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  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.М. Унучек |
|  | \_\_\_.\_\_\_\_.2021 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«Автоматизированная система продажи авиационных билетов»**

БГУИР КП 1-40 05 01-10 005 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 914301  Гулевич Виктория Андреевна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2021  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2021

**РЕФЕРАТ**

БГУИР КР 1-40 05 01 10 005 ПЗ

Гулевич, В. А. Автоматизированная система продажи авиационных билетов

В. А. Гулевич – Минск: БГУИР, 2021. – с. 49

Пояснительная записка 49 с., 26 рис., 6 источников, 2 приложения

ПРОТОКОЛ TCP/IP, ПРИЛОЖЕНИЕ КЛИЕНТ – СЕРВЕР, ПРОДАЖА АВИАЦИОННЫХ БИЛЕТОВ, БАЗА ДАННЫХ

Предмет: авиационные билеты.

Объект: сетевой протокол TCP/IP, работающий по принципу клиент-сервер.

Цель: разработка графического пользовательского интерфейса для покупки и бронирования авиационных билетов по протоколу TCP/IP.

Методология проведения работы: в процессе решения поставленных задач был рассмотрен принцип взаимодействия сетевого протокола TCP/IP, который ориентирован на обмен сообщений между устройствами. Также были использованы принципы объектно-ориентированного программирования и реализация графического интерфейса.

Результаты работы: выполнено изучение работы протокола передачи данных, основанного по принципу «клиент-сервер», изучен ORM фреймворк Hibernate, который необходим для работы с базами данных. Разработан алгоритм взаимодействия клиента, подключающегосяся к серверу, спроектирован пользовательский графический интерфейс для обмена информацией и командами по протоколу, который предназначен для передачи данных между устройствами.

Область применения полученных результатов: в концепции Интернет вещей, для взаимодействия «умных» гаджетов, техники и сетевых устройств в сети.

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ6

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ8

2 ПОСТАНОВКА ЗАДАЧИ И МЕТОДЫ ЕЕ РЕШЕНИЯ12

3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF016

4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЕ ОПИСАНИЕ22

5 ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ25

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ27

6.1 Алгоритм работы пользователя27

6.2 Алгоритм работы администратора32

7 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ35

ЗАКЛЮЧЕНИЕ38

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ39

ПРИЛОЖЕНИЕ А (обязательное) Листинг скрипта базы данных40

ПРИЛОЖЕНИЕ Б (обязательное) Листинг основных элементов42

ПРИЛОЖЕНИЕ В (обязательное) Графический материал48

# **ВВЕДЕНИЕ**

Широкое применение интернет-технологий становится одной из актуальнейших задач в индустрии туризма. Создание мощных компьютерных систем бронирования средств размещения и транспорта, экскурсионного и культурно-оздоровительного обслуживания, информация о наличии и доступности тех или иных туров, маршрутов, туристского потенциала стран и регионов — весь комплекс этих вопросов становится актуальным для организации текущей и будущей деятельности туристских предприятий.

Использование современных разработок позволяет клиенту работать в системе бронирования с любого компьютера, имеющего доступ к Интернету и оснащенного стандартным программным обеспечением MS Internet Explorer.

Аэропорт – транспортное предприятие, обеспечивающее регулярные перевозки пассажиров, грузов и почты средствами воздушного транспорта. Данная структура на физическом уровне включает в себя целый комплекс специализированных сооружений, призванных обеспечивать прием, отправку, базирование воздушных судов и предоставление наземных услуг контрагентам, а также организацию пассажиропотока и прямое обслуживание клиентов авиакомпаний .

Авиаперевозки играют существенную роль в мировом хозяйственном процессе. Данный вид транспорта является одним из весомых факторов, которые помогают стимулировать мировую торговлю и производственные процессы. Для эффективного функционирования информационной системы в крупных регионах необходима устойчивая и развитая база, которая должна удовлетворять текущие и потенциальные потребности. Автоматизация бизнес-процессов на предприятиях – одно из современных направлений в информационном обществе.

Оптимальное использование аэропортовой инфраструктуры является значимым аспектом развития мировой авиации. Опираясь на современные исследования в области авиаперевозок можно заметить, что в крупных регионах количество аэропортов достаточно велико, однако уровень обслуживания нельзя считать качественным. При этом большую значимость имеет эффективность управления информационной системой. Следовательно, для повышения работоспособности аэропортовой инфраструктуры необходима автоматизация системы. Данное направление будет способствовать не только увеличению рентабельности и снижению эксплуатационных затрат, но и повышению производительности системы в целом, которая обеспечит принятие оптимальных решений по управлению.

