Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Программирование сетевых приложений»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  старший преподаватель  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | \_\_\_.\_\_\_\_.2022 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«РАЗРАБОТКА СИСТЕМЫ ПРОДАЖИ АВИАБИЛЕТОВ»**

БГУИР КП 1-40 01 02-08  ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 073601  Карапаева В.А.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку 09.12.2022  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2022

СОДЕРЖАНИЕ

[Введение 8](#_Toc78894453)

[1 Анализ и моделирование предметной области программного средства 9](#_Toc78894454)

[1.1 Описание предметной области 9](#_Toc78894455)

[1.2 Разработка функциональной модели предметной области 9](#_Toc78894456)

[1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 12](#_Toc78894457)

[1.4 Разработка информационной модели предметной области 13](#_Toc78894458)

[1.5 Модели представления программного средства и их описание 16](#_Toc78894459)

[2 Проектирование и конструирование программного средства 19](#_Toc78894460)

[2.1 Постановка задачи 19](#_Toc78894461)

[2.2 Архитектурные решения 19](#_Toc78894462)

[2.3 Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства 23](#_Toc78894463)

[2.4 Проектирование пользовательского интерфейса 25](#_Toc78894464)

[2.5 Обоснование выбора компонентов и технологий для реализации программного средства 27](#_Toc78894465)

[3 Тестирование и проверка работоспособности программного средства 27](#_Toc78894466)

[4 Руководство по развертыванию и использованию программного средства 31](#_Toc78894467)

[Заключение 39](#_Toc78894468)

[Список использованных источников 40](#_Toc78894469)

[Приложение А](#_Toc78894470) [(обязательное)](#_Toc78894471) [Отчет о проверке на заимствования в системе «Антиплагиат» 41](#_Toc78894472)

[Приложение Б](#_Toc78894473) [(обязательное)](#_Toc78894474) [Листинг кода алгоритмов, реализующих бизнес-логику 42](#_Toc78894475)

[Приложение В](#_Toc78894476) [(обязательное)](#_Toc78894477) [Листинг скрипта генерации базы данных 44](#_Toc78894478)

[Ведомость документов курсового проекта 46](#_Toc78894479)

# Перечень условных обозначений, символов и терминов

|  |  |
| --- | --- |
| БД (база данных) | – представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины |
| Информационно-коммуникационные технологии | – процессы, методы поиска, сбора, хранения, обработки, предоставления, распространения информации и способы осуществления таких процессов и методов |
| Информационная система | – система, предназначенная для хранения, поиска и обработки информации, и соответствующие организационные ресурсы (человеческие, технические, финансовые), которые обеспечивают и распространяют информацию |
| Нормальная форма | – свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных |
| Среда выполнения | – вычислительное окружение, необходимое для выполнения компьютерной программы и доступное во время выполнения компьютерной программы |
| СУБД (система управления базами данных) | – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных |
| ТЗ (техническое задание) | – документ, содержащий требования заказчика к объекту разработки, определяющий порядок и условия её проведения |
| *API* (*application programming interface*) | – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой |
| *HTTP* (*HyperText Transfer Protocol*) | – протокол прикладного уровня передачи данных изначально – в виде гипертекстовых документов в формате *html*, в настоящий момент используется для передачи произвольных данных. |
| *IDE* (*Integrated development environment*) | – комплекс программных средств, используемый программистами для разработки программного обеспечения |
| *IDEF* | – методология функционального моделирования (англ. *function modeling*) и графическая нотация, предназначенная для формализации и описания бизнес-процессов |
| *Java* | – строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией *Sun* *Microsystems* |
| *MySQL Server* | – свободная реляционная система управления базами данных |
| *Sybase ASE* (*Adaptive Server Enterprise*) | – реляционная система управления базами данных компании *SAP*, одна из СУБД, использующая в качестве основного процедурного *SQL*-расширения язык *Transact-SQL* |
| *UML* (*Unified Modeling Language*) | – язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур |
| *Usability* | – способность продукта быть понимаемым, изучаемым, используемым и привлекательным для пользователя в заданных условиях |
|  |  |

# Введение

В данный момент времени проблема автоматизации во всех ее проявлениях (комплексная, частичная) является актуальной проблемой для многих предприятий. Особенно это актуально для авиационной отрасли, благодаря которой осуществляется увеличение товарных потоков, налаживание международных связей.

Грамотное обеспечение авиаперевозок, поддержание высокого уровня развития аэропортового и аэродромного хозяйства является залогом комфорта и выгоды пассажиров и грузоотправителей, ключевым моментом, определяющим динамику экономического развития страны.

Такой вид транспорта развивался более сотни лет, улучшая показатели скорости, длительности и стабильности полета, комфорта и меньшей стоимости. В настоящее время, используя услуги воздушных транспортных компаний пассажир может за пару часов пролететь половину земного шара и попасть на другой континент.

Ежедневно аэропортом обслуживается сотни тысяч пассажиров, поэтому очень важен контроль безопасности пассажиров и быстрая регистрация всех пассажиров. Поэтому в нем должна быть четко отрегулированная структура и график, ответственный и обученный персонал, а также удобное и понятное программное обеспечение.

Целью данной курсовой работы является оптимизация процесса продажи авиабилетов.

Поставленная цель потребовала решение следующих задач:

– рассмотреть предметную область и разработать модель, отражающую бизнес-процесс на предприятии.

– найти и проверить уже зарекомендовавшие себя приложения;

– провести конкурентный анализ;

– описать постановку задачи на разработку программного средства;

– разработать и описать алгоритм работы программы;

– выполнить реализацию программного средства;

– разработать и описать руководство пользователя.

Объектом исследования является процесс продажи авиабилетов.

Ключевые слова: ПРОГРАММНОЕ СРЕДСТВО, АЛГОРИТМЫ, ПОЛЬЗОВАТЕЛЬ, ПРОДАЖА АВИАБИЛЕТОВ, КЛИЕНТ-СЕРВЕР.

Курсовой проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности составляет 93,86%. Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников». Скриншот приведен в приложении А.

# Анализ и моделирование предметной области программного средства

## 1.1 Описание предметной области

В данном курсовом проекте рассматривается программное средство для поиска и продажи авиабилетов с целью уменьшения времени и улучшения качества обслуживания пассажиров.

Аэропорт – предприятие, которое обеспечивает быструю, безопасную и эффективную транспортировку по воздуху пассажиров, багажа, грузов и почты, которое осуществляет прием и отправление летательных аппаратов, используя средства, которые необходимы для их взлета или посадки а так же техническое обслуживание и ремонт. В связи с таким большим спектром услуг, авиаперелеты пользуются высокой популярностью. Вследствие этого продаваемые на рейсы билеты востребованы и с высокой вероятностью найдут своего покупателя, при условии, что авиакомпания обеспечила клиенту полноценный доступ к нужной ему информации.

В настоящее время на рынке предложения услуг авиаперелетов действует огромное количество фирм(Amadeus, Galileo,Ryanair.com), каждая из которых отличается собственными методами привлечения клиентов и спецификой работы. Каждая из фирм пытается показать клиенту выгоду, которую он получит, если выберет ту или иную фирму. И, согласно статистике, клиенты выбирают фирму, которая быстрее выполняет свою работу. Следовательно, автоматизация процесса продажи авиабилетов увеличит продажи и количество клиентов.

Для примера возьмем самую популярную систему продаж авиабилетов в мире – Amadeus. Разработанные данной компаний программные продукты широко используются в туристической индустрии, что помогает в расширении границ из-за сотрудничества с ведущими авиаперевозчиками и турагентствами. Согласно статистике за 2019 год, объем бронирований авиабилетов превышает отметку в 500 миллионов. Но у этой программы есть значительные минусы в виде не самого привлекательного и понятного интерфейса и отсутствие русского языка, но даже это не делает ее менее популярной и странах СНГ.

Из вышесказанного делаем вывод, что автоматизация системы продажи авиабилетов просто необходима в нынешнем мире.

## 1.2 Разработка функциональной модели предметной области

Для более ясного представления функций информационной системы была разработана функциональная модель. Такая модель способна обеспечить полное представление о функционировании выбранного процесса, потоках информации и материалов в нем.

Для создания функциональной модели был выбран стандарт IDEF0. IDEF0 – методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчиненность объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность. Методология IDEF0 предписывает построение иерархической системы диаграмм — единичных описаний фрагментов системы. Сначала проводится описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция — система разбивается на подсистемы и каждая подсистема описывается отдельно. Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности.

На рисунке 1.1 представлена контекстная диаграмма верхнего уровня с функциональной моделью «Продать билет онлайн», а также определены потоки входных и выходных данных, механизмы ограничения и управления данными.

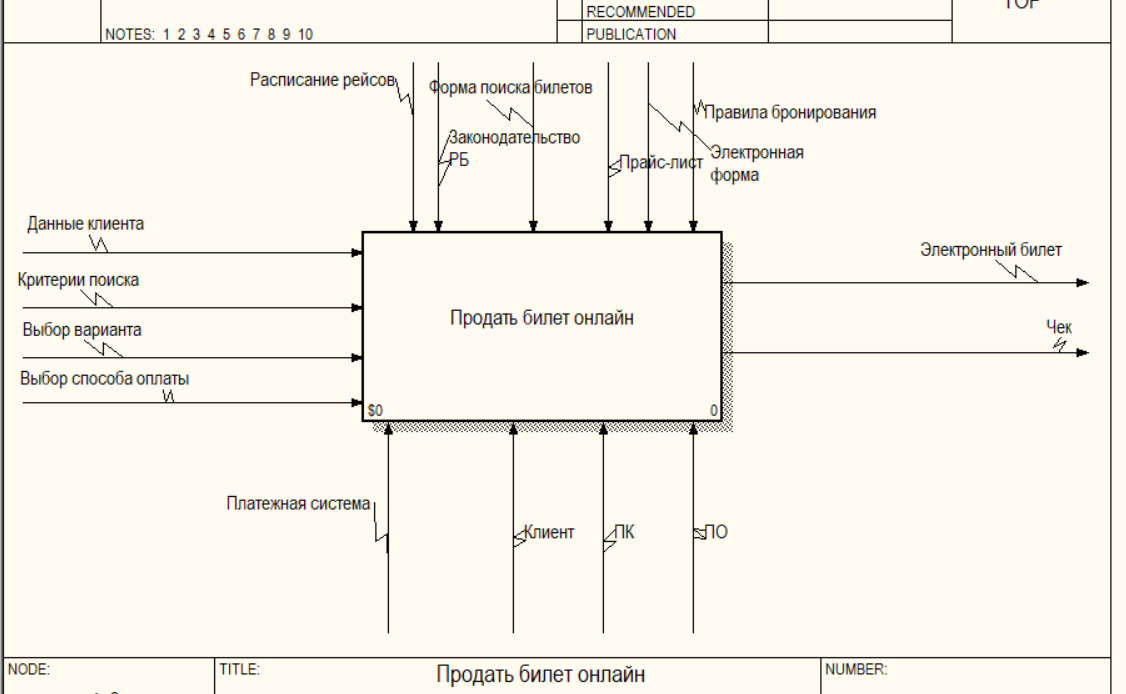


Рисунок 1.1 – Контекстная диаграмма «Продажа билета онлайн»

На рисунке 1.2 отображена декомпозиция контекстной диаграммы, состоящая из трех блоков: «Поиск и выбор варианта» «Забронировать билет», «Оплатить билет».

