МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01 Информационные системы и технологии

Направление специальности 1-40 01 02 03 Информационные системы и технологии (издательско-полиграфический комплекс)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОЙ РАБОТЫ**

по дисциплине «Объектно-ориентированное программирование» Тема Программное средство «Бронирование и поиск авиарейсов»

Исполнитель

студент 2 курса группы 1 Дырда Дмитрий Геннадьевич

(Ф.И.О)

Руководитель работы ст. препод. И.Г. Сухорукова

(учен. степень, звание, должность, подпись, Ф.И.О)

Курсовой проект защищен с оценкой

Председатель Н.В. Пацей

(подпись)

Минск 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий  
Кафедра информационных систем и технологий

Утверждаю

Заведующий кафедрой ПИ

Н.В Пацей

подпись инициалы и фамилия

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Объектно-ориентированное программирование"

|  |  |
| --- | --- |
| Специальность: 1-40 01 05 03 Информационные системы и технологии  Студент: Дырда Д.Г. | Группа: 1 |
| **Тема: Программное средство «Бронирование и поиск авиарейсов»** | |

**1. Срок сдачи студентом законченной работы**: "май 2019 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС должно:

* взаимодействовать с базой данных (хранить информацию о рейсах, о маршрутах, о пользователях);
* производить поиск авиарейса;
* производить бронирование рейса;
* поддерживать системы регистрации и аутентификации;
* хранить историю о заказанных пользователем рейсах;
* поддерживать возможность добавления администратором информации в базу данных (маршрут, рейс);
* поддерживать бонусную систему (предоставление пользователям скидок).

**2.2.** Программное средство должно быть выполнено с использованием ООП, взаимодействовать с базой данных, реализовано под ОС Windows. Отображение, бизнес логика и хранилище данных должны быть максимально независимы друг от друга для возможности расширения. Язык разработки С#. Управление программой должно быть интуитивно понятным и удобным. Использовать архитектурные и шаблоны проектирования.

**3. Примерное содержание расчетно-пояснительной записки**

(перечень вопросов, подлежащих разработке)

* Введение.
* Постановка задачи.
* Обзор литературы и аналогов (алгоритмы решения, обзор литературы, прототипы).
* Проектирование программного средства (концепция, модель, структура, выполняемые функции и взаимосвязь всех компонентов, диаграммы UML).
* Разработка модели данных программного средства.
* Руководство пользователя.
* Тестирование программного средства.
* Заключение.
* Список используемых источников
* Приложения

**4. Форма представления выполненной курсовой работы:**

* + Теоретическая часть курсового проекта должны быть представлены в формате MS Word. Оформление записки должно быть согласно правилам.
  + Необходимые схемы, диаграммы и рисунки допускается делать в MS Office Visio, VS или других редакторах.
  + Листинги программы представляются частично в приложении.
  + К записке необходимо приложить CD (DVD), который должен содержать: пояснительную записку, листинги и инсталляцию проекта.

#### Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2019 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2019 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2019 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 2.04.2019 |  |
| 5 | Кодирование программного средства | 23.04.2019 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2019 |  |
| 7 | Оформление пояснительной записки | 7.05.2019 |  |
| 8 | Сдача проекта | 20.05.2019 |  |

**5. Дата выдачи задания** 07.02.2019

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.Г. Сухорукова

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.Г. Дырда

(дата и подпись студента)

**Оглавление**

[**ВВЕДЕНИЕ** 7](#_Toc10079653)

[**1.** **Обзор аналогов** 8](#_Toc10079654)

[**2.** **Анализ требований к программному средству и разработка функциональных требований** 13](#_Toc10079655)

[**2.1 Функциональные требования** 13](#_Toc10079656)

[**2.2 Варианты использования** 13](#_Toc10079657)

[**2.3 Спецификация функциональных требований** 14](#_Toc10079658)

[**3.** **Проектирование программного средства** 15](#_Toc10079659)

[**3.1 Концепция** 15](#_Toc10079660)

[**3.2 Модель базы данных** 15](#_Toc10079661)

[**3.3 Структура проекта** 18](#_Toc10079662)

[**4.** **Реализация программного средства** 23](#_Toc10079663)

[**4.1 Реализация Data Access Layer** 23](#_Toc10079664)

[**4.2 Реализация Business Logic Layer** 23](#_Toc10079665)

[**4.3 Реализация Presentation Layer** 24](#_Toc10079666)

[**5.** **Тестирование, проверка работоспособности и анализ полученных результатов** 25](#_Toc10079667)

[**6.** **Руководство пользователя по использованию** 29](#_Toc10079668)

[**ЗАКЛЮЧЕНИЕ** 30](#_Toc10079669)

[**СПИСОК ЛИТЕРАТУРЫ** 31](#_Toc10079670)

[**ПРИЛОЖЕНИЕ А** 32](#_Toc10079671)

[**ПРИЛОЖЕНИЕ Б** 33](#_Toc10079672)

**ВВЕДЕНИЕ**

Раньше для того, чтобы заказать авиарейс приходилось проводить много времени в очередях и в аэропорту в целом. Также многие не могли попасть на нужный рейс по причине того, что просто опоздали.

Поэтому для того чтобы избавить заказчиков от ненужных хлопот и сохранить их время разрабатываются приложения, которые ускоряют процесс заказа рейса, а также делают его удобным.

Данные приложения позволяют пользователям с легкостью заказать билет в любую точку мира и при этом они потратят гораздо меньше времени, чем если бы они производили заказ в аэропорту.

Целью данного курсового проекта является разработка ПО, осуществляющего поиск и бронирование авиарейсов, регистрацию и аутентификацию пользователей. Также необходимо реализовать сохранение пользовательских рейсов, возможность добавления рейсов и маршрутов администратором.

