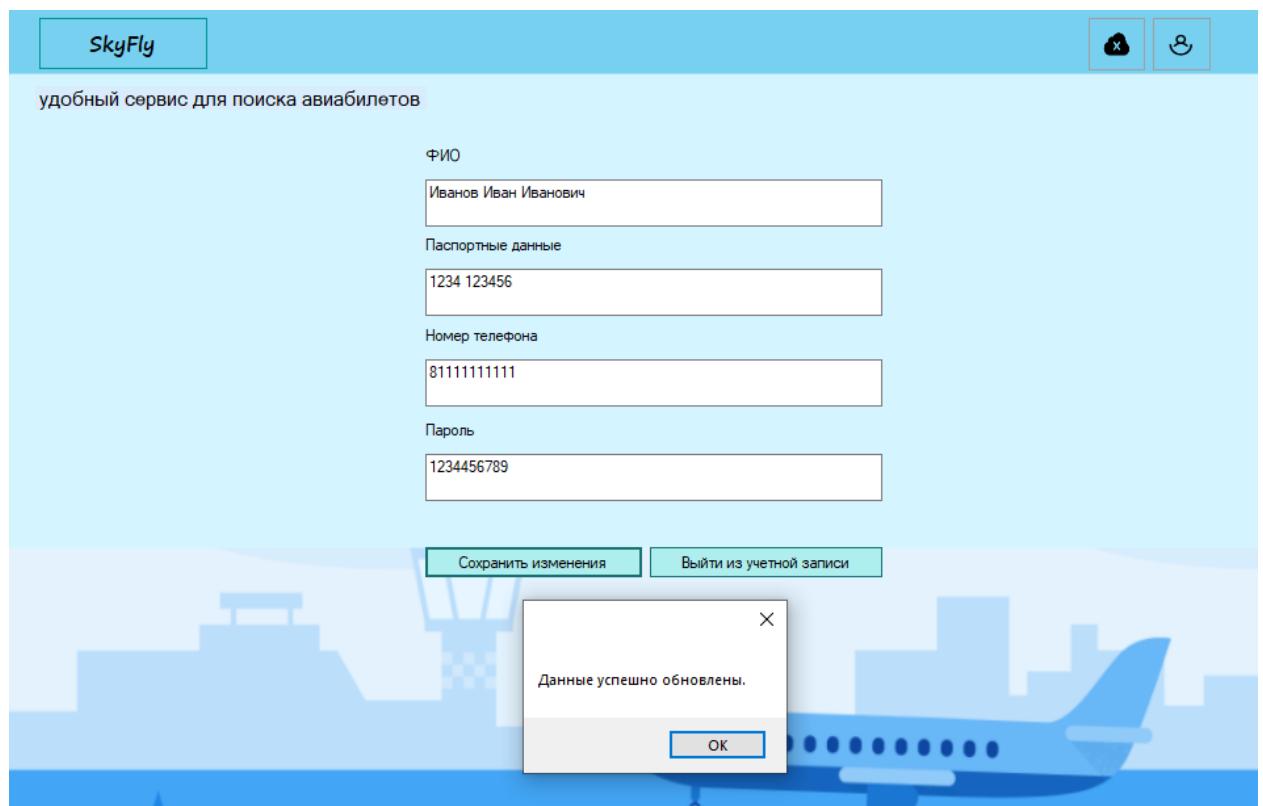


Рисунок 45 - Результат нажатия на кнопку “Сохранить изменения”

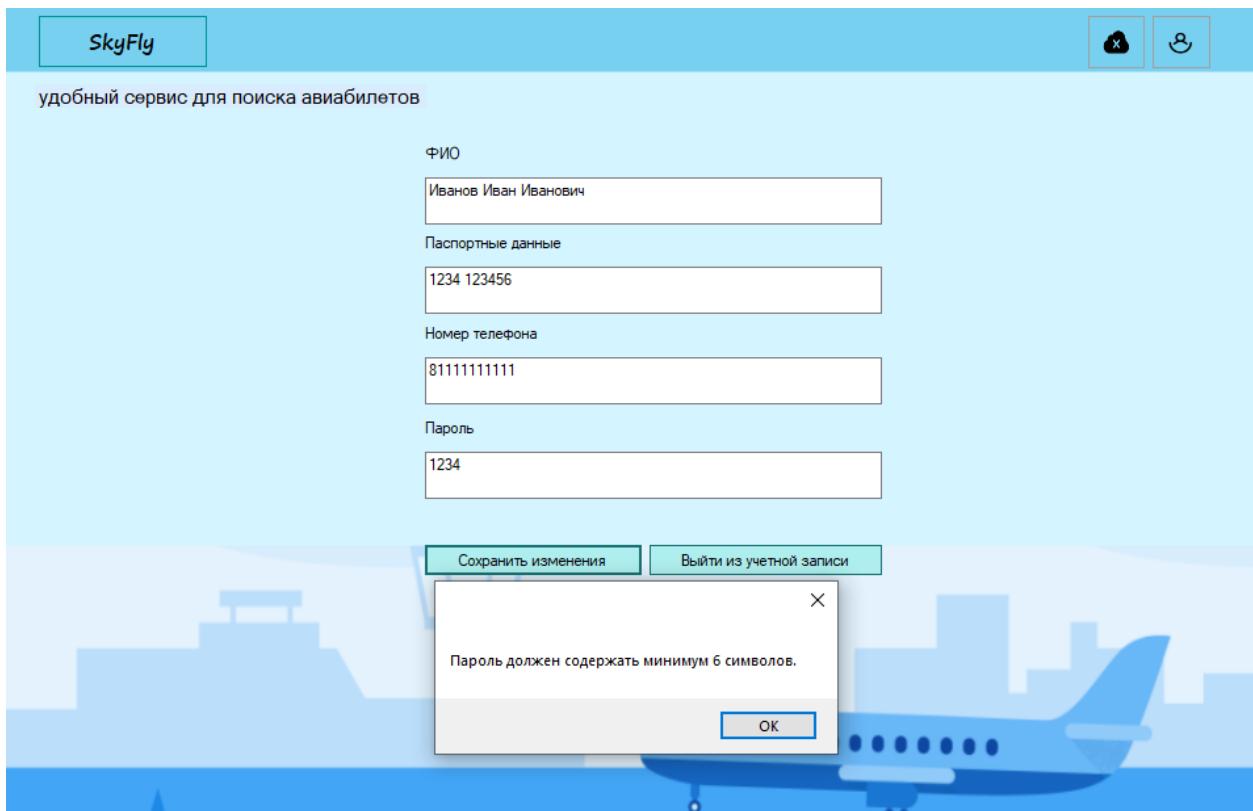


Рисунок 46 – Результат ввода некорректных данных

## 20. Случай использования: Пользователь выходит из учетной записи.

Предусловие: программа запущена, открыта страница личного кабинета;

Тестовый случай: Пользователь нажимает на кнопку “Выйти из учетной записи”;

Ожидаемый результат: Программа очищает все текстовые поля.

Результат представлен на рисунке 47.

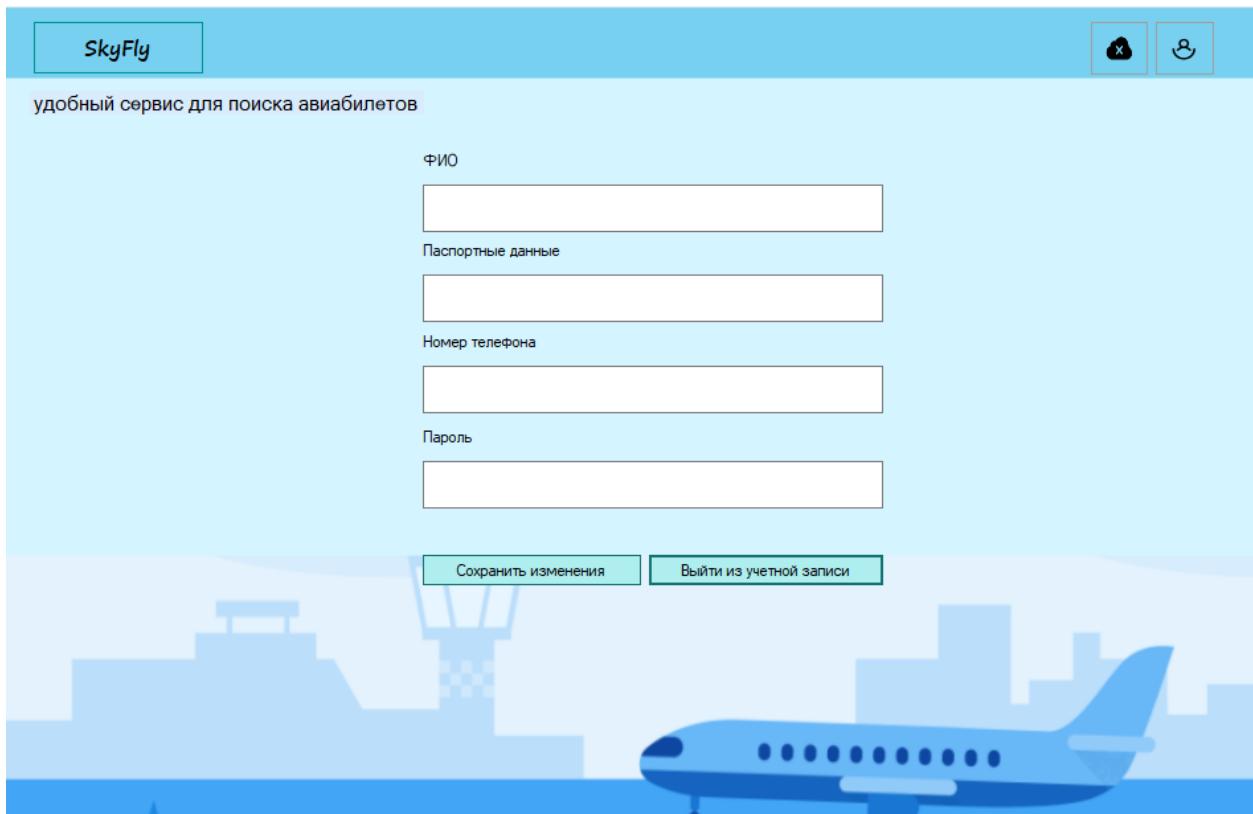


Рисунок 47 – Результат нажатия на кнопку “Выйти из учетной записи”

## **21. Случай использования: Пользователь получает проверочный код для оплаты билета.**

Предусловие: программа запущена, открыта страница для оплаты билета;

Тестовый случай: Пользователь вводит все необходимые данные и нажимает кнопку “Получить код”;

Ожидаемый результат: Программа проверяет введенные пользователем данные, если данные введены корректно, программа открывает уведомление с проверочным кодом.