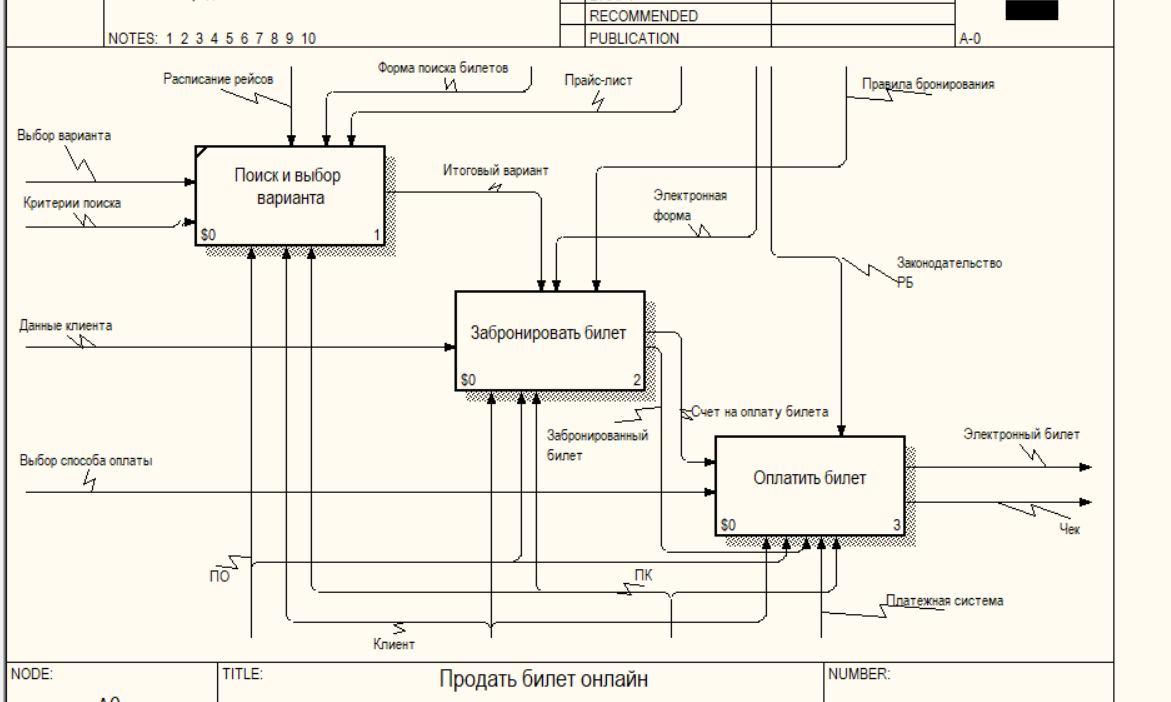


Рисунок 1.2 – Декомпозиция контекстной диаграммы «Продать билет онлайн»

На рисунке 1.3 отображена декомпозиция работы «Забронировать билет», состоящая из трех блоков: «Заполнить электронную форму», «Проверить правильность данных», «Подтвердить правильность данных и получить счет для оплаты».

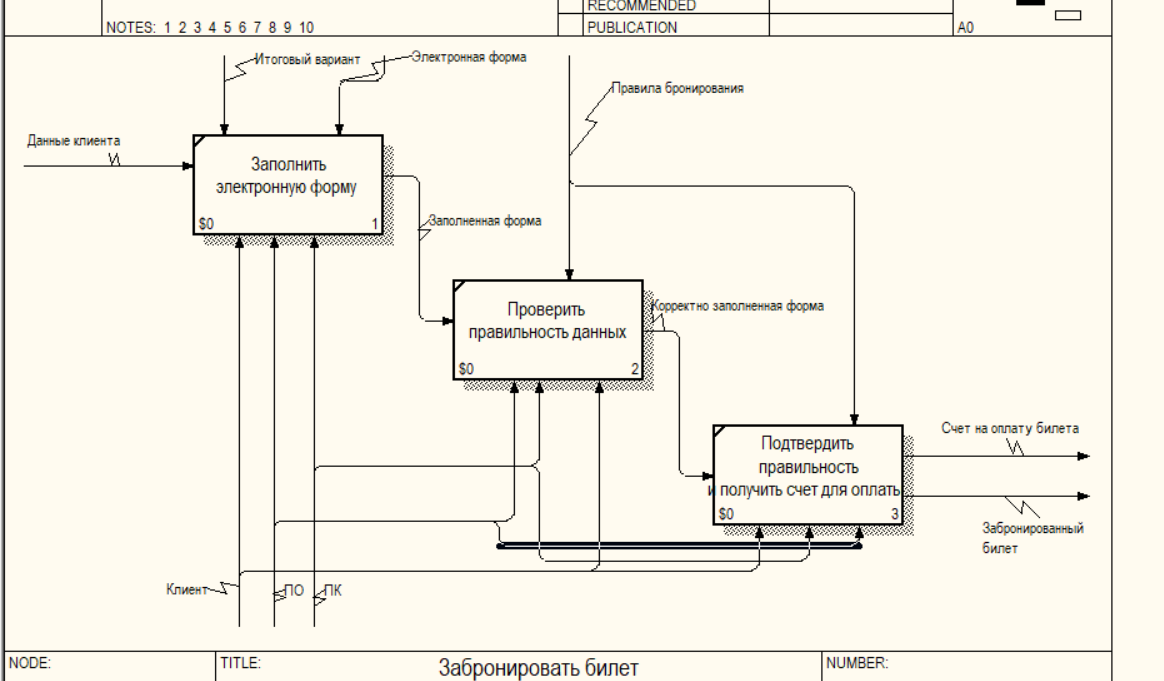


Рисунок 1.3 – Декомпозиция работы «Забронировать билет»

На рисунке 1.4 отображена декомпозиция работы «Оплатить билет», состоящая из трех блоков: «Выбрать способ оплаты», «Ввести данные для оплаты(карта)», «Произвести оплату».

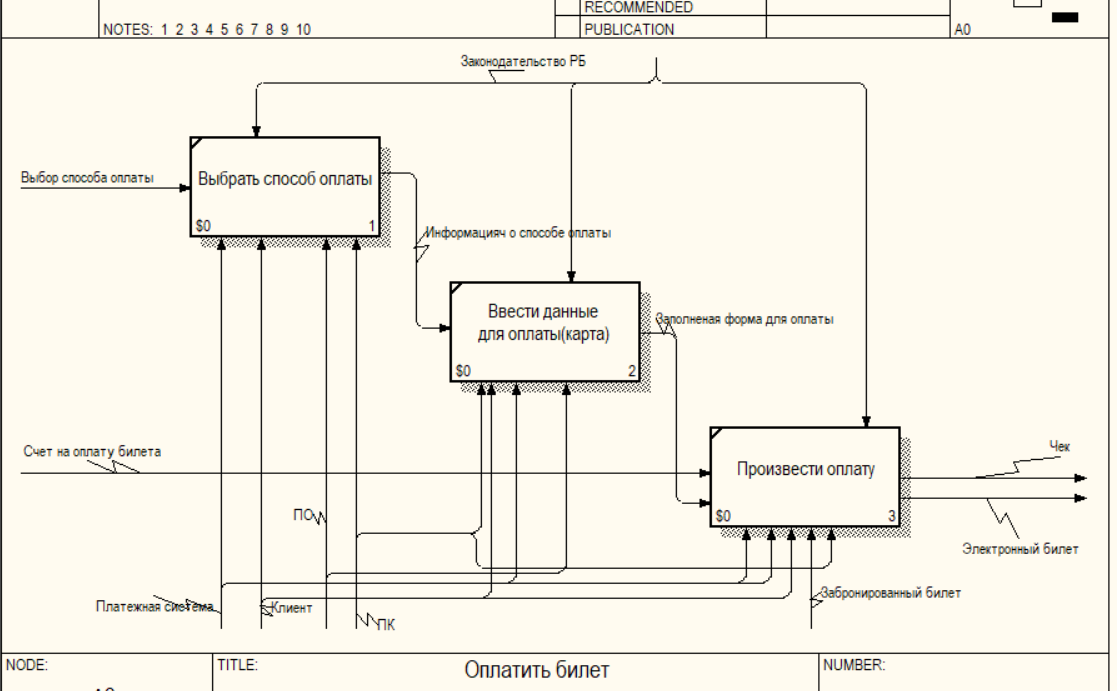


Рисунок 1.4 – Декомпозиция работы «Оплатить билет»

## 1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований

Данное программное приложение представляет собой систему продажи авиабилетов. Оно включает в себя два способа пользования данными: от имени администратора и от имени пользователя. Варианты использования программы у них различаются.

Пользователь может войти или зарегистрироваться. Пользователь может просматривать табло рейсов, покупать билеты на выбранный им рейс, добавлять билеты в корзину, а также просматривать корзину, выводить билеты в файл, просматривать архив билетов. Так же возможна функция просмотра данных пассажира, на которого будет оформлен билет. Есть возможность выбрать наиболее подходящий билет, согласно критериям, который ввел пользователь в своем личном кабинете. При необходимости возможно получение информации о правилах полета, а так же информации о приложении.

Теперь разберем возможности администратора. Зарегистрироваться администратором конечно же нельзя. У администратора есть возможность просматривать профиль пассажира. Также администратор имеет полномочия, для удаления, создания, редактирования и просмотра расписания рейсов, самолетов и т.д. Есть возможность получить отчет о выбранной информации( два подхода формирования), а так же диаграммы(два подхода формирования). Администратор может редактировать бонусную систему для пассажира, согласно рейтингу пассажира. Так же для администратора будет доступна статистика : ТОП 3 места, куда летают; дни, в которые были проданы большее количество билетов; каких билетов продано больше(эконом или бизнес).