1. **Обзор аналогов**

В качестве первого аналога было выбрано программное средство «AnyWayAnyDay».

В главном меню пользователь может осуществить поиск необходимого ему рейса, который представлен на рисунке 1.1.

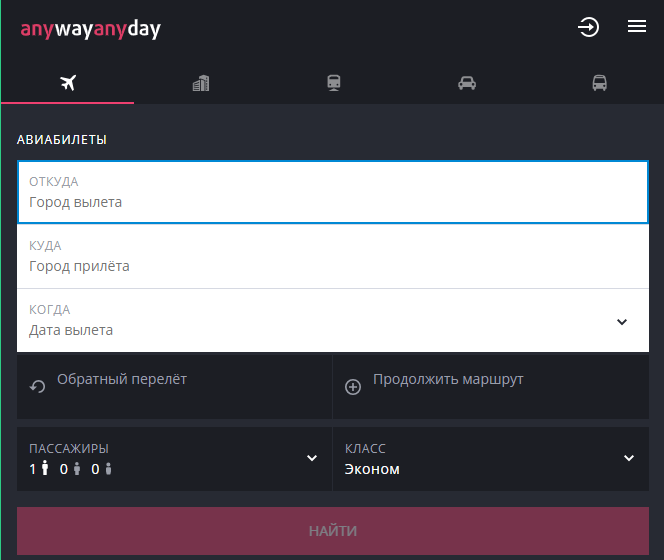


Рисунок 1.1 – Поиск авиарейса «AnyWayAnyDay»

Также на рисунке 1.2 представлена возможность программного средства хранить историю заказов пользователя.

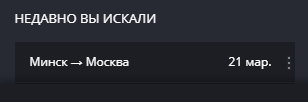


Рисунок 1.2 – Хранение истории «AnyWayAnyDay».

Программное средство поддерживает системы регистрации и аутентификации. Это продемонстрировано на рисунке 1.3.

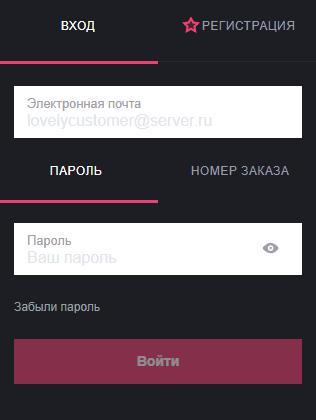


Рисунок 1.3 – Система регистрации и аутентификации.

На рисунке 1.4 продемонстрирована возможность пользователям заказывать билеты на рейс.

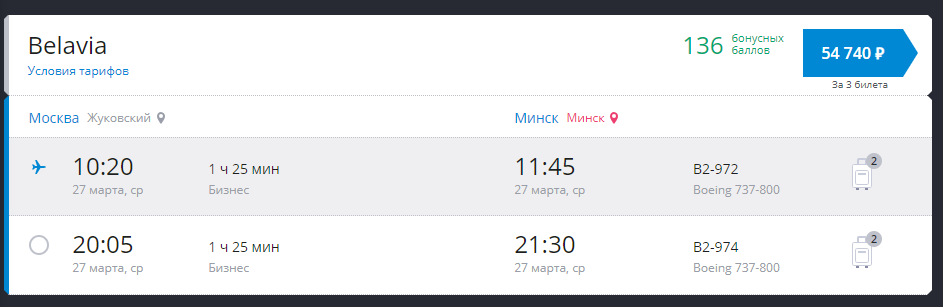


Рисунок 1.4 – Заказ билета

В качестве второго аналога было выбрано приложение «Tickets.ua».

Дизайн приложения простой и понятный. Все, что нужно для поиска, - это указать пункты вылета и прилета, даты отправки и приезда и количество пассажиров. В течение нескольких секунд приложение выдаст рейсы, соответсвующие запросу. Чтобы не запутаться в разнообразии, пользователю предоставлена возможность сортировки по времени перелета и стоимости. На рисунке 1.5 представлены возможности приложения «Tickets.ua».

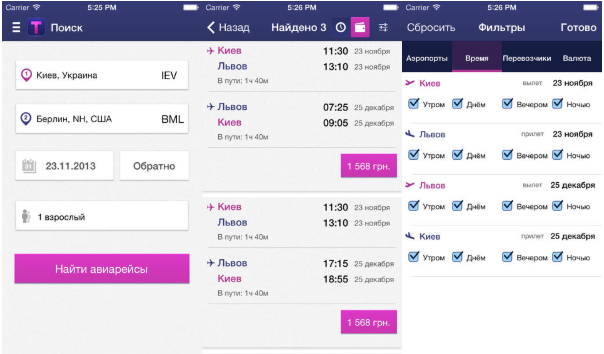


Рисунок 1.5 – Функционал приложения «Tickets.ua».

В качестве третьего аналога выступает программное средство «Biletix».

В приложении возможны поиск и фильтрация авиарейсов, представленные на рисунке 1.6.



Рисунок 1.6 – Поиск и фильтрация авиарейсов «Biletix».

На рисунке 1.7 представлена система аутентификации приложения.