Аэропорты всего мира активно внедряют информационные технологии, реализуют масштабные программы по повышению качества обслуживания всех категорий пассажиров и авиакомпаний, а также комплекс мер по обеспечению безопасности полётов. Повышение эффективности и оперативности обслуживания авиапассажиров и их багажа лежит в основе модернизации информационных систем аэропорта. При этом актуальным остаётся процесс автоматизации аэропортовых формальностей – развитие технологии обслуживания, при котором функции управления и контроля, ранее выполнявшиеся человеком, передаются приборам и автоматическим устройствам .

Для производительности любого предприятия необходимо наличие развитой системы, которая способна осуществить автоматизированный сбор, хранение, обработку и управление данными. Также данная система должна включать в себя технические средства обработки данных, программное обеспечение и обслуживающий персонал.

Целью курсового проекта является повышение эффективности работы аэропорта, что можно достичь автоматизацией системы продажи авиабилетов

Задачи курсового проекта:

– провести описание предметной области;

– разработать функциональную модель на основе стандарта IDEF0;

– составить описание информационной модели (приведение к 3-ей нормальной форме);

– составить руководство пользователя;

– проанализировать результаты тестирования автоматизированной системы.

# **1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

Электронный билет или e-ticket — это электронный документ, удостоверяющий договор воздушной перевозки между пассажиром и авиакомпанией. В отличие от бумажного авиабилета, электронный билет представляет собой цифровую запись в базе данных авиакомпании.

Технологии применения традиционных бумажных авиабилетов создавались в первой половине XX века с учетом возможностей применявшихся в то время технологий хранения, передачи и обработки информации. В частности, для хранения информации применялись бумажные носители, для передачи - телеграфные и телефонные голосовые каналы, для поиска информации большие хранилища бумажных носителей просматривались глазами. Используемые технологии не позволяли проводить в массовом порядке за приемлемо короткое время даже выборочную проверку авиабилетов, предъявляемых пассажирами для получения услуг перевозки или для возврата денег при отказе от перевозки. Таким образом, выписанный авиабилет должен был быть сам по себе носителем полной информации, которая необходима для предоставления услуг и соответствующих денежных расчетов. Вместе с тем, на различных стадиях процесса продажи билетов и перевозки пассажира теоретически возможны намеренные злоумышленные действия по фальсификации информации о заключенных договорах перевозки с целью незаконного получения какой-либо финансовой выгоды. (Например, в законно купленном билете исправлялась информация о маршруте и цене билета на более высокую, после чего такой билет предъявлялся к возврату. Также известны случаи оформления поддельных билетов на похищенных бланках и другие виды махинаций). В результате этих и многих других факторов были выработаны специальные меры по проведению процедур продажи авиабилетов. В частности, существовали следующие ограничения:

1. Авиабилет должен быть выпущен на специальном бланке, который снабжен средствами защиты от подделок (например, бумага с водяными знаками).

2. Бланки авиабилетов должны быть бланками строгой отчетности, в отношении которых действуют специальные правила учета, хранения, передачи для использования, утилизации.

3. Процедура выпуска авиабилета должна была подразумевать копирование информации с билета на другие носители информации (например, отрывные купоны билета) и передачу этих носителей в другие предприятия для учета, использования, приема к исполнению, архивирования и других целей.

Все это делало процедуру выпуска авиабилета достаточно дорогостоящей. Есть сведения, что затраты авиакомпаний на выпуск бланков билетов и другие затраты, связанные собственно с организацией продажи билетов на бумажных бланках, достигали 10 долларов за каждый проданный билет.

В конце XX века ситуация с хранением, передачей, обработкой и поиском информации существенно изменилась. Существенно возрасли скорость передачи и обработки информации, а также емкость информационных хранилищ и скорость поиска информации. Все это сделало возможным появление технологии электронных билетов. В отличие от бумажного билета, электронный билет представляет собой совокупность записей в базах данных различных компьютерных систем. Для электронного билета не требуется специальный бланк - вся информация хранится в памяти компьютеров и при необходимости быстро передается с одного компьютера на другой. Соответственно, снижаются затраты авиакомпаний на изготовление бланков и их обращение. Далее, с развитием средств электронной коммерции и электронных платежей появляется возможность оплаты авиабилета с помощью автоматизированных устройств (банкоматы, платежные терминалы, банковские компьютерные системы). Таким образом, сокращаются затраты авиакомпаний и агентов по продаже билетов на организацию денежного оборота. Кроме того, внедрение технологий электронного билета позволяет снизить и некоторые другие затраты.