Ниже приведена диаграмма вариантов использования (Рисунок 1.5)

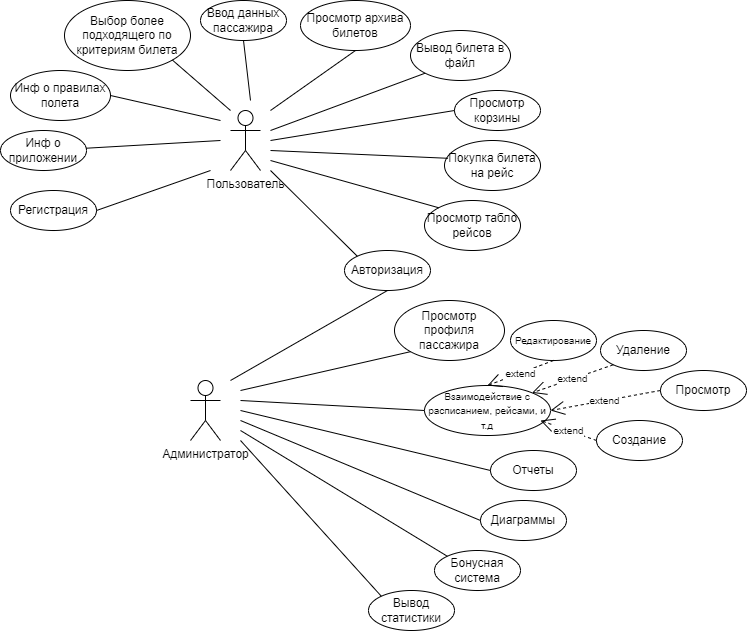


Рисунок 1.5 – Диаграмма вариантов использования

## 1.4 Разработка информационной модели предметной области

Для выделения сущностей, атрибутов и первичных ключей происходит построение информационной модели предметной области. Общепринятым представлением графического изображения реляционной модели данных является ER-диаграмма. ER-диаграмма показывает связь данных между собой и какие данные хранит система.

При проектировании системы было принято решение использовать следующие сущности:

* *users*;
* *pilots*;
* *airports*;
* *user\_persons;*
* *user\_tickets*;
* *persons;*
* *tickets;*
* *flights;*
* *planes.*

Графическое отображение информационной модели приведено на диаграмме рисунке 1.6.

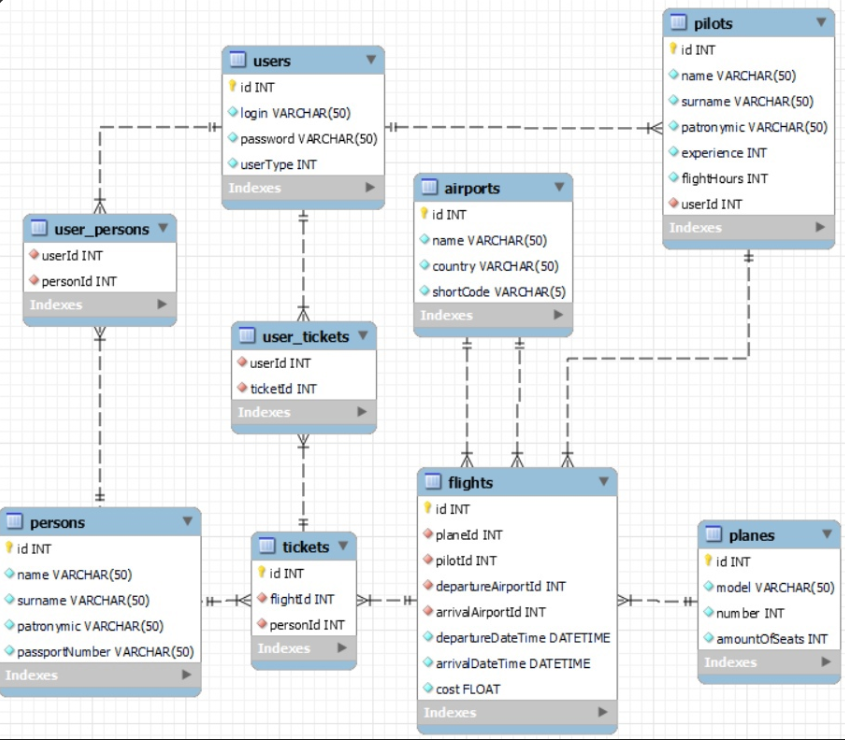


Рисунок 1.6 – Информационная модель системы

Ниже представлено подробное описание всех сущностей, входящих модель.

Сущность *users* содержит в себе следующие атрибуты:

* *id* – хранит идентификационный номер авторизованного пользователя;
* *login* – отвечает за хранение логина пользователя;
* *password* – хранит пароль;
* *userType* – необходимо для хранения типа пользователя.

Сущность *user\_persons* наследуется от сущностей *users* и *persons* содержит следующие атрибуты:

* *person\_id*– атрибут унаследованный от сущности *persons*;
* *user\_id* – атрибут унаследованный от сущности *users*.

Сущность *persons* содержит в себе следующие атрибуты:

* *id* – хранит идентификационный номер авторизованного пользователя;
* *name* – отвечает за хранение имени пользователя;
* *surname* – хранит фамилию;
* *patronymic* – необходимо для хранения отчества пользователя;
* *passportNumber* – хранит информацию о номере паспорта.

Сущность *tickets* содержит в себе следующие атрибуты:

* *id* – хранит идентификационный номер авторизованного пользователя;
* *flightId* – отвечает за хранение идентификационного номера рейса;
* *personId* – хранит идентификационный номер пользователя.

Сущность *flights* содержит в себе следующие атрибуты:

* *id* – хранит идентификационный номер авторизованного пользователя;
* *planeId* – отвечает за хранение идентификационного номера рейса;
* *pilotId* – хранит идентификационный номер пилота;
* *departureAirportId* – необходимо для хранения идентификационного номера аэропорта, из которого осуществляется вылет;
* *arrivalAirportId* – необходимо для хранения идентификационного номера аэропорта, в который осуществляется прилет ;
* *departureDateTime* – хранит время и дату вылета;
* *arrivalDateTime* – хранит время и дату прилета;
* *cost* – стоимость билета.

Сущность *planes* содержит в себе следующие атрибуты:

* *id* – хранит идентификационный номер авторизованного пользователя;
* *model* – отвечает за хранение модели самолета;
* *number* – хранит номер самолета;
* *amountOfSeats* – необходимо для хранения количества мест в самолете.

Сущность *airports* содержит в себе следующие атрибуты:

* *id* – хранит идентификационный номер авторизованного пользователя;
* *name* – отвечает за хранение названия самолета;
* *country* – хранит данные о стране, где самолет находится;
* *shortCode* – необходимо для хранения короткого номера самолета.

Сущность *user\_tickets* содержит в себе следующие атрибуты:

* *userId* – хранит идентификационный номер авторизованного пользователя;
* *ticketId* – отвечает за хранение идентификационного номера билетов.

*SQL*-скрипт для генерации базы данных приведен в приложении В.

## 1.5 Модели представления программного средства и их описание

Диаграмма состояний по существу является графом специального вида, который представляет некоторый автомат. Вершинами этого графа являются состояния и некоторые другие типы элементов автомата (псевдосостояния), которые изображаются соответствующими графическими символами. Дуги графа служат для обозначения переходов из состояния в состояние.

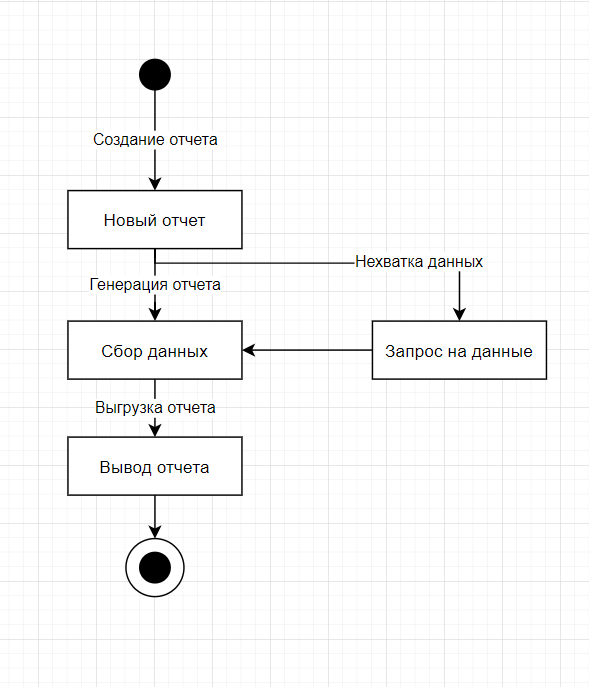


Рисунок 1.7 – Диаграмма состояний

На рисунке 1.8 показана диаграмма последовательности процесса авторизации пользователя.

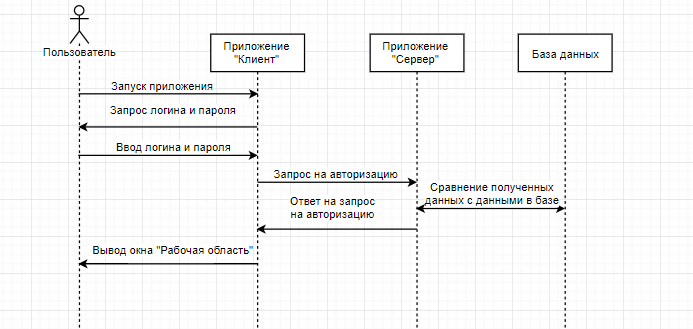


Рисунок 1.8 – Диаграмма последовательности

Диаграмма развёртывания. С помощью диаграммы развертывания отобразим элементы и компоненты системы, существующие на этапе ее исполнения.

Диаграмму развертывания системы приведена на рисунке 1.9.

На диаграмме отражено, что для ПК пользователя обязательно наличие среды выполнения *jdk* 1.9+, также, как и для ПК, который будет выступать в качестве сервера. Взаимодействие между приложением клиента и сервера будет осуществляться посредством *TCP*/*IP* соединения. Для соединения с базой данных необходимо подключение *JDBC* *Driver*. База данных хранится на *MySQL* *SERVER*.