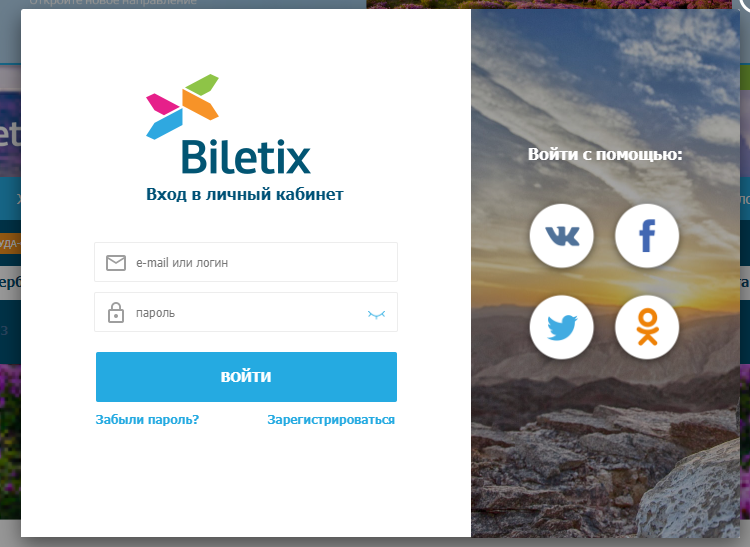


Рисунок 1.7 – Система аутентификации приложения «Biletix».

На рисунке 1.8 представлена возможность бронирования авиарейсов.

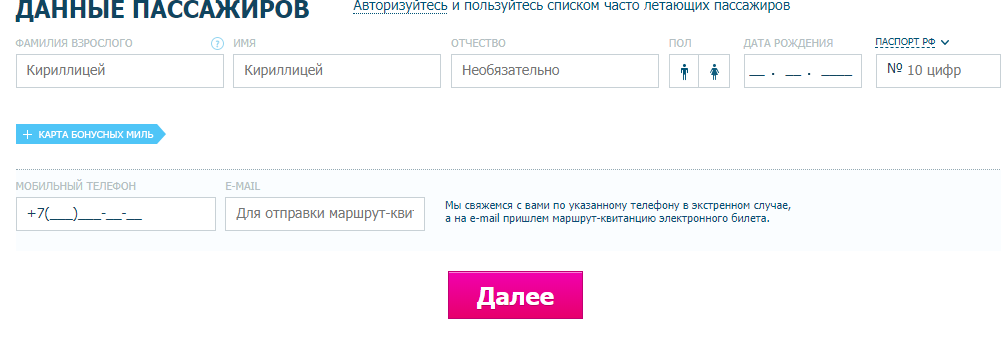


Рисунок 1.8 – Бронирование авиарейсов «Biletix».

Проанализировав аналоги, я решил использовать следующий функционал: возможность аутентификации и регистрации пользователей, поиск и бронирование авиарейсов пользователями, просмотр истории заказов, возможность добавления администратором новых маршрутов, рейсов.

1. **Анализ требований к программному средству и разработка функциональных требований**

**2.1 Функциональные требования**

Рассмотрев аналоги, были представлены следующие функциональные требования:

- взаимодействие с базой данных (хранение информации о рейсах, о маршрутах, о пользователях);

- произведение поиска авиарейса;

- произведение бронирования рейса;

- поддержка системы регистрации и аутентификации;

- хранение историю о заказанных пользователем рейсах;

- поддержка возможности добавления администратором информации в базу данных (маршрут, рейс);

**2.2 Варианты использования**

Далее представлена UML-диаграмма с функционалом пользователей представлена на рисунке 2.1.

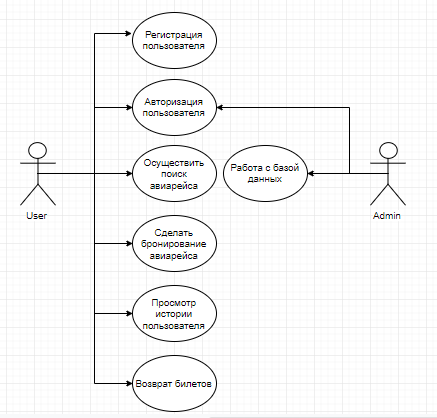


Рисунок 2.1 – Функционал пользователей

На рисунке 2.2 представлена диаграмма деятельности для входа в систему.

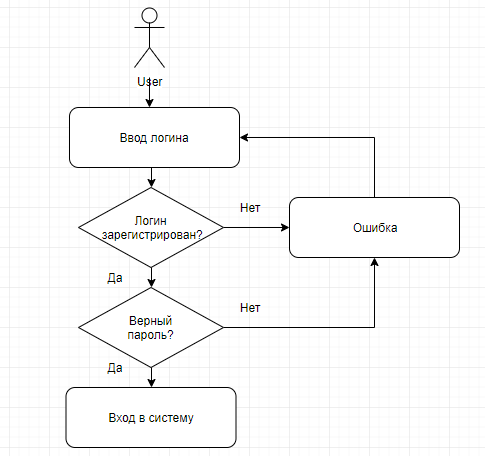


Рисунок 2.2 – Диаграмма деятельности для входа в систему

И последняя диаграмма, представленная на рисунке 2.3, – диаграмма последовательности для поиска авиарейса.

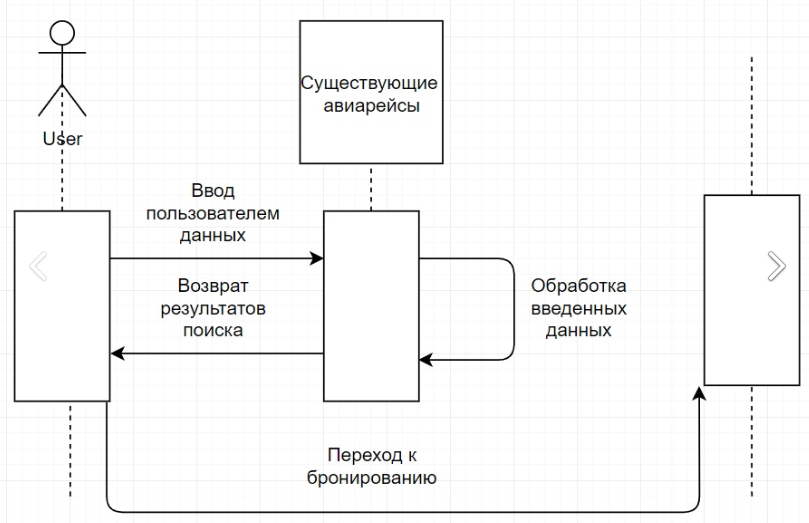


Рисунок 2.3 – Диаграмма последовательности для поиска авиарейса

**2.3 Спецификация функциональных требований**

Данное приложение реализовано на языке программирования C# с использованием WPF. Для дизайна использовал свою фантазию и все стили писал сам вручную на языке XAML.