Результат представлен на рисунке 48-49.

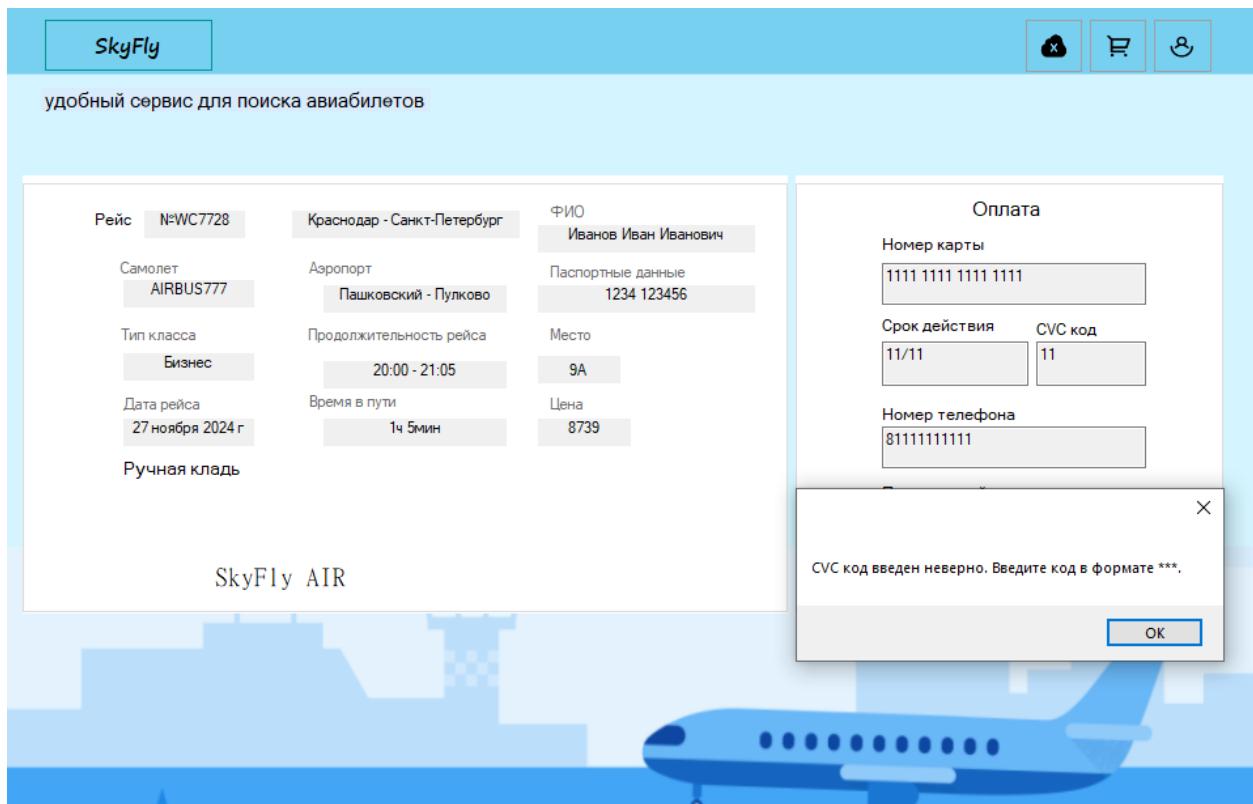


Рисунок 48 – Результат ввода некорректных данных

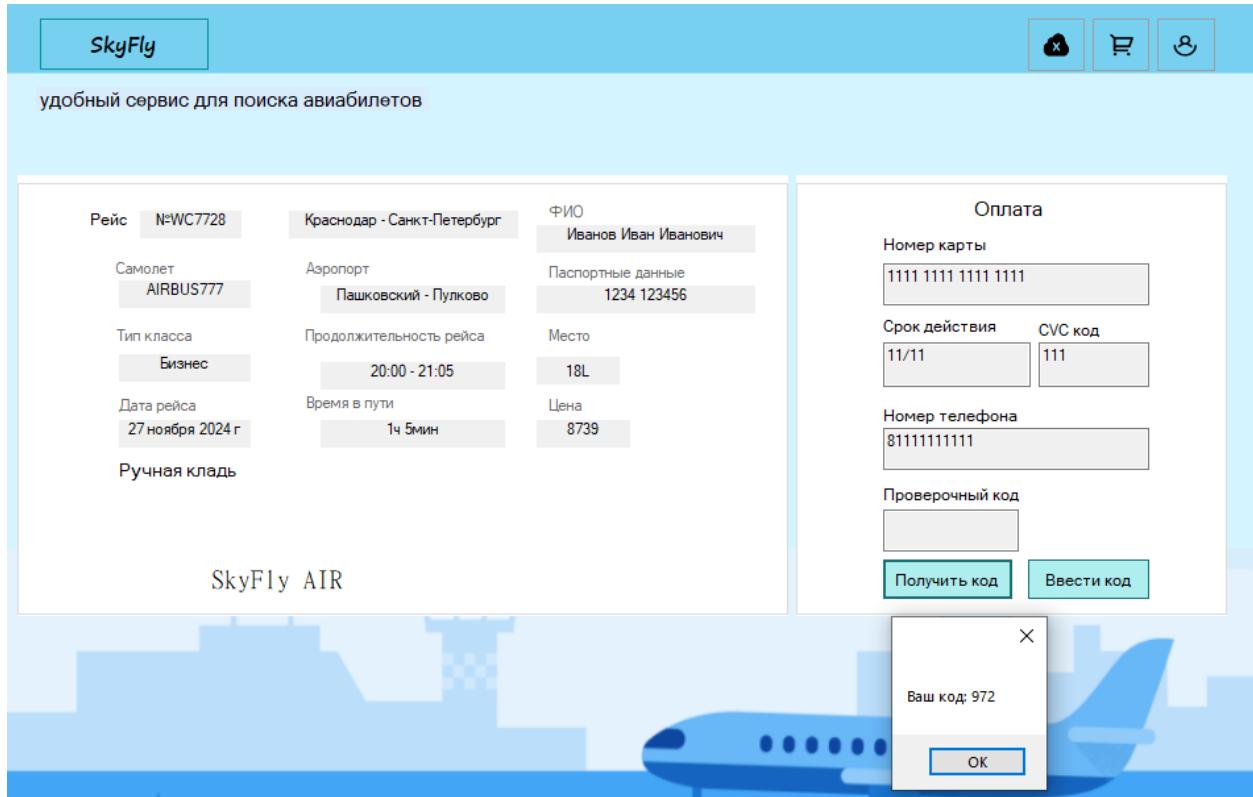


Рисунок 49 – Результат нажатия на кнопку “Получить код”

## 22. Случай использования: Пользователь оплачивает билет.

Предусловие: программа запущена, открыта страница для оплаты билета;

Тестовый случай: Пользователь вводит проверочный код и нажимает кнопку “Ввести код”;

Ожидаемый результат: Программа проверяет, совпадает ли введенный код с тем, что был в уведомлении, если да, то проводит оплату билета пользователя. Иначе открывает уведомление о неверном коде.

Результат представлен на рисунке 50.