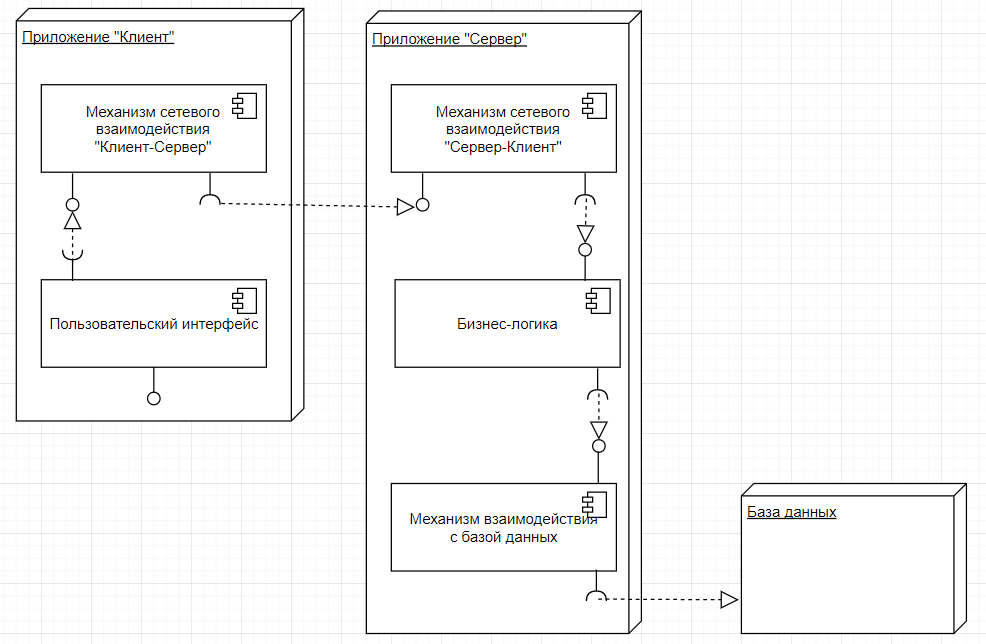


Рисунок 1.9 – Диаграмма развертывания

Таким образом, в данной главе курсового проекта представлены моделирование предметной области и разработка требований к программному средству, анализ и формализация бизнес-процессов службы управления персоналом, разработаны спецификация функциональных требований, информационная модель программного средства, а также модели представления программного средства на основе языка *UML*.

# Проектирование и конструирование программного средства

## 2.1 Постановка задачи

В данном курсовом проекте представлена цель сократить временные затраты на продажу авиабилетов за счет автоматизации обработки необходимой информации и обеспечение надежного хранилища данных с помощью создания БД. Для этого необходимо разработать такое приложение, которое поможет достичь данную цель.

Для разработки качественного программного продукта необходимо разобраться в процессе продажи билетов.

Так как главной задачей любой авиакомпании является организация лучших условий перелета для своих клиентов, то для привлечения большего их количества, необходимо разработать удобное и привлекательное пользовательское приложение.

Так как в аэропорту ежедневно улетаю и прилетают десятки самолетов по разным направлениям, то для надежного и удобного хранения информации о них необходимо спроектировать базу данных, которая служила бы надежным хранилищем данных. БД позволит облегчить поиск необходимой информации как для сотрудников аэропорта, так и для его клиентов. Для корректной и быстрой работы необходимо предусмотреть возможности добавления, удаления, поиск, фильтрации, сортировки и редактирования информации в базе данных. Для хранения информации в данном курсовом проекте будет использован MySQL Server. Подключение к БД осуществляется во время авторизации и в зависимости от того, кто вошел в систему, вошедший будет получать определенный набор функциональных возможностей.

Так как добавление вылетов и составлением расписания сеансов занимается администратор, то для его эффективной работы ему будет предоставлен полный доступ к информации о фильмах и сеансах, хранимых в БД.

Клиент аэропорта сможет с легкостью просмотреть всю информацию о рейсе, а также просмотреть рейсы как за конкретный период, так и на конкретный день. Также клиент сможет очень быстро забронировать билет на удобное ему время и дату, а после подтверждения получит файл с билетом для оплаты и предъявления его на стойке регистрации.

Для анализа работы будет вестись отчетность покупок билетов, администратор по итогу месяца будет составлять отчет, как в целом по дням, так и на конкретный рейс. Это позволит скорректировать работу, а также понять, какие направления больше нравятся клиентам, что в дальнейшем может быть эффективно использовано для улучшения работы аэропорта и повышения прибыли.

## 2.2 Архитектурные решения

Диаграммы классов используются при моделировании программных средств наиболее часто. На диаграммах классов показываются классы, интерфейсы и отношения между ними.

Диаграммы классов показаны на рисунках 2.1 – 2.2.