Для работы с базой данных была взята СУБД Microsoft SQL Server и была связана с приложением при помощи технологии ADO.NET.

ADO.NET представляет собой технологию работы с данными, которая основана на платформе .NET Framework. Эта технология представляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

1. **Проектирование программного средства**

**3.1 Концепция**

В данном программном средстве была использована концепция классической трехуровневой архитектуры, представленная на рисунке 3.1.

Presentation layer (уровень представления): это тот уровень, с которым взаимодействует пользователь. Этот уровень включает компоненты пользовательского интерфейса, механизм получения ввода от пользователя.

Business layer (уровень бизнес-логики): содержит набор компонентов, которые отвечают за обработку полученных от уровня представлений данных, реализует всю необходимую логику приложения, все вычисления, взаимодействует с базой данных и передает уровню представления результат обработки.

Data Access layer (уровень доступа к данным): здесь размещаются специфичные классы для работы с разными технологиями доступа к данным.

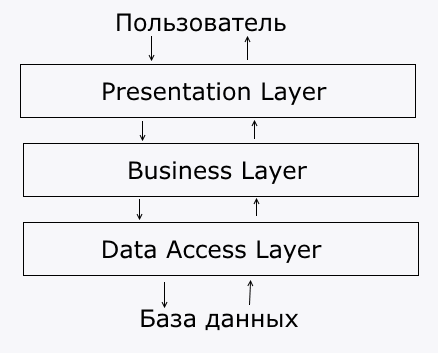
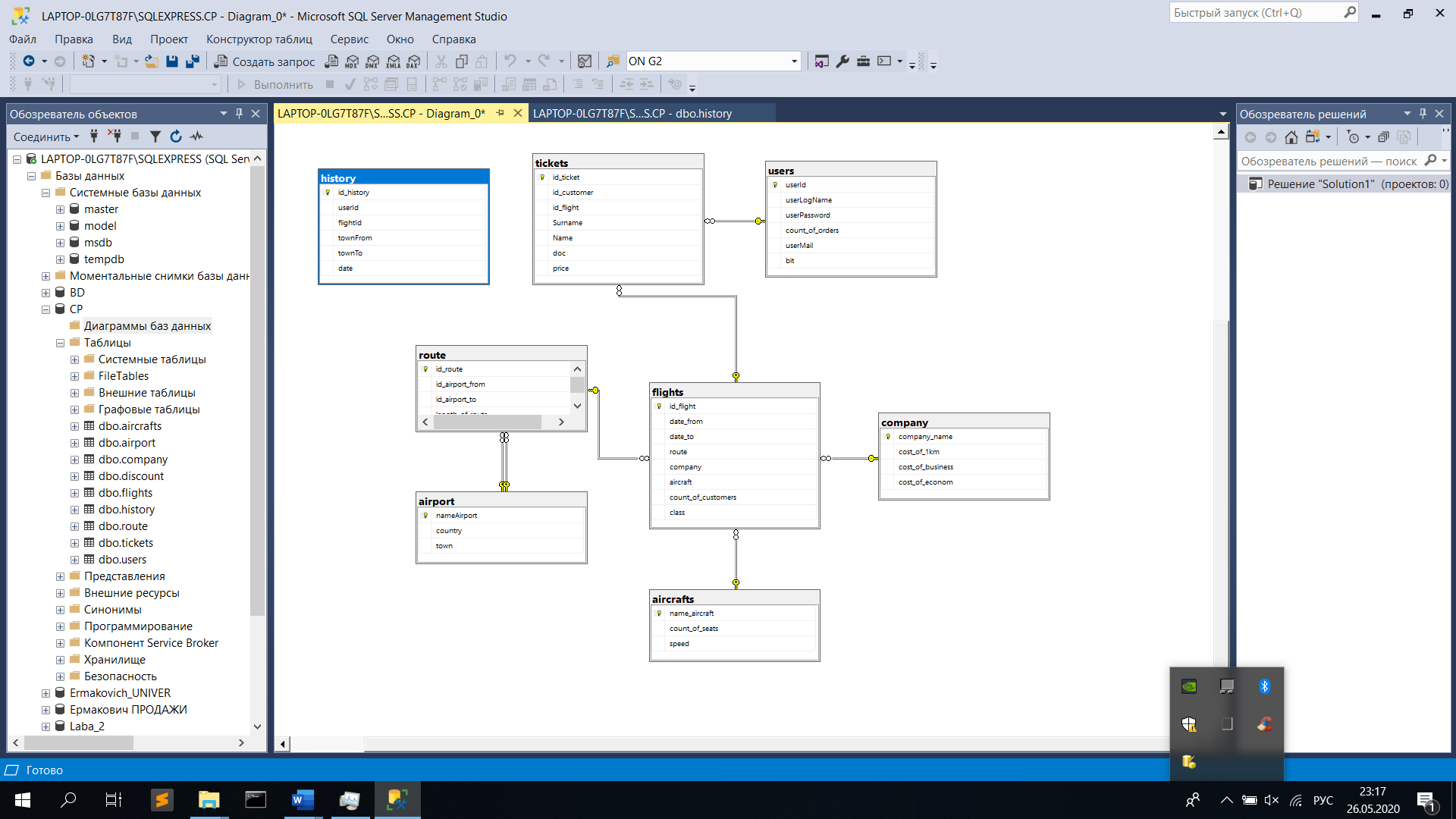


Рисунок 3.1 – Трехуровневая архитектура

**3.2 Модель базы данных**

Для реализации данного программного средства была создана база данных CP.db.