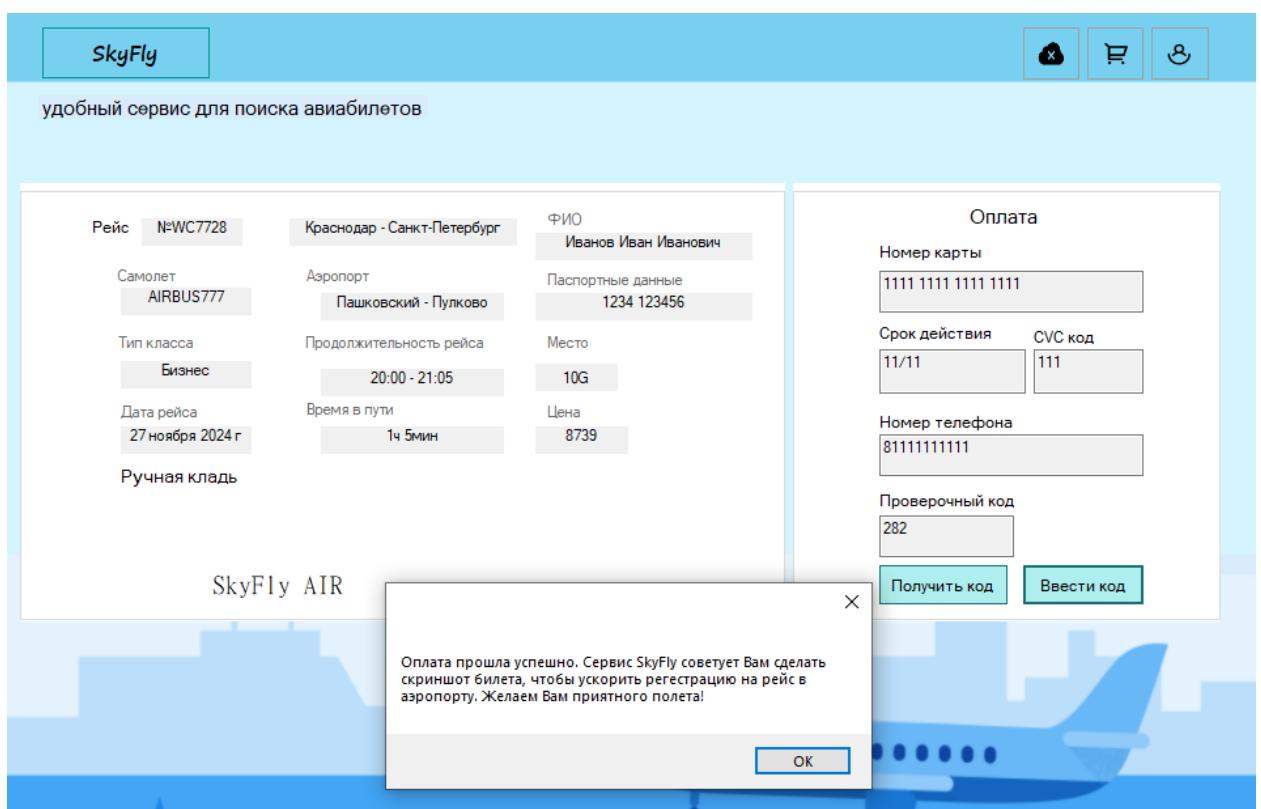


Рисунок 50 – Результат оплаты билета

## 23. Случай использования: Пользователь закрывает приложение и завершает работу программы.

Предусловие: программа запущена, открыта любая страница;

Тестовый случай: Пользователь нажимает на кнопку с изображением облака с крестиком;

Ожидаемый результат: После нажатия на кнопку с изображением облака с крестиком программа завершает работу (можно нажать на любом этапе программы).

## ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была разработана программа для моделирования сервиса для продажи авиабилетов.

Разработанная программа предназначена для улучшения понимания работы сервиса для продажи авиабилетов.

В процессе выполнения курсовой работы на основе исследования предметной области приложения были определены требования к приложению. Для реализации требований к программе разработана логика моделирования; разработан основной алгоритм моделирования; разработана система взаимодействия пользователя и интерфейса программы; спроектирован и реализован графический интерфейс страниц программы; проведено функциональное тестирование программы

Все требования, объявленные в техническом задании, были полностью реализованы в данном программном продукте.

Все задачи, поставленные в начале разработки проекта, были решены.

Таким образом, цель работы достигнута.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Белик, А. Г. Проектирование и архитектура программных систем: учебное пособие / А. Г. Белик, В. Н. Цыганенко. – Омск: ОмГТУ, 2016. – 96 с. – ISBN 978-5-8149-2258-8. – Текст: непосредственный.
2. Мюллер. Джон Пол. Семпф, Билл. Сфер, Чак. C# для чайников. : Пер. с англ. – СПб. : ООО «Диалектика», 2019. – 608 с. – ISBN 978-5-907144-43-9 – Текст: непосредственный.
3. Хорев, П. Объектно-ориентированное программирование с примерами на C#. Учебное пособие / П. Хорев. – Москва : Не Указано, 2022. – 200 с. – ISBN 978-5-00091-713-8.
4. Шилдт, Г. C#: Полное руководство / Г. Шилдт. – Москва: Вильямс, 2021. – ISBN 978-5-8459-1980-9.
5. Мартин, Р. Чистый код. Создание, анализ и рефакторинг / Р. Мартин. – Москва: Манн, Иванов и Фербер, 2020. – ISBN 978-5-00122-151-6.
6. Папалюга, В. C# и .NET: Полное руководство по программированию – Москва: Питер, 2021. – ISBN 978-5-4461-0593-5.
7. Климентьев, А. C# и .NET. Объектно-ориентированное программирование на практике – Москва: БХВ-Петербург, 2022. – ISBN 978-5-9775-0779-9.
8. Тодор, С. Изучаем C# 7.0. Начальный курс / С. Тодор. – Санкт-Петербург: Питер, 2018. – ISBN 978-5-4461-0364-5.
9. Каплун, И. Создание веб-сервиса для онлайн-бронирования авиабилетов / И. Каплун. – Москва: БХВ-Петербург, 2018. – ISBN 978-5-9775-0700-3.
10. Смит, Д. Разработка микросервисов на .NET Core / Д. Смит. – Москва: Вильямс, 2021. – ISBN 978-5-8459-1998-6.

11. Мартин, Р. Чистая архитектура. Искусство структурирования программного обеспечения / Р. Мартин. – Москва: Манн, Иванов и Фербер, 2018. – ISBN 978-5-00122-138-7.
12. Голик, В. Тестирование программного обеспечения и проектирование сервисов / В. Голик. – Санкт-Петербург: Питер, 2022. – ISBN 978-5-4461-0699-0.
13. Робертс, Д. Микросервисы. Паттерны и методы для проектирования и разработки сервисов / Д. Робертс. – Санкт-Петербург: Питер, 2020. – ISBN 978-5-4461-0568-9.
14. Фриман, Э., Риз, Б. C# 8.0 и .NET Core 3.0. Современный подход / Э. Фриман, Б. Риз. – Санкт-Петербург: Питер, 2020. – ISBN 978-5-4461-0540-9.
15. Левицкий, И. Разработка веб-приложений на ASP.NET Core / И. Левицкий. – Москва: БХВ-Петербург, 2020. – ISBN 978-5-9775-0816-8.

## ПРИЛОЖЕНИЕ А

### Исходный код Form 1:

```
using System;
using System.Windows.Forms;

namespace Курсовая_работа_Машенцева
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Hide();

            Form2 form2 = new Form2();
            form2.Show();
        }
    }
}
```

### Исходный код Form 2:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Windows.Forms;

namespace Курсовая_работа_Машенцева
{
    public partial class Form2 : Form
    {
        private List<string> cities = new List<string>
        { "Анапа", "Владивосток", "Воронеж", "Краснодар",
        "Москва", "Мурманск", "Омск",
        "Пермь", "Санкт-Петербург", "Томск" };
        private List<string> classes = new List<string> { "Эконом", "Бизнес" };
        public DateTime SelectedDate { get; private set; }

        public Form2()
        {
            InitializeComponent();
            // Инициализация календаря
            dateTimePicker1.MinDate = DateTime.Now; // Не раньше текущей даты
            dateTimePicker1.MaxDate = DateTime.Now.AddDays(21); // Не позже трех недель
        }

        private void label7_Click(object sender, EventArgs e)
        {
            string selectedCity = ShowSelectionDialog("Выберите город:", cities);
            if (!string.IsNullOrEmpty(selectedCity))
            {
                label7.Text = selectedCity;
                CheckCities();
            }
        }

        private void label8_Click(object sender, EventArgs e)
        {
            string selectedCity = ShowSelectionDialog("Выберите город:", cities);
```

```

if (!string.IsNullOrEmpty(selectedCity))
{
    label8.Text = selectedCity;
    CheckCities();
}
}

private void label9_Click(object sender, EventArgs e)
{
    string selectedClass = ShowSelectionDialog("Выберите тип класса:", classes);
    if (!string.IsNullOrEmpty(selectedClass))
    {
        label9.Text = selectedClass;
    }
}

private string ShowSelectionDialog(string title, List<string> options)
{
    Form selectionForm = new Form { Text = title, Width = 200, Height = 300 };
    ListBox listBox = new ListBox { Dock = DockStyle.Fill };
    listBox.Items.AddRange(options.ToArray());
    listBox.DoubleClick += (s, e) => { selectionForm.DialogResult = DialogResult.OK; selectionForm.Close(); };
    selectionForm.Controls.Add(listBox);
    if (selectionForm.ShowDialog() == DialogResult.OK)
    {
        return listBox.SelectedItem.ToString();
    }
    return null;
}

private void CheckCities()
{
    if (label7.Text == label8.Text && !string.IsNullOrEmpty(label9.Text) && !string.IsNullOrEmpty(label8.Text))
    {
        MessageBox.Show("Вы ввели два одинаковых города!", "Ошибка", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}

private void SaveDataToFile()
{
    string filePath = @"F:\База данных\Рейсы\flight_data.txt";

    try
    {
        using (StreamWriter writer = new StreamWriter(filePath, true)) // true добавляет запись в конец файла
        {
            writer.WriteLine($"Откуда: {label7.Text}");
            writer.WriteLine($"Куда: {label8.Text}");
            // Форматируем дату из dateTimePicker1
            string formattedDate = dateTimePicker1.Value.ToString("dd.MM.yyyy");
            writer.WriteLine($"Дата отправления: {formattedDate}");
            writer.WriteLine($"Класс: {label9.Text}");
            writer.WriteLine("-----"); // Разделитель для читабельности
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка при сохранении данных: " + ex.Message);
    }
}

private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
{
}

```

```

    SelectedDate = dateTimePicker1.Value;
    // Ваш код дальнейшей обработки даты
}

private void button1_Click(object sender, EventArgs e)
{
    this.Hide();