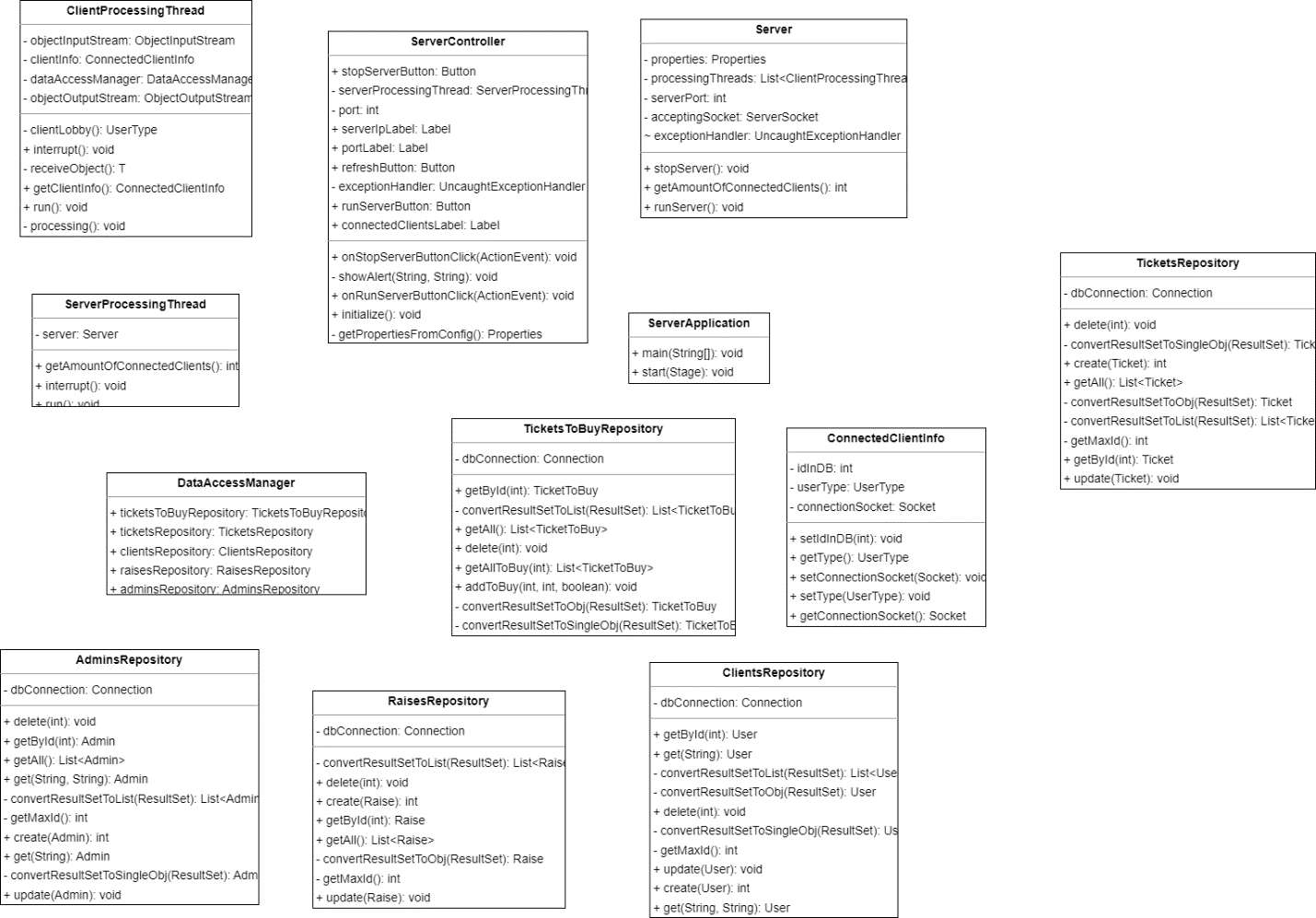


Рисунок 2.1 – Диаграмма классов сервера

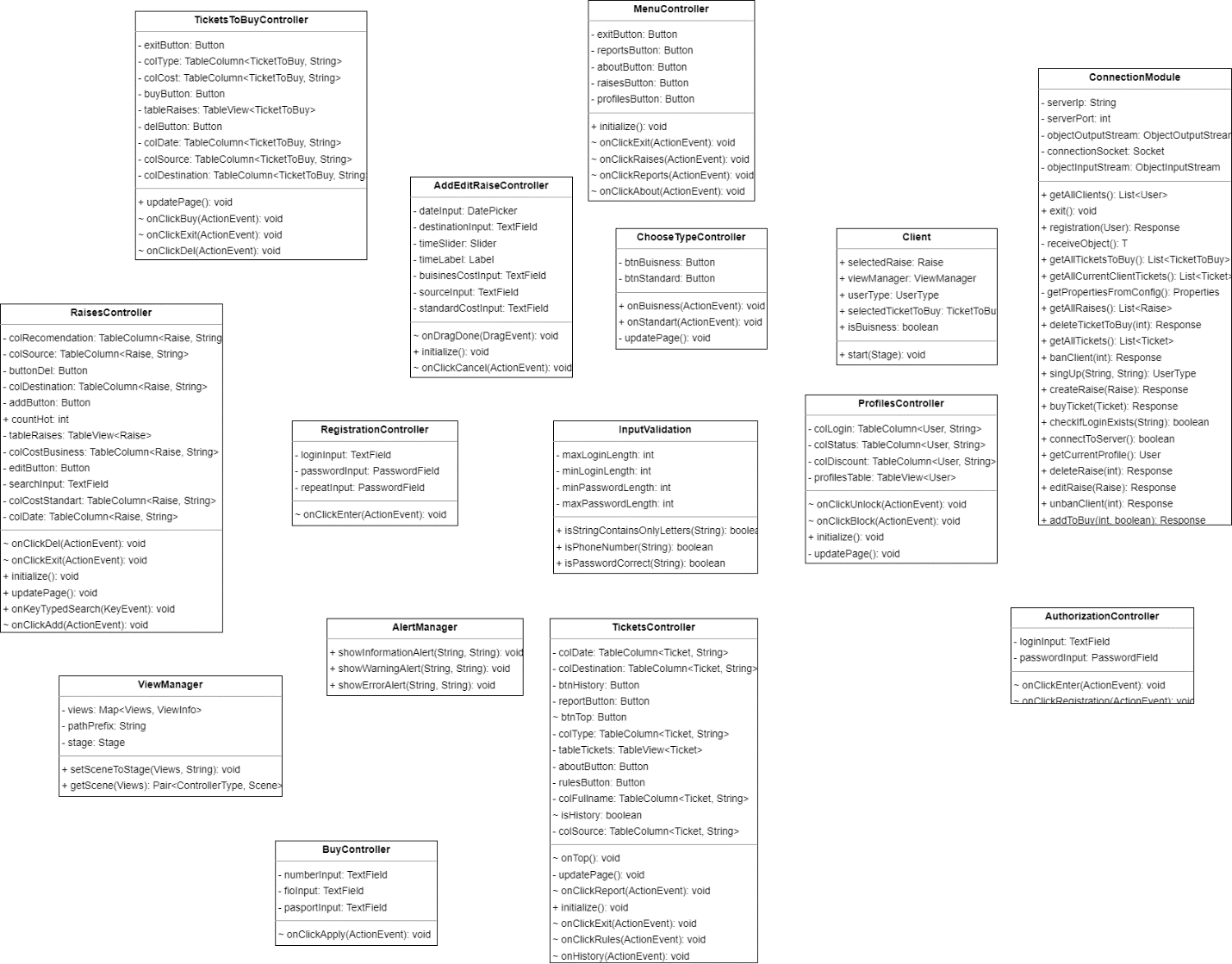


Рисунок 2.2 – Диаграмма классов клиента

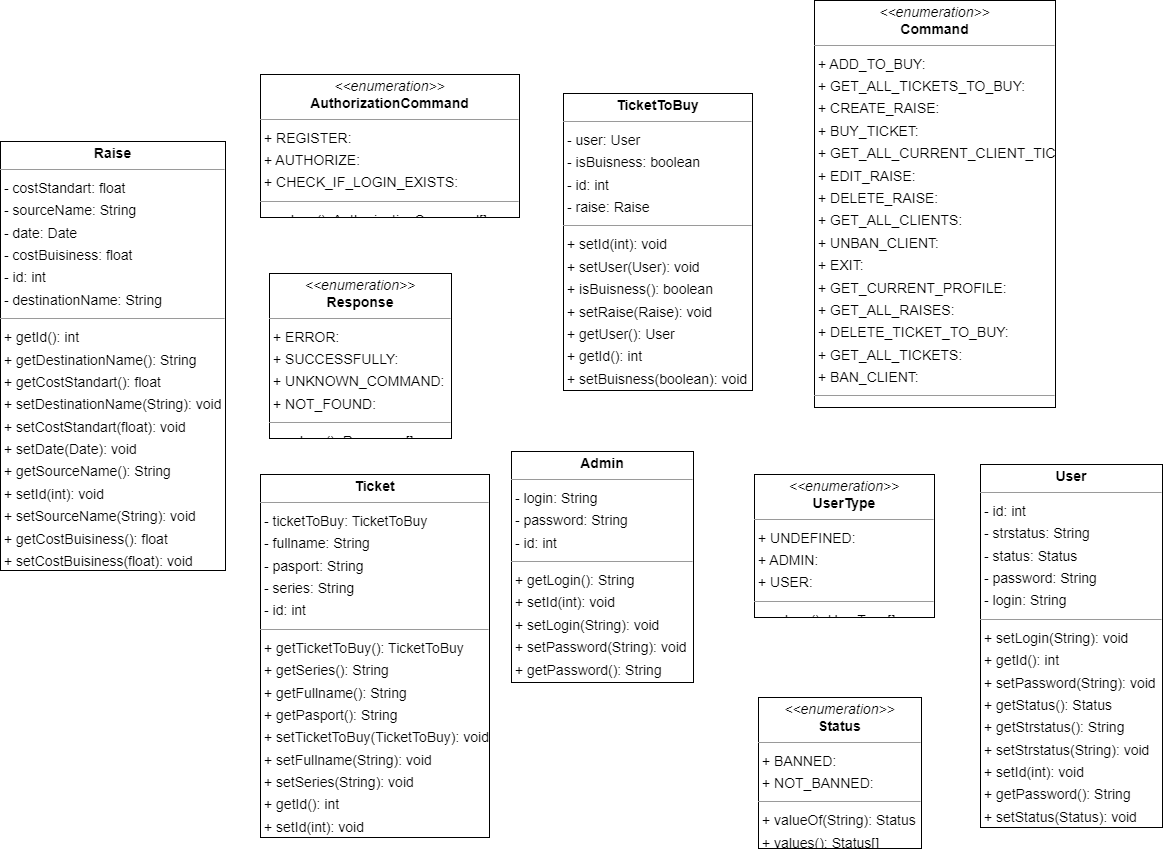


Рисунок 2.3 – Диаграмма классов пакета TransferEntitiesDiagram

Экземпляры клиентских приложений размещаются на машинах сотрудников организации, на которых предварительно должен быть установлен JDK 1.8 и выше. Данные, необходимые для установки соединения с сервером, хранятся в конфигурационном файле клиентского приложения и доступны для настройки пользователем. Настройки, используемые всеми приложениями системы, хранятся в одной из таблиц базы данных. Это позволяет производить изменения, которые распространяются сразу на все экземпляры приложений, с другой стороны, доступ к настройкам системы закрыт для рядовых пользователей.

Система предусматривает один способ ввода информации – вручную через пользовательский интерфейс приложения и один способ вывода – на экран машины сотрудника.

Экземпляры приложений могут взаимодействовать между собой для пересылки уведомлений сотрудникам. Для этого используются протоколы стека TCP/IP. Сетевое взаимодействие обеспечивается средствами архитектуры java.net, сетевые настройки хранятся в базе данных.

Диаграмма компонентов описывает особенности физического представления системы. Она позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный и исполняемый код. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними. Диаграмму можно увидеть на рисунке 2.4.

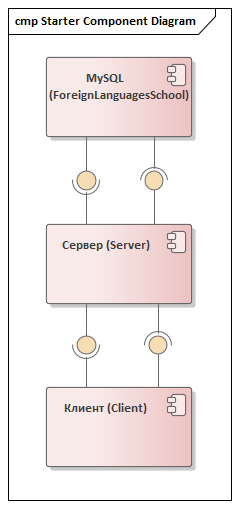


Рисунок 2.4 – Диаграмма компонентов

## 2.3 Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства

Рассмотрим алгоритмы реализации ряда методов серверной части, соответствующих вариантам использования.

При запуске открывается главное окно, в котором пользователь должен пройти авторизацию. При успешной авторизации в зависимости от уровня доступа, пользователь получает разный набор возможностей для работы с программой. (см. рисунок 2.5).

Рассмотрим алгоритмы реализации ряда методов серверной части, соответствующих вариантам использования (см. рисунок 2.5).

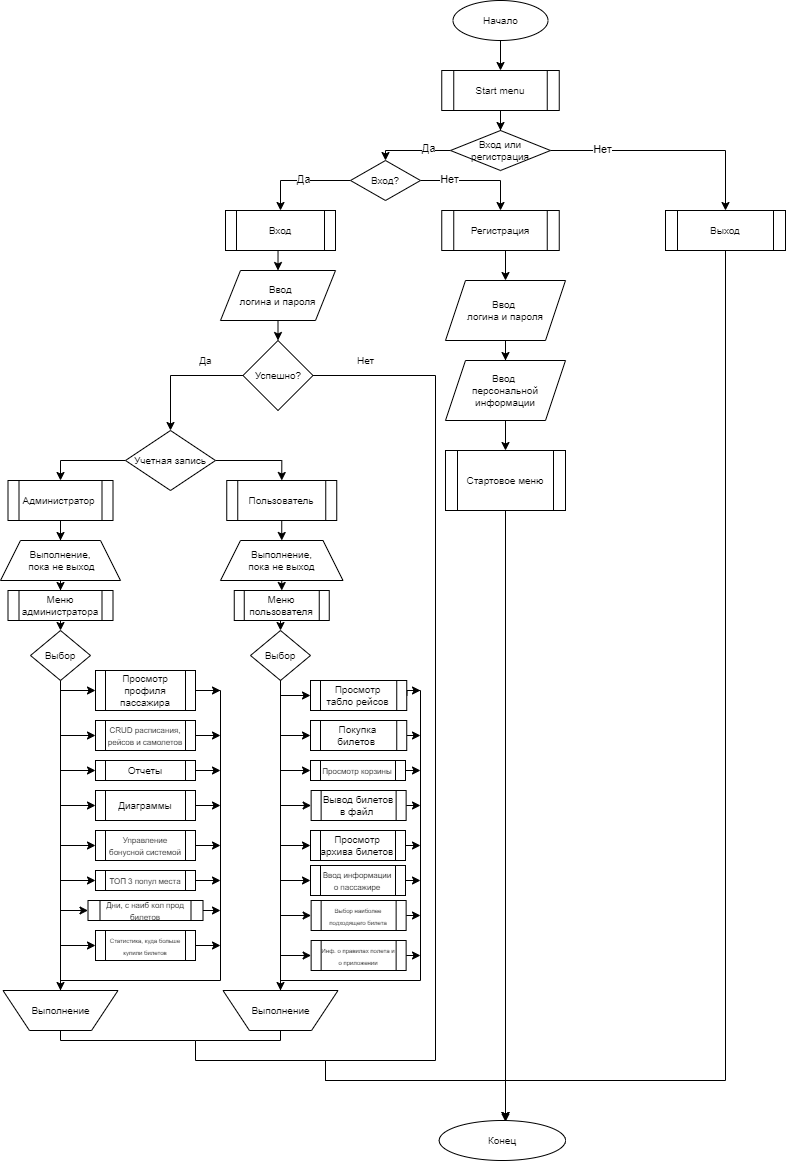


Рисунок 2.5 – Схема алгоритма работы программы

Алгоритм архитектуры «клиент-сервер» представлен на рисунке 2.6 .

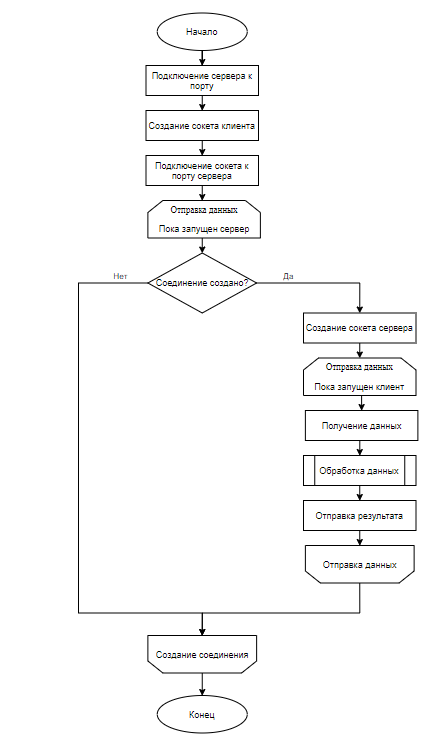


Рисунок 2.5 – Алгоритм клиент-серверного взаимодействия

Обобщенный алгоритм характеризует работу программы целиком.

## 2.4 Проектирование пользовательского интерфейса

Пользовательский интерфейс – это интерфейс, с помощью которого человек может управлять программным средством. Пользовательский интерфейс должен быть удобным в использовании, чтобы взаимодействие с ним происходило на максимально интуитивном уровне.