Структура базы данных представлена на рисунке 3.2.



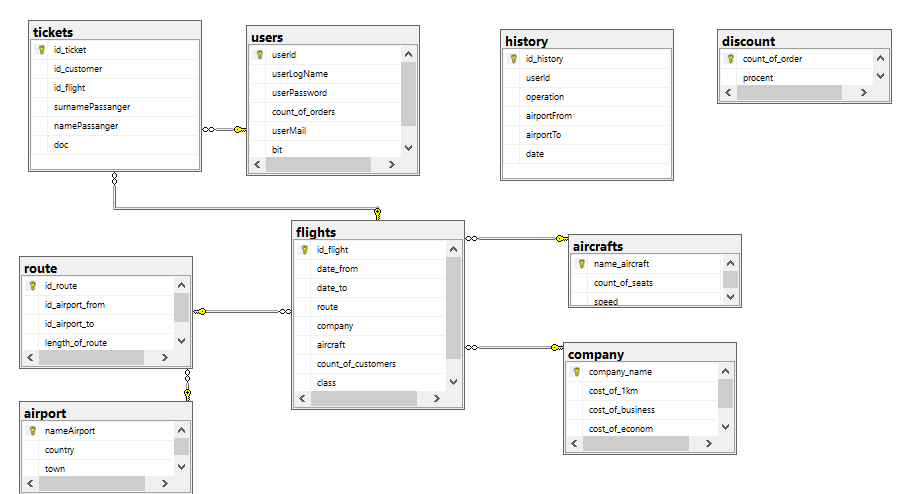


Рисунок 3.2 – Диаграмма базы данных.

Таблица «airport» хранит информацию об аэропортах, она представлена следующими полями:

* nameAirport (название аэропорта)
* country (страна)
* torn (город)

Таблица «route» хранит информацию о маршрутах, состоит их полей:

* id\_route (id маршрута)
* id\_airport\_from (место отправления)
* id\_airport\_to (место прибытия)
* length of\_route (длина маршрута)

Таблица «aircraft» хранит информацию о самолетах, представлена полями:

* name\_aircraft (название самолета)
* count\_of\_seats (количество мест)
* speed (средняя скорость самолета)

Таблица «company» хранит информацию об авиакомпаниях, она состоит из полей:

* company (название компании)
* cost\_of\_1km (стоимости за 1 км пути)
* cost\_of\_business (стоимость за бизнес класс)
* cost\_of\_econom (стоимость за эконом класс)

Таблица «flights» содержит информацию о рейсах, здесь имеются следующие поля:

* id\_flight (id рейса)
* date\_from (дата отправления)
* date\_to (дата прибытия)
* route (id маршрута)
* company (компания)
* aircraft (самолет)
* count\_of\_cutomers (количество заказов рейса)
* class (класс полета)

Таблица «tickets» хранит забронированные билеты на авиарейс, она состоит из полей:

* id\_ticket (id билета)
* id\_customer (id заказчика)
* id\_flight (id рейса)
* surnamePassanger (фамилия пассажира)
* namePassanger (имя пассажира)
* doc (номер пасспорта)

Таблица «users» содержит информацию о пользователе, она содержит следующие поля:

* userId (id пользователя)
* userLogName (пользовательское имя)
* userPassword (пароль пользователя)
* count\_of\_orders (количество бронирований, сделанных пользователем)
* userMail (mail пользователя)
* bit (поле, определяющее пользователя: обычный пользователь / администратор)

Таблица «history» хранит историю пользователей, состоит из:

* id\_history (id истории)
* userId (id пользователя)
* flightId (id рейса)
* townFrom (место отправления)
* townTo (место прибытия)
* date (дата вылета)

**3.3 Структура проекта**

Структура данного программного средства, представленная на рисунке 3.3, была разбита на 3 части.

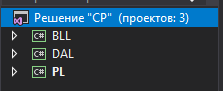


Рисунок 3.3 – Структура проекта

Структурное дерево проекта PL, представленное на рисунке 3.4, состоит из основного класса приложения App.xaml и нескольких папок:

- Frames, содержащая страницы приложения;

- Images, содержащая некоторые изображения, которые используются в данном ПС;

- ComboBox, содержащая класс для работы с выпадающим меню, которое используется в некоторых страницах приложения;

- Styles, содержащая стили для некоторых элементов в проекте;

- UserControls, содержащая пользовательский элемент управления;

- Windows, содержащая окна приложения;

- WorkWithPages, содержащая классы для работы со страницами BronPage и SearchPage.

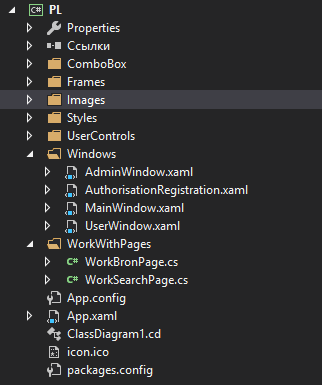


Рисунок 3.4 – Структура проекта PL

Далее на рисунке 3.5 приведена структура проекта BLL содержащая папки, в которых находятся классы, разрешающие работу слоя базы данных и слоя представления:

- Information содержит классы, в которых хранится промежуточная информация о пользователе, о поиске рейса и о самом рейсе. Данные классы реализуют паттерн Singleton;

- Mail содержит класс, который работает с электронной почтой;

- Validation содержит класс, проверяющие правильность вводимых пользователем данных, а также класс, отвечающий за хеширование пароля;

- WorkWithBdData, содержит классы, который связываются с уровнем DAL для получения данных из базы данных;

- WorkWithPriceTime, содержит классы, которые отвечают за обработку данных связанных со временем и ценой.