    Form2 form2 = new Form2();
    form2.Show();
}

private void button8_Click(object sender, EventArgs e)
{
    this.Close();

    // Завершаем приложение
    Application.Exit();
}

private void button2_Click(object sender, EventArgs e)
{
    this.Hide();

    Form4 form4 = new Form4();
    form4.Show();
}

private void button3_Click(object sender, EventArgs e)
{
    this.Hide();

    Form6 form6 = new Form6();
    form6.Show();
}

private void button4_Click(object sender, EventArgs e)
{
    this.Hide();

    Form3 form3 = new Form3();
    form3.Show();

    SaveDataToFile();
}
}
}
}

```

### Исходный код Form 3:

```

using System;
using System.IO;
using System.Windows.Forms;

namespace Курсовая_работа_Машенцева
{
    public partial class Form3 : Form
    {
        public Form3()
        {
            InitializeComponent();
        }

        private void button6_Click(object sender, EventArgs e)
        {

```

```

string filePath = @"F:\База данных\Рейсы\flight_data.txt";
Reys reys = new Reys(filePath);

textBox1.Text = reys.NomEr; // Номер рейса
textBox4.Text = $"{reys.Otkuda} - {reys.Kuda}"; // Откуда - Куда
textBox6.Text = reys.DataOtpravleniya.ToString("dd ММММ yyyy г"); // Дата отправления
textBox3.Text = reys.Klass; // Класс
textBox5.Text = $"{reys.AeroportOtkuda} - {reys.AeroportKuda}"; // Аэропорт отправления - Аэропорт
прибытия
textBox7.Text = $"{reys.VremyaVyleta:hh\:mm} - {reys.VremyaPrileta:hh\:mm}"; // Время вылета - Время
прилета
textBox2.Text = reys.ModelSamolyota; // Модель самолета
textBox8.Text = $"{reys.VremyaVPute.Hours} ч {reys.VremyaVPute.Minutes} мин"; // Время в пути
textBox9.Text = reys.Tsena.ToString(); // Цена

}

private void SaveDataToFile()
{
    string filePath = @"F:\База данных\Найденные рейсы\gotovyreys_data.txt";

    try
    {
        using (StreamWriter writer = new StreamWriter(filePath, true)) // true добавляет запись в конец файла
        {
            writer.WriteLine($"Номер рейса: {textBox1.Text}");
            writer.WriteLine($"Маршрут: {textBox4.Text}");
            writer.WriteLine($"Дата рейса: {textBox6.Text}");
            writer.WriteLine($"Модель самолёта: {textBox2.Text}");
            writer.WriteLine($"Тип класса: {textBox3.Text}");
            writer.WriteLine($"Аэропорт: {textBox5.Text}");
            writer.WriteLine($"Продолжительность рейса: {textBox7.Text}");
            writer.WriteLine($"Время в пути: {textBox8.Text}");
            writer.WriteLine($"Цена: {textBox9.Text}");
            writer.WriteLine("-----"); // Разделитель для читабельности
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка при сохранении данных: " + ex.Message);
    }
}
private void button4_Click_1(object sender, EventArgs e)
{
    SaveDataToFile();
}

private void button1_Click(object sender, EventArgs e)
{
    this.Hide();

    Form2 form2 = new Form2();
    form2.Show();
}

private void button8_Click(object sender, EventArgs e)
{
    this.Close();

    // Завершаем приложение
    Application.Exit();
}

```

```

private void button2_Click(object sender, EventArgs e)
{
    this.Hide();

    Form4 form4 = new Form4();
    form4.Show();
}

private void button3_Click(object sender, EventArgs e)
{
    this.Hide();

    Form6 form6 = new Form6();
    form6.Show();
}
}

```

### Исходный код Form 4:

```

using System;
using System.Windows.Forms;

namespace Курсовая_работа_Машенцева
{
    public partial class Form4 : Form
    {
        private string filePath = @"F:\База данных\Найденные рейсы\gotovyreys_data.txt";
        public string Seat { get; set; }
        public Form4()
        {
            InitializeComponent();
            GenerateSeat();
            LoadUserData();
        }
        private void LoadUserData()
        {
            try
            {
                // Чтение всех строк из файла
                string[] lines = System.IO.File.ReadAllLines(filePath);

                if (lines.Length >= 8)
                {
                    // Извлечение данных из строк
                    string NomerReysa = lines[0].Split(new[] { ';' }, 2)[1].Trim();
                    string Marshrut = lines[1].Split(new[] { ';' }, 2)[1].Trim();
                    string DateNumber = lines[2].Split(new[] { ';' }, 2)[1].Trim();
                    string ModelSamolyta = lines[3].Split(new[] { ';' }, 2)[1].Trim();
                    string TipKlassa = lines[4].Split(new[] { ';' }, 2)[1].Trim();
                    string Airport = lines[5].Split(new[] { ';' }, 2)[1].Trim();
                    string ProdReysa = lines[6].Split(new[] { ';' }, 2)[1].Trim();
                    string VremVPute = lines[7].Split(new[] { ';' }, 2)[1].Trim();
                    string Tsena = lines[8].Split(new[] { ';' }, 2)[1].Trim();

                    // Заполнение текстовых полей
                    textBox1.Text = NomerReysa;
                    textBox5.Text = Marshrut;
                    textBox4.Text = DateNumber;
                    textBox2.Text = ModelSamolyta;
                    textBox3.Text = TipKlassa;
                    textBox6.Text = Airport;
                    textBox7.Text = ProdReysa;
                    textBox8.Text = VremVPute;
                    textBox9.Text = Tsena;
                }
            }
        }
    }
}

```

```

// Применение модификаций по CheckBox
UpdateTsena();

}

else
{
    MessageBox.Show("Недостаточно данных в файле.");
}

}

catch (Exception ex)
{
    MessageBox.Show("Произошла ошибка при чтении файла: " + ex.Message);
}
}

private void GenerateSeat()
{
    Random random = new Random();
    int row = random.Next(1, 22); // допустимые номера рядов от 1 до 22
    char seatLetter = (char)(A' + random.Next(0, 12)); // допустимые буквы от A до L
    Seat = $"{{row}}{{seatLetter}}";
    label4.Text = Seat;
}

private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    UpdateTsena();
}

private void checkBox2_CheckedChanged(object sender, EventArgs e)
{
    UpdateTsena();
}

private void UpdateTsena()
{// Проверка на пустое значение textBox9
if (string.IsNullOrWhiteSpace(textBox9.Text))
{
    // Если значение пустое, просто возвращаемся, ничего не делая
    return;
}

// Попытка преобразовать текст в decimal
if (!decimal.TryParse(textBox9.Text, out decimal tsena))
{
    // Если формат неверный, можно вывести сообщение
    MessageBox.Show("Неверный формат цены в поле textBox9.");
    return; // Выход из метода, если формат неверный
}

// Если чекбоксы отмечены, добавляем соответствующие цены
if (checkBox1.Checked)
{
    tsena += 700;
}
if (checkBox2.Checked)
{
    tsena += 700;
}
if (checkBox3.Checked)
{
    tsena += 1050;
}
}

```

```

// Обновляем значение в textBox9
textBox9.Text = tsena.ToString();
}

private void checkBox3_CheckedChanged(object sender, EventArgs e)
{
    UpdateTsena();
    this.Hide();
    Form5 form5 = new Form5();
    form5.Show();
}

private void button8_Click(object sender, EventArgs e)
{
    this.Close();

    // Завершаем приложение
    Application.Exit();
}

private void button2_Click(object sender, EventArgs e)
{
    this.Hide();

    Form4 form4 = new Form4();
    form4.Show();
}

private void button3_Click(object sender, EventArgs e)
{
    this.Hide();

    Form6 form6 = new Form6();
    form6.Show();
}

private void button1_Click(object sender, EventArgs e)
{
    this.Hide();

    Form2 form2 = new Form2();
    form2.Show();
}

public string GetSelectedClassType()
{
    return textBox3.Text; // Возвращаем текст из label25
}

public void SetLabelValue(string value)
{
    label4.Text = value;
}

private void button4_Click(object sender, EventArgs e)
{
    // Очищаем все поля
    textBox1.Text = string.Empty;
    textBox2.Text = string.Empty;
    textBox3.Text = string.Empty;
    textBox4.Text = string.Empty;
    textBox5.Text = string.Empty;
    textBox6.Text = string.Empty;
    textBox7.Text = string.Empty;
    textBox8.Text = string.Empty;
    textBox9.Text = string.Empty;
    label4.Text = string.Empty;
}

```