Электронный билет можно самостоятельно купить в интернете на специальных сайтах. Преимущества интернет-покупки билета заключаются в том, что бронирующий получает полную информацию о всех возможных перелётах, предложенных авиакомпаниями на данном маршруте, а также видит структуру образования цены авиабилета,с указанием платы за саму перевозку и сборов аэропорта. Покупка производится гораздо быстрее стандартной покупки через турбюро. Покупка и оплата электронного билета осуществляется на сайтах, которые подключены к ресурсам продаж билетов авиакомпаний. Для оплаты принимается кредитная карта. Подтверждение о покупке высылается на электронный адрес.

Бронирование электронных билетов может осуществляться как традиционным способом (через операторов и компьютерные системы бронирования), так и самостоятельно пассажирами через интернет-сайты авиакомпаний и агентств. Оплата билета также может происходить как традиционным способом (наличными деньгами кассиру), так и с помощью средств электронных платежей.

Регистрация пассажиров на рейс производится через компьютерную систему управления отправками пассажиров, установленную в аэропорту. Предварительно в эту систему передается из систем бронирования информация о выпущенных билетах (электронных и бумажных). При регистрации пассажира по электронному билету предъявлять маршрут-квитанцию не обязательно - вся необходимая информация уже есть в системе. При регистрации вне зависимости от типа билета пассажиру выдается посадочный талон, который является основным документом для пропуска пассажира в зону предполетного досмотра и посадки в самолет. Такой подход унифицирует работу сотрудников аэропорта на этапе от регистрации пассажира до его посадки в самолет.

Аэропортовая инфраструктура осуществляет различную категорию перевозок. При этом весомую часть перевозимого груза составляют товары повседневной необходимости: продукты питания, электроника, одежда и т.д. Однако особое внимание концентрируется на пассажирских перевозках. Следовательно, данная категория является одним из важных факторов, которые стимулируют мировую экономику. Авиаперевозки облегчают жизнь людей и положительно сказываются на общем экономическом росте.

Корректная работа аэропортовой структуры способствует эффективному взаимодействию с пассажирами и посредниками. При этом планирование данного процесса должно включать структурированный анализ различных областей. Для слаженной работы аэропортовой системы необходимо определить осуществляемые услуги.

Аэропорт предоставляет услуги по авиационной и неавиационной деятельности. Первая категория, в свою очередь, подразделяется на не аэропортовые и аэропортовые услуги. Не аэропортовые услуги связаны с:

* продажей пассажирских авиаперевозок;
* продажей грузовых авиаперевозок;
* организационным обеспечением полётов;
* техническое обслуживание и ремонт воздушных судов и двигателей;
* материально-техническое снабжение запасными авиачастями.

Аэропортовые услуги связаны с использованием аэродромов, в том числе предоставлением: ВВП и аэродромного оборудования, мест стоянок, ангара, аэронавигационного обслуживания в зоне ответственности аэродрома, услуг САБ аэродрома, аэровокзала пассажирам.

Следующая группа подразделяется на оказываемые авиакомпаниями содействующим организациям и оказываемые организациям и частным лицам услуги.

Услуги, осуществляемые авиакомпаниями связаны с наземным обслуживанием:

* общее руководство и работа с экипажем;
* пассажиров, ручной клади и багажа, включая досмотр;
* наружную и внутреннюю уборку;
* топливообеспечение;
* противообледенительную обработку перед вылетом;
* обеспечение бортпитанием.

Другая категория связана с предоставлением в аренду зданий и сооружений, помещений и рабочих площадей, территорий, автостоянок, каналов связи, информационных систем, технологического оборудования для выполнения наземного обслуживания.

Выделяют следующие цели развития:

1. Создание своевременного транзитного комплекса, обеспечивающего

качественное обслуживание пассажирских авиаперевозок и увеличение транзита грузов;

1. Увеличение экспорта авиационных услуг;
2. Развитие неавиационных видов деятельности, связанных с

оказанием услуг пассажирам;

1. Получение дополнительных объёмов доходов от неавиационных

видов деятельности, необходимых для дальнейшей модернизации, технического оснащения производства;

1. Соответствие международным стандартам и требованиям;
2. Развитие инфраструктуры аэропорта и приаэропортовой

территории.