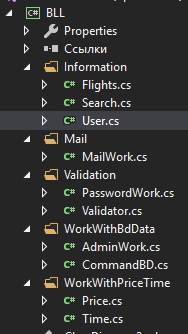


Рисунок 3.5 – Структура проекта BLL

Третий проект, представленный на рисунке 3.6, реализует работу с базой данных при помощи класса BdWork, который содержит все необходимые методы для работы с базой данных:

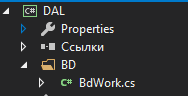


Рисунок 3.6 – Структура проекта DAL

Далее будут приведены диаграммы классов данных проектов:

* BLL, представлена на рисунке 3.7

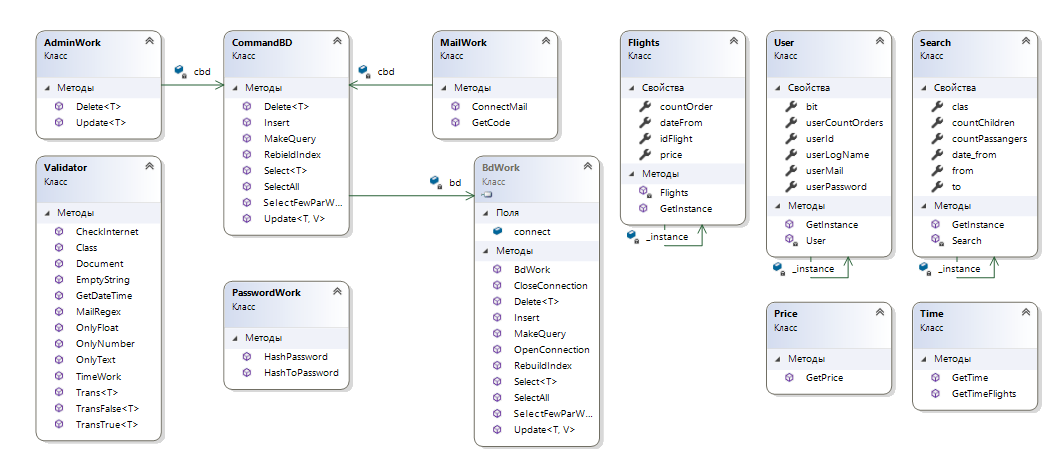


Рисунок 3.7 – Business Logic Layer

* DAL, представлена на рисунке 3.8

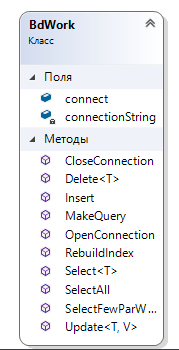


Рисунок 3.8 – Data Access Layer

* PL, представлена на рисунке 3.9

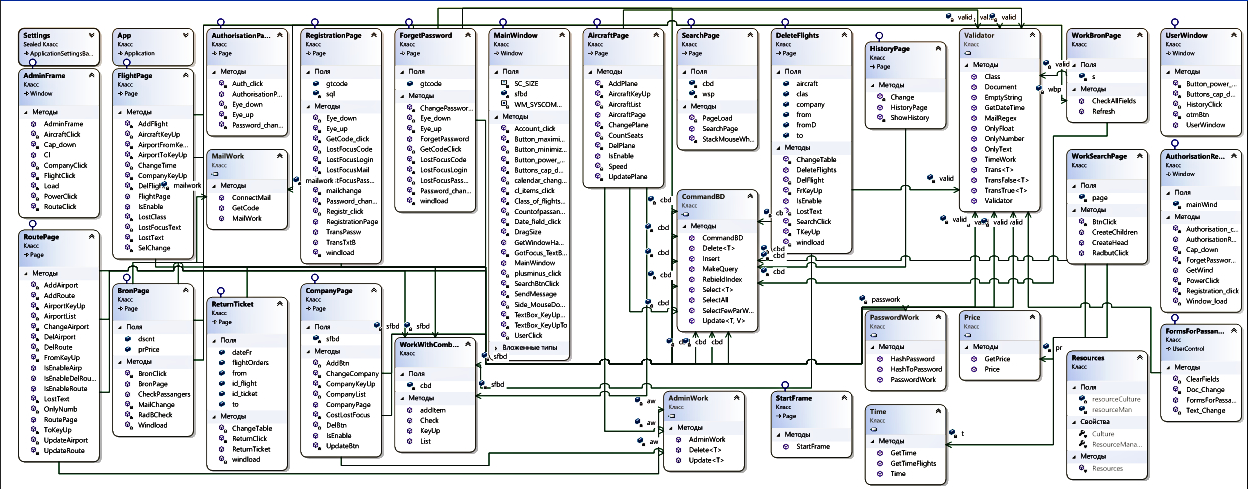


Рисунок 3.9 - Presentation Layer

1. **Реализация программного средства**

**4.1 Реализация Data Access Layer**

Данный уровень (уровень доступа к данным) хранит класс, позволяющий получать объекты из базы данных.