```
// Сбрасываем состояние чекбоксов
checkBox1.Checked = false;
checkBox2.Checked = false;
}

private void button5_Click(object sender, EventArgs e)
{
    string path = @"F:\База данных\Регистрация\user_data.txt";

    if (!System.IO.File.Exists(path))
    {
        MessageBox.Show("Для перехода к оплате авторизуйтесь или зарегистрируйтесь на сервисе.",
                       "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
    else
    {
        this.Hide();
        Form8 form8 = new Form8();
        form8.Label4Text = label4.Text;
        form8.Show();
    }
}
```

## Исходный код Form 5:

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace Курсовая_работа_Машенцева
{
    public partial class Form5 : Form
    {
        private Label selectedLabel = null; // Переменная для отслеживания выбранного label.
        private Color defaultColor = Color.Black; // Цвет по умолчанию для меток.
        private string classType; // Хранит тип класса (эконом, бизнес)

        public Form5()
        {
            InitializeComponent();
            InitializeLabels();
            LoadClassType();
        }

        private void LoadClassType()
        {
            // Получаем тип класса из Form4
            Form4 form4 = (Form4)Application.OpenForms["Form4"];
            if (form4 != null)
            {
                classType = form4.GetSelectedClassType();
            }
        }

        private void InitializeLabels()
        {
            label2.Text = "К1"; label2.Click += Label_Click;
            label11.Text = "А1"; label11.Click += Label_Click;
            label10.Text = "Б1"; label10.Click += Label_Click;
            label13.Text = "А2"; label13.Click += Label_Click;
            label12.Text = "Б2"; label12.Click += Label_Click;
            label15.Text = "А3"; label15.Click += Label_Click;
            label14.Text = "Б3"; label14.Click += Label_Click;
        }
    }
}
```

```

label17.Text = "A4"; label17.Click += Label_Click;
label16.Text = "B4"; label16.Click += Label_Click;
label5.Text = "J2"; label5.Click += Label_Click;
label4.Text = "K2"; label4.Click += Label_Click;
label7.Text = "J3"; label7.Click += Label_Click;
label6.Text = "K3"; label6.Click += Label_Click;
label9.Text = "J4"; label9.Click += Label_Click;
label8.Text = "K4"; label8.Click += Label_Click;

label20.Text = "J5"; label20.Click += Label_Click;
label18.Text = "L5"; label18.Click += Label_Click;
label32.Text = "A5"; label32.Click += Label_Click;
label31.Text = "B5"; label31.Click += Label_Click;
label30.Text = "C5"; label30.Click += Label_Click;
label33.Text = "C6"; label33.Click += Label_Click;
label23.Text = "J6"; label23.Click += Label_Click;
label22.Text = "K6"; label22.Click += Label_Click;
label21.Text = "L6"; label21.Click += Label_Click;
label38.Text = "A7"; label38.Click += Label_Click;
label37.Text = "B7"; label37.Click += Label_Click;
label36.Text = "C7"; label36.Click += Label_Click;
label26.Text = "J7"; label26.Click += Label_Click;
label25.Text = "K7"; label25.Click += Label_Click;
label24.Text = "L7"; label24.Click += Label_Click;
label41.Text = "A8"; label41.Click += Label_Click;
label40.Text = "B8"; label40.Click += Label_Click;
label39.Text = "C8"; label39.Click += Label_Click;
label29.Text = "J8"; label29.Click += Label_Click;
label28.Text = "K8"; label28.Click += Label_Click;
label27.Text = "L8"; label27.Click += Label_Click;
label45.Text = "B9"; label45.Click += Label_Click;
label44.Text = "C9"; label44.Click += Label_Click;
label43.Text = "J9"; label43.Click += Label_Click;
label42.Text = "K9"; label42.Click += Label_Click;
label51.Text = "A10"; label51.Click += Label_Click;
label50.Text = "B10"; label50.Click += Label_Click;
label49.Text = "C10"; label49.Click += Label_Click;
label48.Text = "J10"; label48.Click += Label_Click;
label46.Text = "L10"; label46.Click += Label_Click;
label87.Text = "A11"; label87.Click += Label_Click;
label86.Text = "B11"; label86.Click += Label_Click;
label85.Text = "C11"; label85.Click += Label_Click;
label54.Text = "J11"; label54.Click += Label_Click;
label53.Text = "K11"; label53.Click += Label_Click;
label52.Text = "L11"; label52.Click += Label_Click;
label90.Text = "A12"; label90.Click += Label_Click;
label89.Text = "B12"; label89.Click += Label_Click;
label88.Text = "C12"; label88.Click += Label_Click;
label57.Text = "J12"; label57.Click += Label_Click;
label56.Text = "K12"; label56.Click += Label_Click;
label55.Text = "L12"; label55.Click += Label_Click;
label93.Text = "A13"; label93.Click += Label_Click;
label60.Text = "J13"; label60.Click += Label_Click;
label59.Text = "K13"; label59.Click += Label_Click;
label58.Text = "L13"; label58.Click += Label_Click;
label96.Text = "A14"; label96.Click += Label_Click;
label95.Text = "B14"; label95.Click += Label_Click;
label94.Text = "C14"; label94.Click += Label_Click;
label63.Text = "J14"; label63.Click += Label_Click;
label62.Text = "K14"; label62.Click += Label_Click;
label61.Text = "L14"; label61.Click += Label_Click;
label99.Text = "A15"; label99.Click += Label_Click;
label98.Text = "B15"; label98.Click += Label_Click;
label97.Text = "C15"; label97.Click += Label_Click;
label66.Text = "J15"; label66.Click += Label_Click;
label65.Text = "K15"; label65.Click += Label_Click;

```

```

label64.Text = "L15"; label64.Click += Label_Click;
label102.Text = "A16"; label102.Click += Label_Click;
label101.Text = "B16"; label101.Click += Label_Click;
label100.Text = "C16"; label100.Click += Label_Click;
label69.Text = "J16"; label69.Click += Label_Click;
label105.Text = "A17"; label105.Click += Label_Click;
label104.Text = "B17"; label104.Click += Label_Click;
label103.Text = "C17"; label103.Click += Label_Click;
label72.Text = "J17"; label72.Click += Label_Click;
label71.Text = "K17"; label71.Click += Label_Click;
label70.Text = "L17"; label70.Click += Label_Click;
label108.Text = "A18"; label108.Click += Label_Click;
label107.Text = "B18"; label107.Click += Label_Click;
label106.Text = "C18"; label106.Click += Label_Click;
label75.Text = "J18"; label75.Click += Label_Click;
label73.Text = "L18"; label73.Click += Label_Click;
label111.Text = "A19"; label111.Click += Label_Click;
label110.Text = "B19"; label110.Click += Label_Click;
label109.Text = "C19"; label109.Click += Label_Click;
label78.Text = "J19"; label78.Click += Label_Click;
label77.Text = "K19"; label77.Click += Label_Click;
label76.Text = "L19"; label76.Click += Label_Click;
label114.Text = "A20"; label114.Click += Label_Click;
label113.Text = "B20"; label113.Click += Label_Click;
label112.Text = "C20"; label112.Click += Label_Click;
label81.Text = "J20"; label81.Click += Label_Click;
label80.Text = "K20"; label80.Click += Label_Click;
label79.Text = "L20"; label79.Click += Label_Click;
label116.Text = "B21"; label116.Click += Label_Click;
label115.Text = "C21"; label115.Click += Label_Click;
label84.Text = "J21"; label84.Click += Label_Click;
label83.Text = "K21"; label83.Click += Label_Click;
label82.Text = "L21"; label82.Click += Label_Click;
}
private void Label_Click(object sender, EventArgs e)
{
    Label clickedLabel = sender as Label;

    if (clickedLabel != null)
    {
        if (IsValidSelection(clickedLabel))
        {
            selectedLabel = clickedLabel;
            selectedLabel.ForeColor = Color.Red; // Если метка выбрана, изменяем цвет
            // Здесь можно вызвать код для обработки выбора метки
        }
        else
        {
            MessageBox.Show("Эта метка не соответствует выбранному типу класса.");
        }
    }
}
private bool IsValidSelection(Label clickedLabel)
{
    // Проверяем, является ли класс бизнес или эконом
    // Логика для проверки относится ли метка к бизнес или эконом классу

    if (string.IsNullOrEmpty(classType))
    {
        // Если класс не выбран, возвращаем false
        return false;
    }

    // Определяем, является ли место бизнес или эконом местом
    if (classType == "Бизнес")
    {

```

## Исходный код Form 6.

```
using System;  
using System.Windows.Forms;
```

```

namespace Курсовая_работа_Машенцева
{
    public partial class Form6 : Form
    {
        private UserRegistration userRegistration = new UserRegistration();
        private CodeGenerator codeGenerator = new CodeGenerator();

        public Form6()
        {
            InitializeComponent();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            // Проверка введенных данных
            if (userRegistration.ValidateInput(textBox1, textBox2, textBox3, textBox4, textBox5))
            {
                userRegistration.Code = codeGenerator.GenerateCode(); // Генерация кода
                MessageBox.Show($"Ваш код: {userRegistration.Code}"); // Уведомление с кодом
            }
        }

        private void button7_Click(object sender, EventArgs e)
        {
            // Проверка ввода кода
            if (textBox5.Text == userRegistration.Code)
            {
                MessageBox.Show("Вы зарегистрированы!");
            }
            else
            {
                MessageBox.Show("Неверный код.");
            }
            // Создаем экземпляр класса UserRegistration
            UserRegistration registration = new UserRegistration(
                textBox1.Text, // текстбокс для ФИО
                textBox2.Text, // Здесь паспортные данные
                textBox3.Text, // Здесь номер телефона
                textBox4.Text, // Здесь пароль
                textBox5.Text // Здесь код
            );

            // Сохранение данных
            registration.SaveToFile("user_data.txt"); // Имя файла для сохранения данных
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Hide();

            Form2 form2 = new Form2();
            form2.Show();
        }

        private void button8_Click(object sender, EventArgs e)