Графический пользовательский интерфейс – это вид пользовательского интерфейса, в котором все элементы (кнопки, меню, пиктограммы, списки и другое) представленные пользователю на дисплее, выполнены в виде картинок, графики.

При запуске программы, пользователю выводится окно авторизации(рисунок 2.6).

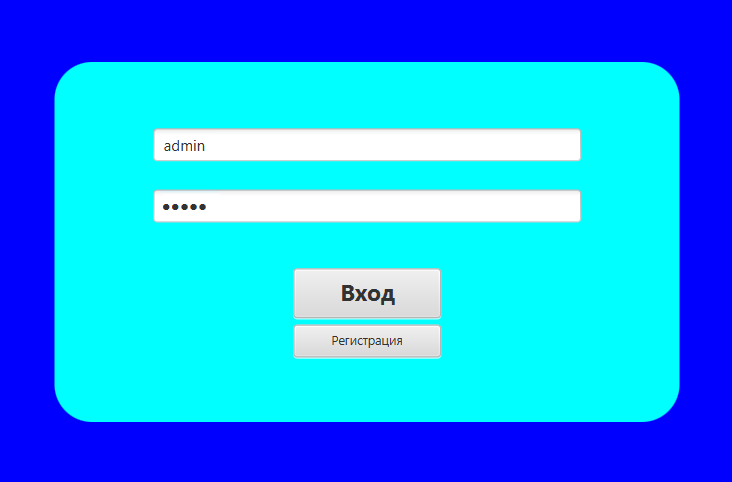


Рисунок 2.6 – Окно авторизации

При успешной авторизации администратора на экране появляется главное и основное меню администратора.

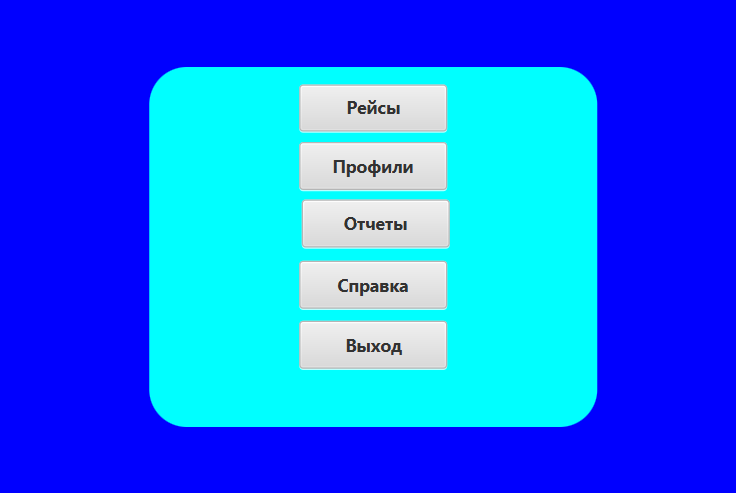


Рисунок 2.8 – Главное окно администратора

Все компоненты взаимодействуют друг с другом и связаны. Так же могут работать по отдельности. Для каждой части разработана таблица в которой содержатся и хранятся данные.

При запуске программы появится окно аутентификации (проверка имени и пароля пользователя программы). Данное окно служит для идентификации пользователя программы в целях предоставления ему определенных прав в работе с программой. После первой установки программы имеется только один пользователь – "Администратор". Он является полноправным владельцем всех данных, с которыми может взаимодействовать программа. Смысл создания дополнительных пользователей – ограничение доступа к программе. Таким образом, можно создать дополнительного пользователя, который будет иметь право добавлять или редактировать штатные единицы на предприятии.

## 2.5 Обоснование выбора компонентов и технологий для реализации программного средства

Данное программное приложение будет иметь архитектуру «клиент-сервер» и написано оно будет на языке программирования Java.

Сервер – логический процесс, который обеспечивает некоторый сервис по запросу от клиента. Обычно сервер не только выполняет запрос, но и управляет очередностью запросов, буферами обмена, извещает своих клиентов о выполнении запроса и т. д. Клиент – процесс, который запрашивает обслуживание от сервера. Процесс не является клиентом по каким-то параметрам своей структуры, он является клиентом только по отношению к серверу.

Достоинства архитектуры «клиент-сервер»:

– центральное хранилище файлов;

– простая управляемость при большом числе пользователей;

– сильная централизованная защита;

– нет дублирования кода.

Связь между сервером и клиентом в приложении осуществляется при помощи TCP/IP. Также в реализации системы используются следующие техники:

* разработка и использование собственной иерархии, расширение базовых классов, предоставляемых JDK;
* реализация двух паттернов проектирования;
* использование сокрытия данных (инкапсуляция), перегрузка методов, переопределение методов, абстрактные типы данных (интерфейсы, абстрактные классы), статические методы, обработка исключительных ситуаций.

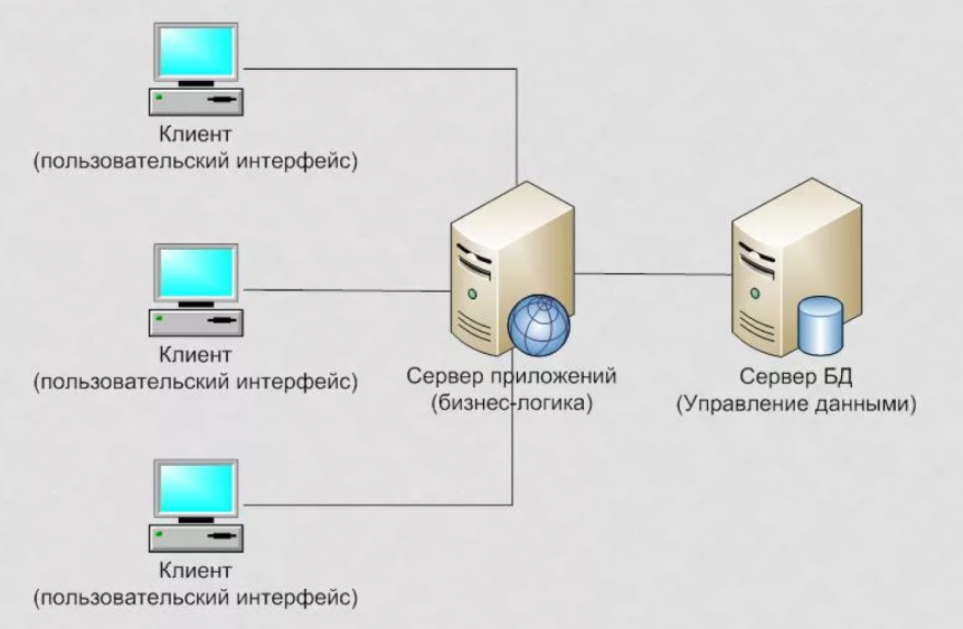


Рисунок 2.9 – Архитектура «клиент-сервер»

Для более удобного хранения данных необходимо спроектировать базу данных, которая может осуществить хранение данных, и дает возможность поиска информации, ее редактирование, добавление, удаление и фильтрацию. В качестве основной среды разработки в проекте используется Intellij IDEA. В качестве СУБД в курсовом проекте используется MySQL Workbench. Для создания UML-диаграмм в проекте используется средство *Drawio*, которое позволяет осуществлять создание диаграмм вариантов использования, состояний, последовательности с помощью просто моделирования. Для выполнения UML-моделей в стандарте IDEF0 используется CASE-средство CA AllFusion Process Modeler r7 (BPwin). Для информационного моделирования применяется средство CA AllFusion ERwin Data Modeler r7 (ERwin). Подключение к нему будет осуществляться при авторизации и в зависимости от полученных прав, пользователь будет получать определённый набор возможностей для выполнения своей работы.

# Тестирование и проверка работоспособности программного средства

Тестирование – это процесс проверки функционала программы с целью подтверждения того, что она работает в соответствии с определёнными требованиями. Юнит-тест (unit test), или модульный тест, — это программа, которая проверяет работу небольшой части кода. Разработчики регулярно обновляют сайты и приложения, добавляют фичи, рефакторят код и вносят правки, а затем проверяют, как всё работает. Проверять всё это надо.

Junit — библиотека для модульного тестирования программного обеспечения на языке java[8].

В ходе тестирования функций получения роли авторизации и выборки из таблицы выяснилось, что функция работает корректно и проходит все тесты. Результат тестирования функции представлен на рисунке 3.1.

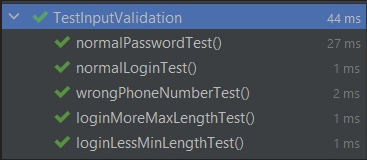


Рисунок 3.1 – Результат тестирования

Код теста представлен на рисунке 3.2.



Рисунок 3.2 – Тест на добавление объекта

Тестирование необходимо потому, что все мы совершаем ошибки. Некоторые из них могут быть незначительными, в то время как другие – иметь самые разрушительные последствия. Все, что производится человеком, может содержать ошибки. Именно поэтому любой продукт нуждается в проверке – тестирование, прежде чем его можно будет эффективно и безопасно использовать.

# Руководство по развертыванию и использованию программного средства

В первую очередь необходимо пройти авторизацию, введя логин и пароль (см. рисунок 4.1).

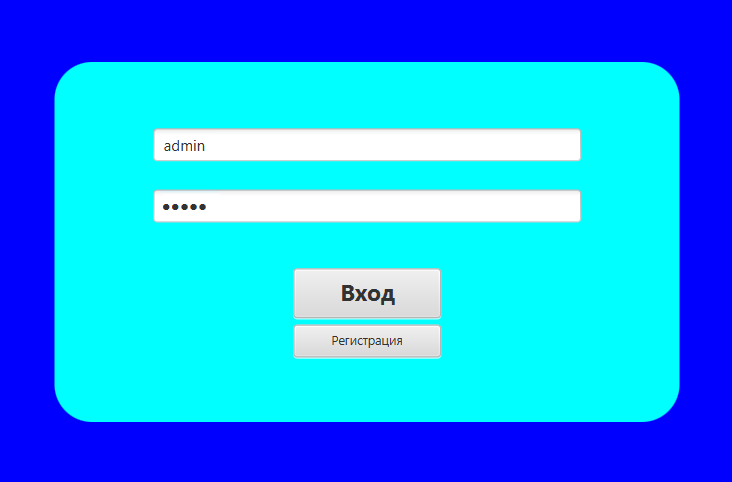


Рисунок 4.3 – Окно авторизации

Для наглядной демонстрации работы программы была произведена авторизация под логином администратора. У всех администраторов имеется полный набор возможных действий. В данном случае администратор имеет полный уровень прав и может выполнять любые манипуляции с базой данных пользователей. На рисунке 4.2 показано меню администратора.