Принципы, применяющиеся при проектировании DAL:

* не ставится задача автоматического, безостановочного, не требующего затрат или мгновенного переключения работающей АС с одного средства хранения на другое. Проектировочные решения должны не полностью исключать (что долго и сложно), а существенно снижать зависимость логики приложения от средства хранения таким образом, чтобы значительно облегчить возможный переход. Нужно рассчитывать на разумный подход проектировщиков АС и понимание ими архитектурных принципов и целей, а потому не пытаться выстроить суперзащищённую техническую систему;
* при проектировании должны максимально использоваться готовые, уже существующие на рынке и стандартизованные средства и компоненты. Если предприятие хочет обеспечить максимальную независимость от вендоров, то предпочтительно использование Open Source продуктов и технологий, но только достаточно зрелых, с подобающим размером сообщества и примерами внедрений на ответственных предприятиях;
* следует стремиться к простоте перевода существующих АС на DAL;
* лучше наращивать возможности реализуемого DAL постепенно, итерационно. Первый шаг должен быть простым, дешевым и быстрым, направленным на решение наиболее актуальных задач (в частности, переносимость реляционных СУБД);
* новый подход к управлению хранением может не ограничиваться техническими средствами, а включать, например, изменения в организационных механизмах предприятия в части управления инфраструктурой, а также в процессах проектирования и технического аудита АС.

**4.2 Реализация Business Logic Layer**

Business Logic Layer или бизнес-уровень инкапсулирует всю бизнес-логику, все необходимые вычисления, получает объекты из уровня доступа к данным (DAL) и передаёт их на уровень представления и наоборот, получает данные с уровня представления и передаёт их на уровень данных.

Бизнес слой нужен для сокрытия всей бизнес логики текущего приложения, причем, как платформа приложения может быть любого вида, так и данные приложения могут быть с любого источника.

Большую часть работы, которая заключается во взаимодействии PL и DAL, осуществляет несколько классов, код, одного из которых, а именно класс CommandBD, представлен в Приложении А.

**4.3 Реализация Presentation Layer**

Уровень представления отвечает за взаимодействие с пользователем. Часть функционала, которая зависит от данных из БД, связна с функционалом, прописанным в бизнес логике.

1. **Тестирование, проверка работоспособности и анализ полученных результатов**

Для корректной работы программы необходимо обеспечить защиту работы пользователя от ошибок и сбоев. Для этого лучше использовать конструкцию «if…else» для обработки ситуации неверного ввода информации пользователем.

В любой программе есть допустимые значения на ввод информации. Кроме того, пользователь может допустить ошибку при выборе поля и значений для него. Для всех этих ошибок нужно написать исключения для дальнейшей возможности работы приложения.

Далее будут представлено с тестирование приложения, показанные на рисунках 5.1-5.5:

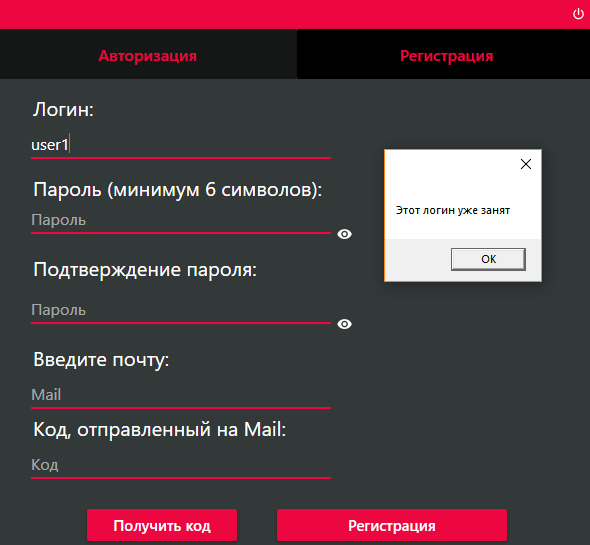


Рисунок 5.1 – Ввод занятого логина

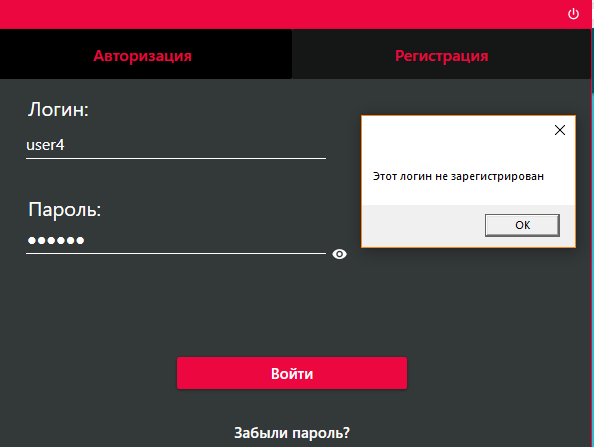


Рисунок 5.2 – Ввод несуществующего логина

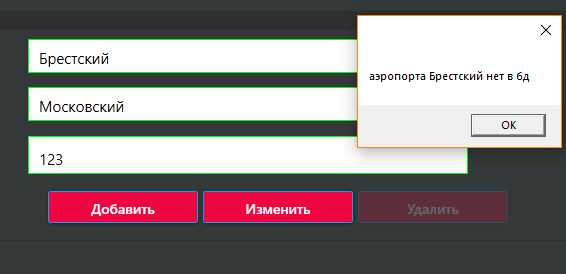
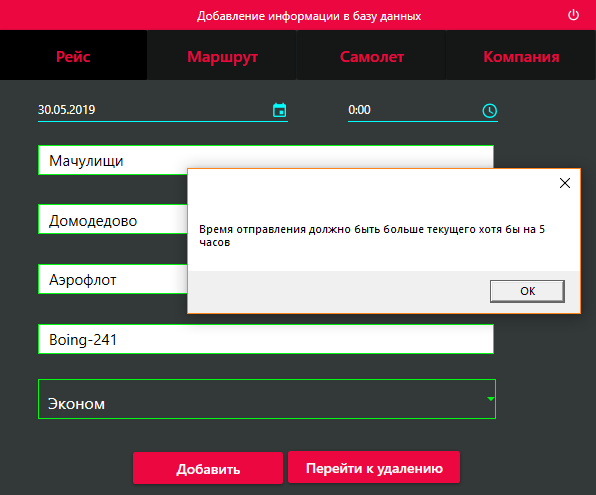