```

```
{  
    this.Close();  
  
    // Завершаем приложение  
    Application.Exit();  
}  
  
private void button9_Click(object sender, EventArgs e)  
{  
    this.Hide();  
  
    Form4 form4 = new Form4();  
    form4.Show();  
}  
  
private void button3_Click(object sender, EventArgs e)  
{  
    this.Hide();  
  
    Form6 form6 = new Form6();  
    form6.Show();  
}  
  
private void button4_Click(object sender, EventArgs e)  
{  
  
}  
  
private void button5_Click(object sender, EventArgs e)  
{  
    this.Hide();  
  
    Form7 form7 = new Form7();  
    form7.Show();  
}  
  
private void button2_Click(object sender, EventArgs e)  
{  
    this.Hide();  
  
    Form9 form9 = new Form9();  
    form9.Show();  
}  
}
```

## Исходный код Form 7:

```
using System;
using System.IO;
using System.Windows.Forms;

namespace Курсовая_работа_Машенцева
{
    public partial class Form7 : Form
    {
        private string generatedCode;
```

```

public Form7()
{
    InitializeComponent();
}

private void button6_Click(object sender, EventArgs e)
{
    // Получаем введенные данные
    string fullName = textBox4.Text.Trim(); // Добавлено Trim() для удаления пробелов
    string phoneNumber = textBox1.Text.Trim(); // Добавлено Trim() для удаления пробелов
    string password = textBox2.Text.Trim(); // Добавлено Trim() для удаления пробелов

    // Указываем путь к файлу с данными
    string filePath = @"F:\База данных\Регистрация\user_data.txt";

    // Проверка данных пользователя
    if (CheckUserCredentials(filePath, fullName, phoneNumber, password))
    {
        MessageBox.Show("Вы авторизованы!"); // Уведомление о успешной авторизации
    }
    else
    {
        MessageBox.Show("Пользователь не найден. Проверьте введенные данные."); // Уведомление об ошибке
    }
}

private bool CheckUserCredentials(string filePath, string fullName, string phoneNumber, string password)
{
    try
    {
        string[] lines = System.IO.File.ReadAllLines(filePath);
        string userName = null, userPhone = null, userPassword = null;

        for (int i = 0; i < lines.Length; i++)
        {
            string line = lines[i].Trim();
            if (line.StartsWith("Пользователь:"))
            {
                userName = line.Substring("Пользователь:".Length).Trim();
            }
            else if (line.StartsWith("Телефон:"))
            {
                userPhone = line.Substring("Телефон:".Length).Trim();
            }
            else if (line.StartsWith("Пароль:"))
            {
                userPassword = line.Substring("Пароль:".Length).Trim();
            }

            // Если найдены все необходимые данные, прерываем цикл
            if (userName != null && userPhone != null && userPassword != null)
                break;
        }

        // Проверяем, что все данные были найдены
    }
}

```

```

if (userName != null && userPhone != null && userPassword != null)
{
    // Проверяем, что введенные данные совпадают с данными в файле
    if (userName.Equals(fullName, StringComparison.OrdinalIgnoreCase) &&
        userPhone.Equals(phoneNumber) &&
        userPassword.Equals(password))
    {
        return true; // Данные совпадают
    }
}
catch (Exception ex)
{
    MessageBox.Show($"Ошибка при чтении файла: {ex.Message}");
}
return false; // Не найдено совпадений
}

```

```

private void button9_Click(object sender, EventArgs e)
{
    string filePath = @"F:\База данных\Регистрация\user_data.txt";
    string enteredName = textBox4.Text.Trim();
    string enteredPhone = textBox1.Text.Trim();

    // Проверка данных в файле
    if (CheckUserData(filePath, enteredName, enteredPhone))
    {
        // Генерация кода
        CodeGenerator codeGenerator = new CodeGenerator();
        generatedCode = codeGenerator.GenerateCode();

        // Отображение кода пользователю
        MessageBox.Show($"Код подтверждения: {generatedCode}");

        // Активация поля для ввода кода
        textBox3.Enabled = true;
        button7.Enabled = true;
    }
    else
    {
        MessageBox.Show("Пользователь не найден.");
    }
}

private bool CheckUserData(string filePath, string name, string phone)
{
    bool userFound = false;

    try
    {
        using (StreamReader reader = new StreamReader(filePath))
        {
            string line;
            while ((line = reader.ReadLine()) != null)
            {
                Console.WriteLine("Считанная строка: " + line);
            }
        }
    }
}

```

```

    if (line.StartsWith("Пользователь: "))
    {
        string enteredNameFromFile = line.Substring("Пользователь: ".Length).Trim();
        Console.WriteLine("ФИО из файла: " + enteredNameFromFile);

        line = reader.ReadLine();
        if (line.StartsWith("Телефон: "))
        {
            string enteredPhoneFromFile = line.Substring("Телефон: ".Length).Trim();
            Console.WriteLine("Номер телефона из файла: " + enteredPhoneFromFile);

            if (enteredNameFromFile == name && enteredPhoneFromFile == phone)
            {
                userFound = true;
                break;
            }
        }
    }
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка при чтении файла: " + ex.Message);
}

return userFound;
}

private void button1_Click(object sender, EventArgs e)
{
    this.Hide();

    Form2 form2 = new Form2();
    form2.Show();
}

private void button8_Click(object sender, EventArgs e)
{
    this.Close();

    // Завершаем приложение
    Application.Exit();
}

private void button10_Click(object sender, EventArgs e)
{
    this.Hide();

    Form4 form4 = new Form4();
    form4.Show();
}

private void button3_Click(object sender, EventArgs e)
{
    this.Hide();
}

```

```

        Form6 form6 = new Form6();
        form6.Show();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        this.Hide();

        Form9 form9 = new Form9();
        form9.Show();
    }

    private void button4_Click(object sender, EventArgs e)
    {
        this.Hide();

        Form6 form6 = new Form6();
        form6.Show();
    }

    private void button7_Click(object sender, EventArgs e)
    {
        string enteredCode = textBox3.Text.Trim();

        if (enteredCode == generatedCode)
        {
            MessageBox.Show("Вы авторизованы!");
            // ... (здесь можно добавить действия, которые нужно выполнить после успешной авторизации)
        }
        else
        {
            MessageBox.Show("Неверный код.");
        }
    }

    private void textBox4_TextChanged(object sender, EventArgs e)
    {
    }
}

```

## Исходный код Form 8:

```

using System;
using System.Windows.Forms;

namespace Курсовая_работа_Машенцева
{
    public partial class Form8 : Form
    {
        private Ticket ticket;
        private CodeGenerator codeGenerator = new CodeGenerator();

        public Form8()
        {
            InitializeComponent();
            LoadUserData();
        }
    }
}

```

```

}

public string Label4Text { get => label4.Text; set => label4.Text = value; }

private void LoadUserData()
{
    string filePath = @"F:\База данных\Найденные рейсы\gotovyreys_data.txt";
    string userDataPath = @"F:\База данных\Регистрация\user_data.txt";

    ticket = new Ticket(filePath, userDataPath);

    // Заполнение TextBox данными из класса Ticket
    textBox14.Text = ticket.NomEr;
    textBox13.Text = ticket.ModelSamolyota;
    textBox12.Text = ticket.Klass;
    textBox11.Text = ticket.DepartureDate.ToString("dd MMMM yyyy г");
    textBox10.Text = ticket.Marshrut;
    textBox6.Text = ticket.Airport;
    textBox7.Text = ticket.Duration;
    textBox8.Text = ticket.TravelTime;
    label4.Text = ticket.Seat;
    textBox9.Text = ticket.Tsena.ToString();
    textBox17.Text = ticket.FullName;
    textBox18.Text = ticket.PassportData;
    textBox4.Text = ticket.PhoneNumber;
}

private void button1_Click(object sender, EventArgs e)
{
    this.Hide();

    Form2 form2 = new Form2();
    form2.Show();
}

private void button8_Click(object sender, EventArgs e)
{
    this.Close();

    // Завершаем приложение
    Application.Exit();
}

private void button2_Click(object sender, EventArgs e)
{
    this.Hide();

    Form4 form4 = new Form4();
    form4.Show();
}

private void button3_Click(object sender, EventArgs e)
{
    this.Hide();

    Form6 form6 = new Form6();
    form6.Show();
}

private void button9_Click(object sender, EventArgs e)
{
    // Проверка введенных данных
    if (ticket.ValidateInput(textBox1, textBox2, textBox3))
    {
        ticket.Code = codeGenerator.GenerateCode(); // Генерация кода
        MessageBox.Show($"Ваш код: {ticket.Code}"); // Уведомление с кодом
    }
}

```

```

        }
    }

    private void button7_Click(object sender, EventArgs e)
    {
        // Проверка ввода кода
        if (textBox5.Text == ticket.Code)
        {
            MessageBox.Show("Оплата прошла успешно. Сервис SkyFly советует Вам сделать скриншот билета," +
                " чтобы ускорить регистрацию на рейс в аэропорту. Желаем Вам приятного полета!");

        }
        else
        {
            MessageBox.Show("Неверный код. Попробуйте снова.");
        }
    }
}
}