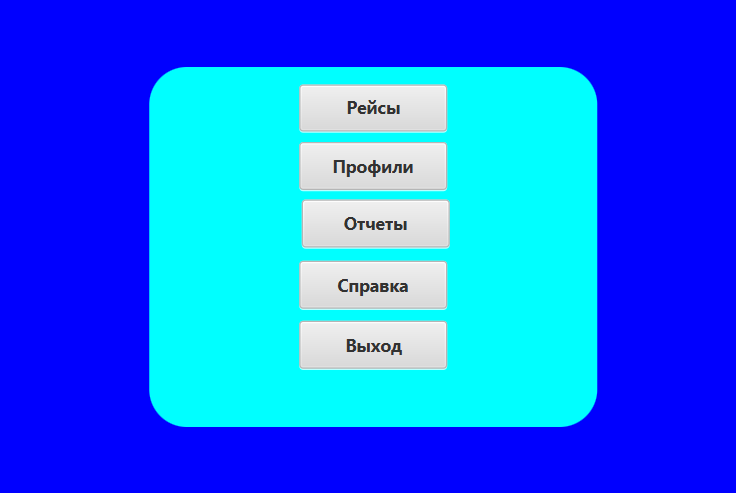


Рисунок 4.2 – Основное меню администратора

При нажатии на кнопку «Рейсы», на экране появится дополнительное меню, в котором предоставляется возможность добавлять, редактировать и удалять рейсы(рисунок 4.3).

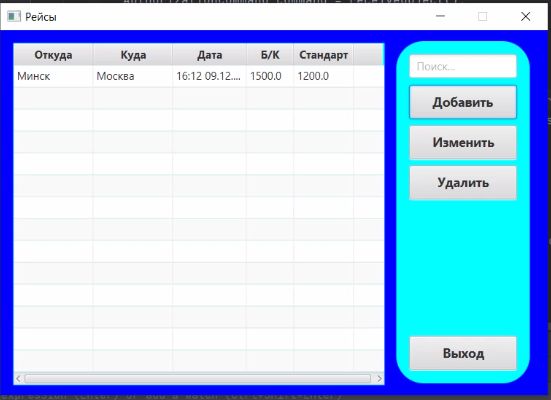


Рисунок 4.3 – Дополнительное меню «Рейсы»

Приведем также пример добавления рейса(рисунок 4.4).

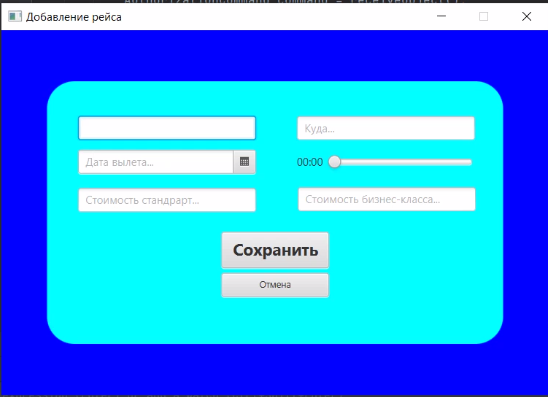


Рисунок 4.4 – Добавление рейса

Для того, чтобы отредактировать данные о рейсе, необходимо нажать кнопку «Изменить»(рисунок 4.5).

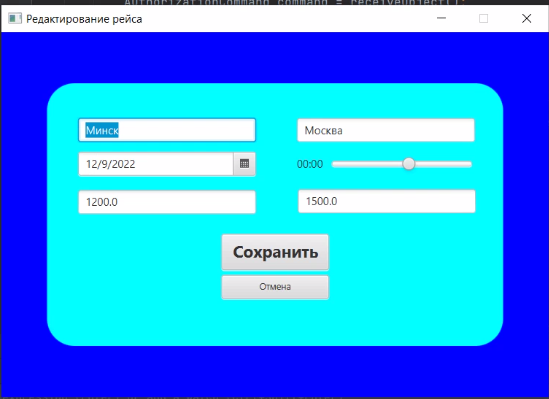


Рисунок 4.5 – Редактирование рейса

Администратору также доступна работа с базой данных пользователей, с возможностью блокировки/разблокировки пользователей(рисунок 4.6).

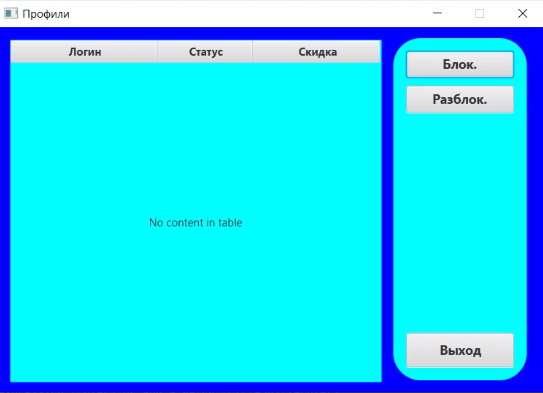


Рисунок 4.6 – База данных пользователей

Администратор с полным доступом прав имеет возможность просматривать отчеты истории покупок(рисунок 4.7).

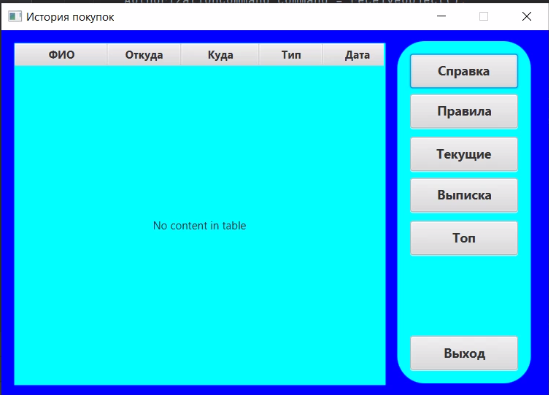


Рисунок 4.7 – История покупок

Для отображения правил полета, необходимо нажать одноименную кнопку «Правила» (см. рисунок 4.8).

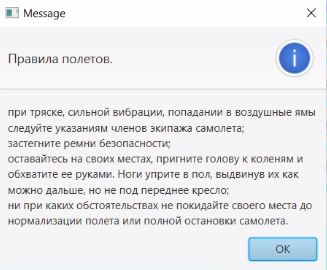


Рисунок 4.8 – Правила полета

Если авторизация пользователя прошла успешно, то откроет основное меню клиента(рисунок 4.9).

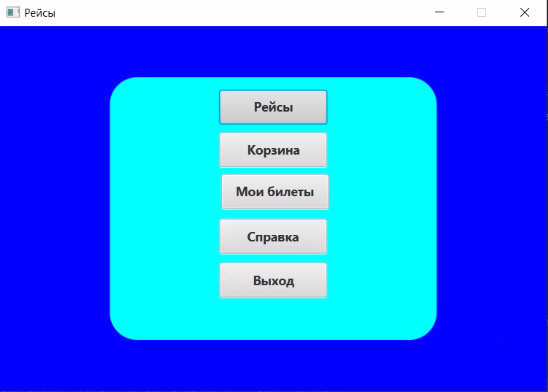


Рисунок 4.9 – Главное меню клиента

При нажатии на первую кнопку, на экране появится список рейсов, с возможностью добавить в корзину(рисунки 4.10-4.11).

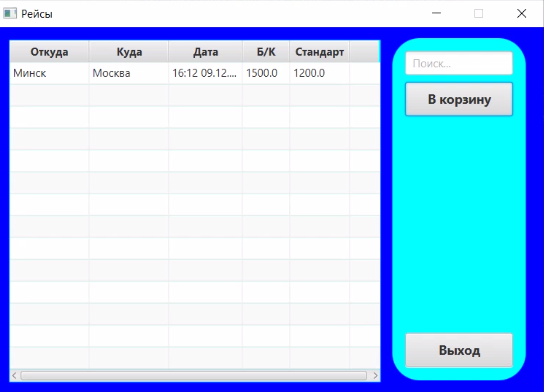


Рисунок 4.10 – Дополнительное меню «Рейсы»

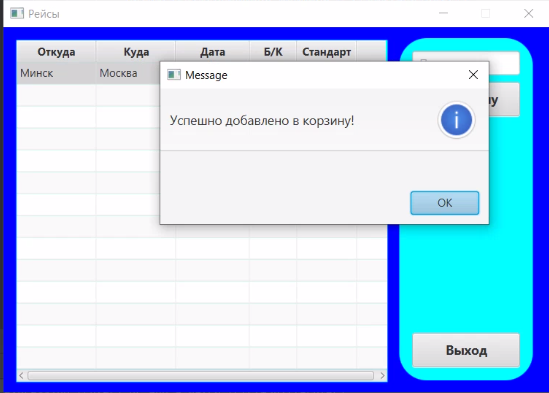


Рисунок 4.11 – Успешное добавление рейса в корзину

Также пользователю предоставляется взаимодействие с «Корзиной». Там есть возможность купить билеты на рейс или удалить их(рисунок 4.12).

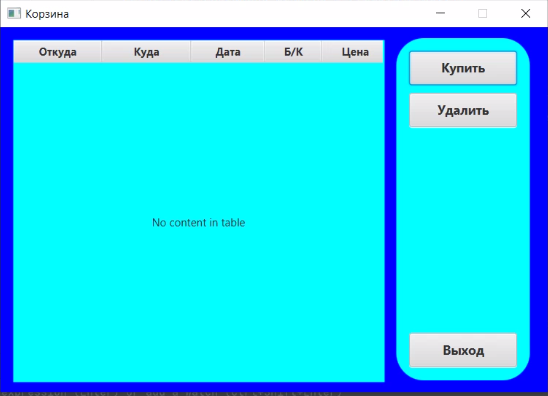


Рисунок 4.12 – Пункт меню «Корзина»

Пункт меню «Мои билеты» предоставляет список билетов с возможностью просматривать правила полетов, выписку, и историю билетов(рисунок 4.13).

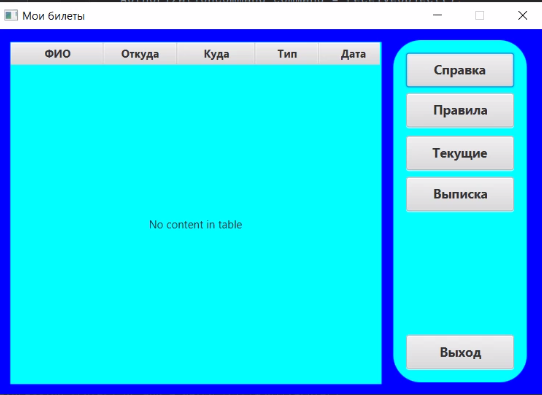


Рисунок 4.13 – Пункт меню «Мои билеты»

Таким образом, выполнены проектирование и разработка программного средства планирования и оценки нагрузки персонала медицинского учреждения. Выполнена постановка задачи и определены основные методы ее решения; в ходе объектного моделирования системы построен ряд *UML*-диаграмм; разработана информационная модель системы, представленная в виде схемы базы данных; описаны основные алгоритмы работы программного средства; разработано руководство пользователя; выполнено тестирование системы.