Рисунок 5.3 – Ввод некорректных значений при добавлении маршрута



5.4 – Попытка ввести неверное время при добавлении рейса

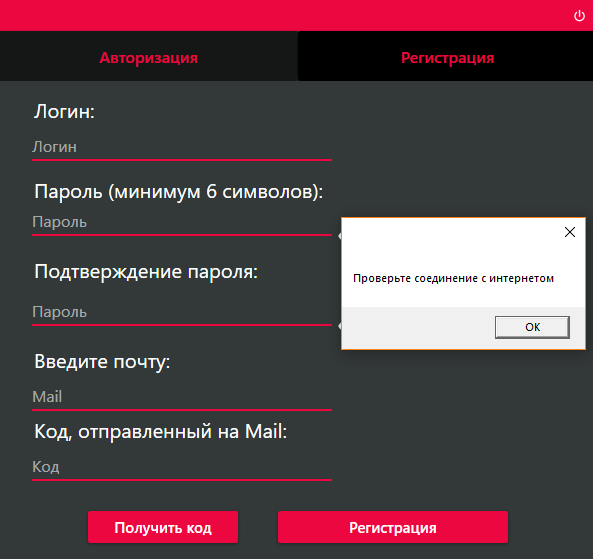


Рисунок 5.5 – Проверка на наличие интернет-соединения

Были затронуты всевозможные ошибки пользователя, для более лучшего теста программы следует использовать динамическое тестирование интерфейса, которые проводится во время выполнения программы.

1. **Руководство пользователя по использованию**

Правила заполнения полей при регистрации:

- логин должен быть уникальным;

- электронный адрес должен быть корректным;

- электронный адрес должен быть уникальным;

- подтверждение почты осуществляется с помощью, высланного на почту, кода;

- пароль должен состоять минимум из 6-ти символов.

Правила заполнения полей при авторизации:

- логин должен быть введён верно;

- пароль должен быть введён верно.

Правила заполнения полей при восстановлении пароля:

- логин должен быть верным;

- проверка пользователя осуществляется с помощью кода, высланного на почту которая указывалась при регистрации данного аккаунта;

Правила добавления данных в базу данных:

- необходимо учитывать связь между таблицами;

- некоторые таблицы требуют ввода существующих данных;

- данные должны быть корректными.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения работы были достигнуты следующие результаты:

- изучены основы объектно-ориентированного программирования и применены в практической разработке программного средства;

- исследована сфера управления пользователями приложения;

- сформулирована теория, что внедрение многоуровневой архитектуры способствует простому внедрению дополнительного функционала, простому расширению проекта, также повышает эффективность работы нескольких человек над проектом, возможность разрабатывать параллельно различные части проекта, и упрощает тестирование программного средства;

- показано и испытанно взаимодействие между уровнями проектирования во время выполнения программы;

- подготовлены материалы для дальнейшей разработки приложения и изучения данной сферы.

**СПИСОК ЛИТЕРАТУРЫ**

[1] Wikipedia [Свободная энциклопедия]. – Электронные данные. – Режим доступа: <https://en.wikipedia.org/wiki/Main_Page>.

[2] draw.io [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.draw.io/>.

[3] Metanit [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://metanit.com/>.

[4] Habr [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://habr.com/ru/>.

[5] StackOverflow [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://stackoverflow.com/>.

[6] Пацей, Н.В. Курс лекций по языку программирования C# / Н.В. Пацей. Минск: БГТУ, 2016. – 175 с.

[7] Microsoft Docs [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://docs.microsoft.com/en-us/>.

**ПРИЛОЖЕНИЕ А**

public class CommandBD

{

BdWork bd = new BdWork();

public DataTable Select<T>(string tableFrom, string where, T wheretxt)

{

return bd.Select(tableFrom, where, wheretxt);

}

public void Insert(string tableTo, string[] Param, string[] Param2, ArrayList arraylist)

{

bd.Insert(tableTo, Param, Param2, arraylist);

}

public DataTable SelectFewParWhere(string tableFrom, string[] where, ArrayList whereValue)

{

return bd.SelectFewParWhere(tableFrom, where, whereValue);

}

public void Delete<T>(string table, string where, T wherevalue)

{

bd.Delete(table, where, wherevalue);

}

public void Update<T, V>(string tableTo, string param, T paramValue, string where, V whereValue)

{

bd.Update(tableTo, param, paramValue, where, whereValue);

}

public DataTable SelectAll(string table)

{

return bd.SelectAll(table);

}

public DataTable MakeQuery(string query)

{

return bd.MakeQuery(query);

}

public void RebieldIndex(string[] i, string[] t)

{

bd.RebuildIndex(i, t);

}

}

**ПРИЛОЖЕНИЕ Б**

class WorkBronPage

{

StackPanel s = null;

CommandBD cbd = new CommandBD();

Validator valid = new Validator();

public bool CheckAllFields()

{

foreach(var ch in s.Children)

{

Grid grid = (Grid)ch;

foreach (var child in grid.Children)

{

if (child.GetType() == typeof(StackPanel))

{

StackPanel st = (StackPanel)child;

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

{

TextBox txtb = (TextBox)st.Children[i];

if (valid.EmptyString(txtb.Text))

return false;

}

}

}

}

return true;

}

public void Refresh(MainWindow main)

{

DataTable dt = cbd.Select("users", "userLogName", User.GetInstance().userLogName);

main.to.Text = main.from.Text = main.class\_of\_flight.Text = "";

main.count\_ch.Content = main.count\_vs.Content = "0";

main.countofpassangers.Text = "0";

User.GetInstance().userCountOrders = int.Parse(dt.Rows[0]["count\_of\_orders"].ToString());

}

}