```

### Исходный код Form 9:

```

using System;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace Курсовая_работа_Машенцева
{
    public partial class Form9 : Form
    {
        private string filePath = @"F:\База данных\Регистрация\user_data.txt";
        public string FullName { get; set; }
        public string PassportData { get; set; }
        public string PhoneNumber { get; set; }
        public string Password { get; set; }

        public Form9()
        {
            InitializeComponent();
            LoadUserData();
        }

        // Метод для загрузки данных из файла
        private void LoadUserData()
        {
            try
            {
                // Чтение всех строк из файла
                string[] lines = System.IO.File.ReadAllLines(filePath);

                if (lines.Length >= 4)
                {
                    // Извлечение данных из строк
                    string fullName = lines[0].Split(new[] { ';' }, 2)[1].Trim();
                    string passportData = lines[1].Split(new[] { ';' }, 2)[1].Trim();
                    string phoneNumber = lines[2].Split(new[] { ';' }, 2)[1].Trim();
                    string password = lines[3].Split(new[] { ';' }, 2)[1].Trim();

                    // Заполнение текстовых полей
                }
            }
        }
    }
}

```

```

        textBox4.Text = fullName;
        textBox1.Text = passportData;
        textBox2.Text = phoneNumber;
        textBox3.Text = password;
    }
    else
    {
        MessageBox.Show("Недостаточно данных в файле.");
    }
}
catch (Exception ex)
{
    MessageBox.Show("Произошла ошибка при чтении файла: " + ex.Message);
}
}

public bool ValidateInput()
{
    if (!Regex.IsMatch(textBox4.Text, @"^[A-Za-zA-Яа-яёЁ\s]+$"))
    {
        MessageBox.Show("Данные не введены или введены неверно для ФИО.");
        return false;
    }

    if (!Regex.IsMatch(textBox1.Text, @"^\d{4} \d{6}$"))
    {
        MessageBox.Show("Данные введены некорректно для паспортных данных.");
        return false;
    }

    if (!Regex.IsMatch(textBox2.Text, @"^8\d{10}$"))
    {
        MessageBox.Show("Данные введены некорректно для номера телефона.");
        return false;
    }

    if (textBox3.Text.Length < 6)
    {
        MessageBox.Show("Пароль должен содержать минимум 6 символов.");
        return false;
    }

    // Сохранение данных в свойства
    this.FullName = textBox1.Text;
    this.PhoneNumber = textBox3.Text;
    this.PassportData = textBox2.Text;
    this.Password = textBox4.Text;

    return true;
}

private void button6_Click(object sender, EventArgs e)
{
    if (ValidateInput())
    {

```

```

        UpdateUserData();
    }
}

private void UpdateUserData()
{
    try
    {
        // Подготовка данных для сохранения
        string updatedData =
            $"Пользователь: {textBox4.Text}\n" +
            $"Паспортные данные: {textBox1.Text}\n" +
            $"Телефон: {textBox2.Text}\n" +
            $"Пароль: {textBox3.Text}";

        // Запись обновленных данных в файл
        System.IO.File.WriteAllText(filePath, updatedData);

        MessageBox.Show("Данные успешно обновлены.");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Произошла ошибка при сохранении файла: " + ex.Message);
    }
}

private void button2_Click(object sender, EventArgs e)
{
    textBox4.Text = string.Empty;
    textBox2.Text = string.Empty;
    textBox3.Text = string.Empty;
    textBox1.Text = string.Empty;
}

private void button1_Click(object sender, EventArgs e)
{
    this.Hide();

    Form2 form2 = new Form2();
    form2.Show();
}

private void button8_Click(object sender, EventArgs e)
{
    this.Close();

    // Завершаем приложение
    Application.Exit();
}

private void button3_Click(object sender, EventArgs e)
{
    this.Hide();

    Form6 form6 = new Form6();
    form6.Show();
}

```

```
}
```

## Исходный код класса CodeGenerator.cs

```
using System;

namespace Курсовая_работа_Машенцева
{
    public class CodeGenerator
    {
        private Random random = new Random();

        public string GenerateCode()
        {
            return random.Next(100, 1000).ToString();
        }
    }
}
```

## Исходный код класса Reys.cs

```
using System;
using System.Globalization;

namespace Курсовая_работа_Машенцева
{
    public class Reys
    {
        public string NomEr { get; set; } // Номер рейса
        public string Otkuda { get; set; } // Откуда
        public string Kuda { get; set; } // Куда
        public string Klass { get; set; } // Класс
        public DateTime DataOtpravleniya { get; set; } // Дата отправления
        public TimeSpan VremyaVyleta { get; set; } // Время вылета

        public TimeSpan VremyaPrileta { get; set; } // Время прилета

        public TimeSpan VremyaVPute { get; set; } // Время в пути

        public string ModelSamolyota { get; set; } // Модель самолёта
        public int Tsena { get; set; } // Цена
        public string AeroportOtkuda { get; set; } // Аэропорт отправления
        public string AeroportKuda { get; set; } // Аэропорт прибытия

        private static Random random = new Random();
        // Генерирует номер рейса

        private string GenerateNomEr1()
        {
            Random random = new Random();
            char letter1 = (char)random.Next('A', 'Z' + 1);
            char letter2 = (char)random.Next('A', 'Z' + 1);
            int number = random.Next(1000, 9999);
            return $"№{letter1}{letter2}{number}";
        }

        public Reys(string filePath)
        {
            var lines = System.IO.File.ReadAllLines(filePath);
```

```

Otkuda = lines[0].Split(':')[1].Trim();
Kuda = lines[1].Split(':')[1].Trim();
string dateString = lines[2].Split(':')[1].Trim().Replace(" ", "");
DataOtpravleniya = DateTime.ParseExact(dateString, "dd.MM.yyyy", CultureInfo.InvariantCulture.InvariantCulture);
Klass = lines[3].Split(':')[1].Trim();

NomEr = GenerateNomEr1(); // Генерация номера рейса

Random rand = new Random();
VremyaVyleta = TimeSpan.FromHours(rand.Next(24));
VremyaPrileta = VremyaVyleta.Add(TimeSpan.FromMinutes(rand.Next(60, 120))); // Время в пути от 1 до 2
часов
VremyaVPute = VremyaPrileta - VremyaVyleta;

ModelSamolyota = "AIRBUS777";
Tsena = rand.Next(5000, 10001); // Генерация цены от 5000 до 10000

// Определяем аэропорты исходя из городов
AeroportOtkuda = GetAeroport(Otkuda);
AeroportKuda = GetAeroport(Kuda);
}

// Простой механизм для получения аэропорта
private string GetAeroport(string city)
{
    switch (city)
    {
        case "Москва":
            return "Внуково";
        case "Владивосток":
            return "Кневичи";
        case "Санкт-Петербург":
            return "Пулково";
        case "Анапа":
            return "Витязево";
        case "Краснодар":
            return "Пашковский";
        case "Мурманск":
            return "Мурманск";
        case "Омск":
            return "Омск";
        case "Пермь":
            return "Пермь";
        case "Томск":
            return "Богашево";
        case "Воронеж":
            return "Чертовицкое";
        default:
            return "Неизвестный аэропорт";
    }
}
}

```

## Исходный код класса Ticket.cs

using System;

```

using System.IO;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace Курсовая_работа_Машенцева
{
    public class Ticket
    {
        public string NomEr { get; set; }
        public string Marshrut { get; set; }
        public DateTime DepartureDate { get; set; }
        public string ModelSamolyota { get; set; }
        public string Klass { get; set; }
        public string Airport { get; set; }
        public string Duration { get; set; }
        public string TravelTime { get; set; }
        public decimal Tsena { get; set; }
        public string Seat { get; set; }
        public string FullName { get; set; }
        public string PhoneNumber { get; set; }
        public string PassportData { get; set; }
        public string NumberCard { get; set; }
        public string SrokCard { get; set; }
        public string CVC { get; set; }
        public string Code { get; set; }

        // Конструктор класса, где добавлен путь к файлу
        public Ticket(string filePath, string userDataPath)
        {
            ReadFromFile(filePath);

            LoadUserData(userDataPath);
        }

        private void ReadFromFile(string filePath)
        {
            foreach (var line in System.IO.File.ReadAllLines(filePath))
            {
                if (line.StartsWith("Номер рейса: "))
                {
                    NomEr = line.Substring("Номер рейса: ".Length).Trim();
                }
                else if (line.StartsWith("Маршрут: "))
                {
                    Marshrut = line.Substring("Маршрут: ".Length).Trim();
                }
                else if (line.StartsWith("Дата рейса: "))
                {
                    DepartureDate = DateTime.Parse(line.Substring("Дата рейса: ".Length).Trim());
                }
                else if (line.StartsWith("Самолёт: "))
                {
                    ModelSamolyota = line.Substring("Самолёт: ".Length).Trim();
                }
                else if (line.StartsWith("Тип класса: "))
                {
                    Klass = line.Substring("Тип класса: ".Length).Trim();
                }
            }
        }
    }
}