# Заключение

В результате выполнения данного курсового проекта было разработано программное приложение, позволяющие оптимизировать работу аэропорта. В нем предусмотрена система администрирования, а также проверка учетных данных при авторизации пользователей.

Приложение удовлетворяет основным характеристикам, которые были заявлены. Оно удобно в эксплуатации, целостно, конкретизировано в рамках заданной предметной области.

При разработке данного программного продукта была учтена логика пользователя, и интерфейс данной программы сделан удобным и понятным.

Использование данной программы в реальной аэропорту сделает работу автоматизированной, более быстрой и упрощенной. В дальнейшем данная программа может редактироваться и совершенствоваться в соответствии с требованиями предметной области. Подводя итог, можно сказать, что цели и задачи, поставленные перед данной работой, успешно достигнуты и выполнены.

Выполнено проектирование и разработка программного средства продажи авиабилетов: выполнена постановка задачи и определены основные методы ее решения; в ходе объектного моделирования системы построен ряд *UML*-диаграмм; разработана информационная модель системы, представленная в виде схемы базы данных; описаны основные алгоритмы работы программного средства; разработано руководство пользователя; выполнено тестирование системы, показавшее ее соответствие функциональным требованиям, поставленным в задании на разработку.

# Список использованных источников

1. Хабр. Что внутри аэропорта: центры управления [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://habr.com/ru/company/tuturu/blog/477298/> .
2. Что внутри лучшего регионального аэропорта страны: службы терминала [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://story.tutu.ru/kak-ustroen-ajeroport-iznutri/?ysclid=lbf7jdz4gp453353407> .
3. Как устроен аэропорт [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://tolmachevo.ru/name/pokryshkin/kak-ustroen-aeroport/?ysclid=lbf7mlenoz64168519>.
4. Ашфорд Н. Функционирование аэропорта : справ. пособие / С. В. Маклов. – Лондон: Диалог-Мифи, 1991. – 74 с.
5. Дубейковский, В. И. Эффективное моделирование с AllFusion Process Modeler 4.1.4 и AllFusion PM / В. И. Дубейковский. – Москва: Нолидж, 2011. – 284 с.
6. Леоненков, А. В. Самоучитель UML / А. В. Леоненков. – СПб.: BHV, 2014. – 304 с.
7. Сергеев, В. И., Григорьев, М. Н., Уваров, С. А. Логистика: информационные системы и технологии / В. И. Сергеев, М. Н. Григорьев, С. А. Уваров. – М.: Альфа-Пресс, 2018 – 355 с.
8. Харрингтон Джен Л. Проектирование реляционных баз данных – М: Лори, 2016. – 514 с.
9. UML. Классика CS. 2-е изд./Пер. с англ.; Под общей редакцией проф. С.Орлова – СПб.: Питер, 2006. – 736 с.
10. Басс, Л., Архитектура программного обеспечения на практике. 2-е издание / Л. Басс, П. Клементс, Р. Кацман. – СПб.: Питер, 2006. – 575 с.
11. Фаулер, М. Архитектура корпоративных программных приложений.: Пер. с англ. / М. Файлер. – М.: Издательский дом «Вильямс», 2006. – 544с.
12. Новиков, Ф. Моделирование на UML [Электронный ресурс] / Ф. Новиков, Д. Иванов. – Режим доступа : <http://book.uml3.ru/sec_1_5>
13. Кагарлицкий, Ю.В. Разработка документации пользователя программного продукта. Методика и стиль изложения / Ю. В. Кагарлицкий. – М. : Философт Сервисы, 2008. – 200 с.

# Приложение А

# (обязательное)

## Отчет о проверке на заимствования в системе «Антиплагиат»



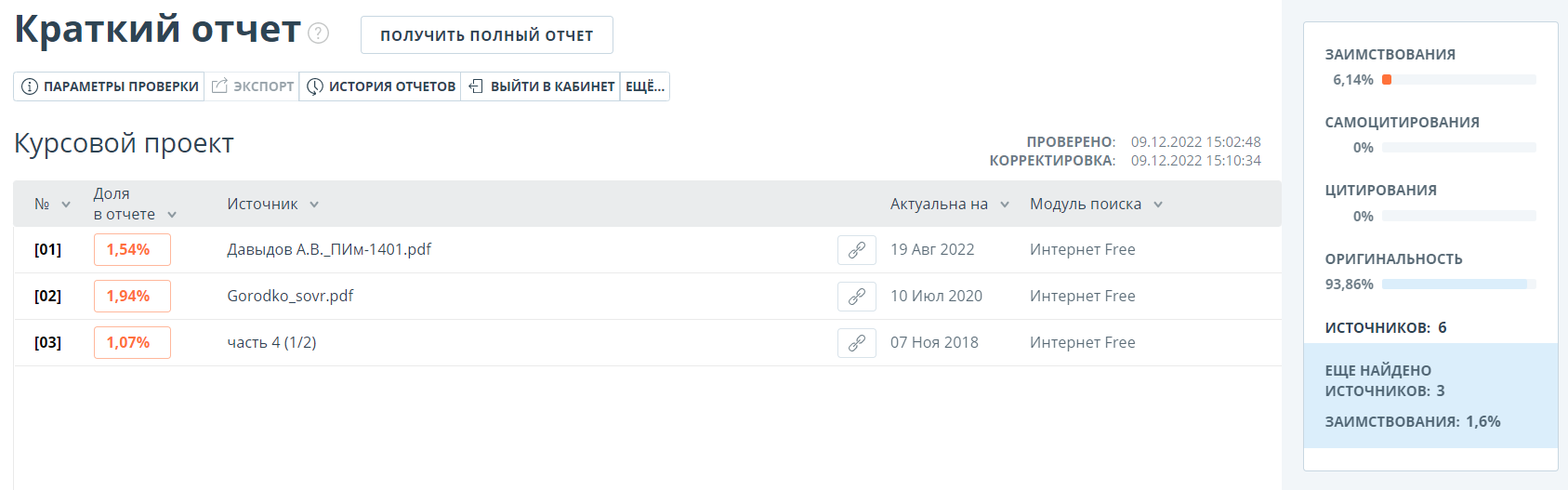


Рисунок А.1 – Отчет о проверке на заимствования в системе «Антиплагиат»

# Приложение Б

# (обязательное)

## Листинг кода алгоритмов, реализующих бизнес-логику

package com.example.cwkarapaeva;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.TableView;

public class TicketsToBuyController {

@FXML

private Button buyButton;

@FXML

private Button delButton;

@FXML

private Button exitButton;

@FXML

private TableView<?> tableRaises;

@FXML

void onClickBuy(ActionEvent event) {

}

@FXML

void onClickDel(ActionEvent event) {

}

@FXML

void onClickExit(ActionEvent event) {

}

}

package com.example.cwkarapaeva;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.TableView;

public class TicketsController {

@FXML

private Button aboutButton;

@FXML

private Button rulesButton;

@FXML

private TableView<?> tableTickets;

@FXML

void onClickAbout(ActionEvent event) {

}

@FXML

void onClickExit(ActionEvent event) {

}

@FXML

void onClickRules(ActionEvent event) {

}

}

# Приложение В

# (обязательное)

## Листинг скрипта генерации базы данных

create schema if not exists `airport\_database`;

use `airport\_database`;

create table if not exists `planes`(

`id` int not null auto\_increment,

`model` varchar(50) not null,

`number` int not null,

`amountOfSeats` int not null,

constraint `PK\_planes` primary key (`id` ASC)

);

create table if not exists `airports`(

`id` int not null auto\_increment,

`name` varchar(50) not null,

`country` varchar(50) not null,

`shortCode` varchar(5) not null,

constraint `PK\_airports` primary key (`id` ASC)

);

create table if not exists `users`(

`id` int not null auto\_increment,

`login` varchar(50) not null,

`password` varchar(50) not null,

`userType` int not null,

constraint `PK\_users` primary key (`id` ASC)

);

create table if not exists `pilots`(

`id` int not null auto\_increment,

`name` varchar(50) not null,

`surname` varchar(50) not null,

`patronymic` varchar(50) not null,

`experience` int not null,

`flightHours` int not null,

`userId` int not null,

constraint `PK\_pilots` primary key (`id` ASC),

constraint `FK\_pilots\_users` FOREIGN KEY(`userId`) REFERENCES `users` (`id`)

);

create table if not exists `flights`(

`id` int not null auto\_increment,

`planeId` int not null,

`pilotId` int not null,

`departureAirportId` int not null,

`arrivalAirportId` int not null,

`departureDateTime` datetime not null,

`arrivalDateTime` datetime not null,

`cost` float not null,

constraint `PK\_flights` primary key (`id` ASC),

constraint `FK\_flights\_planes` FOREIGN KEY(`planeId`) REFERENCES `planes` (`id`),

constraint `FK\_flights\_departureAirport` FOREIGN KEY(`departureAirportId`) REFERENCES `airports` (`id`),

constraint `FK\_flights\_arrivalAirport` FOREIGN KEY(`arrivalAirportId`) REFERENCES `airports` (`id`),

constraint `FK\_flights\_pilots` FOREIGN KEY(`pilotId`) REFERENCES `pilots` (`id`)

);

create table if not exists `persons`(

`id` int not null auto\_increment,

`name` varchar(50) not null,

`surname` varchar(50) not null,

`patronymic` varchar(50) not null,

`passportNumber` varchar(50) not null,

constraint `PK\_persons` primary key (`id` ASC)

);

create table if not exists `tickets`(

`id` int not null auto\_increment,

`flightId` int not null,

`personId` int not null,

constraint `PK\_tickets` primary key (`id` ASC),

constraint `FK\_tickets\_flights` FOREIGN KEY(`flightId`) REFERENCES `flights` (`id`),

constraint `FK\_tickets\_persons` FOREIGN KEY(`personId`) REFERENCES `persons` (`id`)

);

create table if not exists `user\_persons`(

`userId` int not null,

`personId` int not null,

constraint `FK\_user\_persons\_users` FOREIGN KEY(`userId`) REFERENCES `users` (`id`),

constraint `FK\_user\_persons\_persons` FOREIGN KEY(`personId`) REFERENCES `persons` (`id`)

);

create table if not exists `user\_tickets`(

`userId` int not null,

`ticketId` int not null,

constraint `FK\_user\_tickets\_users` FOREIGN KEY(`userId`) REFERENCES `users` (`id`),

constraint `FK\_user\_tickets\_tickets` FOREIGN KEY(`ticketId`) REFERENCES `tickets` (`id`)

);

# Ведомость документов курсового проекта