```

```

        }
        else if (line.StartsWith("Аэропорт: "))
        {
            Airport = line.Substring("Аэропорт: ".Length).Trim();
        }
        else if (line.StartsWith("Продолжительность рейса: "))
        {
            Duration = line.Substring("Продолжительность рейса: ".Length).Trim();
        }
        else if (line.StartsWith("Время в пути: "))
        {
            TravelTime = line.Substring("Время в пути: ".Length).Trim();
        }
        else if (line.StartsWith("Цена: "))
        {
            Tsena = int.Parse(line.Substring("Цена: ".Length).Trim());
        }
        ModelSamolyota = "AIRBUS777";
    }

}

private void LoadUserData(string path)
{
    // Чтение пользовательских данных из файла
    var lines = File.ReadAllLines(path);
    if (lines.Length > 0)
    {
        FullName = lines[0].Split(new[] { ':' }, 2)[1].Trim(); // ФИО
        PassportData = lines[1].Split(new[] { ':' }, 2)[1].Trim(); // Паспортные данные
        PhoneNumber = lines[2].Split(new[] { ':' }, 2)[1].Trim(); // Номер телефона
    }
}

public bool ValidateInput(TextBox textBox1, TextBox textBox2, TextBox textBox3)
{
    if (!Regex.IsMatch(textBox1.Text, @"^(\d{4})\s(\d{4})\s(\d{4})\s(\d{4})$"))
    {
        MessageBox.Show("Данные карты введены неверно. Введите данные в формате **** * * * *");
        return false;
    }

    if (!Regex.IsMatch(textBox2.Text, @"^(1[0-2]|1[9])/(3[01]|1[9][0-9])$"))
    {
        MessageBox.Show("Данные срока действия карты введены неверно. Введите данные в формате **/**.");
        return false;
    }

    if (textBox3.Text.Length > 3)
    {
        MessageBox.Show("CVC код введен неверно. Введите код в формате ***.");
        return false;
    }
    if (textBox3.Text.Length < 3)
    {
        MessageBox.Show("CVC код введен неверно. Введите код в формате ***.");
        return false;
    }
}

```

```

        }
        else if (textBox3.Text.Contains(" "))
        {
            MessageBox.Show("CVC код не должен содержать пробелы.");
            return false;
        }

        // Сохранение данных в свойства
        this.NumberCard = textBox1.Text;
        this.SrokCard = textBox2.Text;
        this.CVC = textBox3.Text;
        return true;
    }
}
}
}

```

### Исходный код класса UserRegistration.cs

```

using System;
using System.IO;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace Курсовая_работа_Машенцева
{
    public class UserRegistration
    {
        public string FullName { get; set; }
        public string PassportData { get; set; }
        public string PhoneNumber { get; set; }
        public string Password { get; set; }
        public string Code { get; set; }

        public UserRegistration(string fullName, string passportData, string phoneNumber,
            string password, string code)
        {
            FullName = fullName;
            PassportData = passportData;
            PhoneNumber = phoneNumber;
            Password = password;
            Code = code;
        }

        public UserRegistration()
        {
        }

        public bool ValidateInput(TextBox textBox1, TextBox textBox2, TextBox textBox3, TextBox textBox4, TextBox
textBox5)
        {
            if (!Regex.IsMatch(textBox1.Text, @"^[\u0410-\u043a-\u043a-\u043f\u0435\u0447\u0435\u043d\u0438\u0435]+$"))
            {
                MessageBox.Show("Данные не введены или введены неверно для ФИО.");
                return false;
            }

            if (!Regex.IsMatch(textBox2.Text, @"^\d{4} \d{6}$"))

```

```
        }

        MessageBox.Show("Данные введены некорректно для паспортных данных.");
        return false;
    }

    if (!Regex.IsMatch(textBox3.Text, @"^8\d{10}$"))
    {
        MessageBox.Show("Данные введены некорректно для номера телефона.");
        return false;
    }

    if (textBox4.Text.Length < 6)
    {
        MessageBox.Show("Пароль должен содержать минимум 6 символов.");
        return false;
    }

}

// Сохранение данных в свойства
this.FullName = textBox1.Text;
this.PassportData = textBox2.Text;
this.PhoneNumber = textBox3.Text;
this.Password = textBox4.Text;

return true;
}

// Метод для сохранения данных в файл

public void SaveToFile(string fileName)
{
    string path = @"F:\База данных\Регистрация\" + fileName; // Путь до файла
    using (StreamWriter writer = new StreamWriter(path, true))
    {
        writer.WriteLine($"Пользователь: {FullName}");
        writer.WriteLine($"Паспортные данные: {PassportData}");
        writer.WriteLine($"Телефон: {PhoneNumber}");
        writer.WriteLine($"Пароль: {Password}");
        writer.WriteLine("-----");
    }
}
```

**ПРИЛОЖЕНИЕ Б**  
**Блок-схемы алгоритмов**

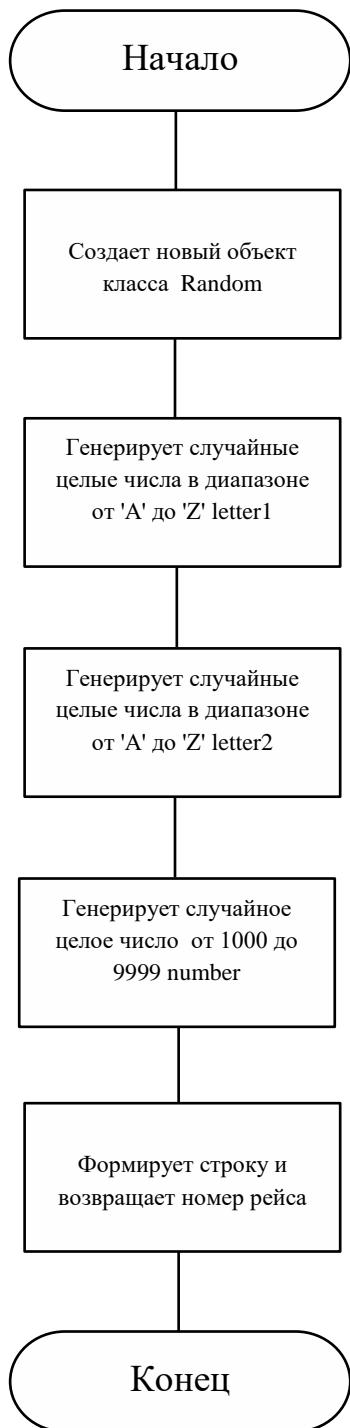


Рисунок Б.1 – Блок-схема алгоритма функции GenerateNomEr1() (Reys).

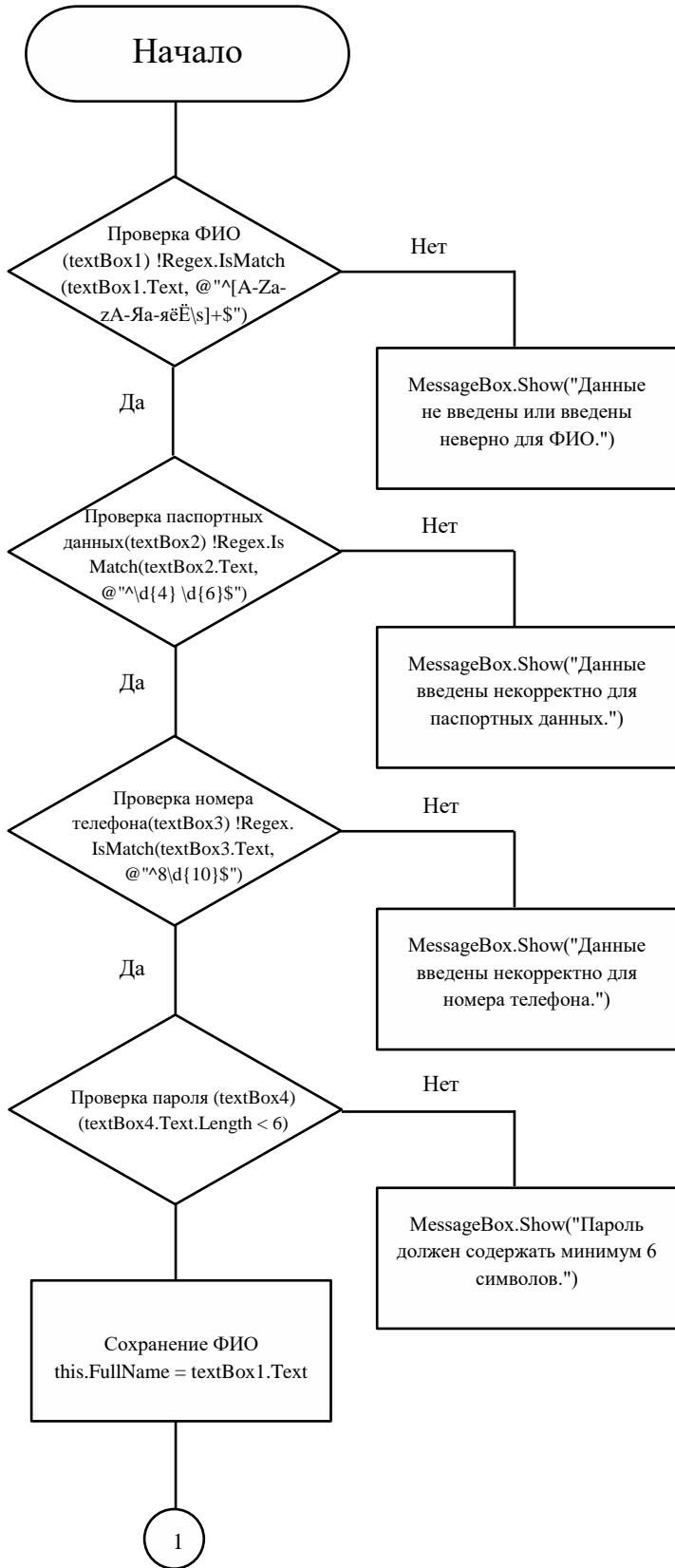




Рисунок Б.2 – Блок-схема алгоритма функции `ValidateInput()`  
(`UserRegistration.cs`).