Достижение данных целей смогут обеспечить слаженную и эффективную работу всех сфер аэропортовой структуры.

В сложившихся условиях решающим фактором выживания для аэропортовой инфраструктуры является возможность оптимизации применяемых бизнес-решений при сохранении и даже повышении безопасности полётов.

Обеспечение безопасности и эффективности работы аэропорта является главной целью развития программных обеспечений, которые в свою очередь, будут удовлетворять внешним требованиям системы.

**2 ПОСТАНОВКА ЗАДАЧИ И МЕТОДЫ ЕЕ РЕШЕНИЯ**

Автоматизация – одно из направлений научно-технического прогресса, использующее саморегулирующие технические средства и математические методы с целью освобождения человека от участия в процессах получения преобразования, передачи и использования энергии, материалов, изделий или информации, либо существенного уменьшения степени этого участия или трудоемкости. Автоматизированная система продажи авиабилетов – это комплекс операций, которые предназначены для поиска, добавления, редактирования, покупки или бронирования билетов .

Разрабатываемая система должна содержать в себе следующие подсистемы:

* подсистема администрирования, позволяющая осуществлять поддержку системы;
* клиентская подсистема, позволяющая просматривать справочную информацию и отправлять запросы на бронирование или возврат авиабилетов.

Для доступа к любой из данных подсистем пользователь должен пройти предварительную регистрацию или, если пользователь уже зарегистрирован, авторизацию. Разным группам пользователей доступны разные функциональные возможности и уровень доступа к информации. В ходе данной курсового проекта должна быть создана автоматизированная информационная система продажи авиабилетов, решающая следующие задачи:

* продажа авиабилетов на запланированные рейсы;
* поиск рейса по запросу пользователя ;
* администрирование информационной системы;
* создание приложения, предоставляющего пользователям графический интерфейс для доступа к системе.

Информационная система продажи авиабилетов имеет ряд специфических особенностей, которые предусматривают наличие надежной защиты информации, обеспечивающей сохранность данных аэропорта.

Перспективы развития аэропортовой инфраструктуры, а также ее состояние являются ключевыми элементами международного хозяйства. Аэропорт способен выполнять функцию интегрирующей и социально ориентированной инфраструктуры экономики страны.

Автоматизация информационной системы достаточно сложный процесс. Он предполагает задействовать все сферы деятельности данной инфраструктуры.

Для оптимизации кода, который позволит создать автоматизированную систему продажи авиационных билетов, в данном курсовом проекте будут использоваться паттерны прооектирования.

Шаблон проектирования или паттерн в разработке программного обеспечения – повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Обычно шаблон не является законченным образцом, который может быть прямо преобразован в код; это лишь пример решения задачи, который можно использовать в различных ситуациях. Объектно-ориентированные шаблоны показывают отношения и взаимодействия между классами или объектами, без определения того, какие конечные классы или объекты приложения будут использоваться.

«Низкоуровневые» шаблоны, учитывающие специфику конкретного языка программирования, называются идиомами. Это хорошие решения проектирования, характерные для конкретного языка или программной платформы, и потому не универсальные.

На наивысшем уровне существуют архитектурные шаблоны, они охватывают собой архитектуру всей программной системы.

Алгоритмы по своей сути также являются шаблонами, но не проектирования, а вычисления, так как решают вычислительные задачи.

Есть три основных типа шаблонов проектирования:

* структурный;
* порождающий;
* поведенческий.

Структурные шаблоны, в общем случае, имеют дело с отношениями между объектами, облегчая их совместную работу.

Порождающие шаблоны обеспечивают механизмы инстанцирования, облегчая создание объектов способом, который наиболее соответствует ситуации.

Поведенческие шаблоны используются в коммуникации между объектами, делая её более лёгкой и гибкой.

Важно знать не только удачные способы решения задач, но и возможные ошибки при их решении. Для этого используют антипаттерны проектирования.

Антипаттерн – ­это подход к решению задач, который является неэффективным. Рассмотрение антипаттерна включает в себя не только неправильное решение проблемы, но и нахождение правильного решения.

В курсовом проекте использованы паттерны MVC, Singletone и DAO.

Паттерн MVC расшифровывается как "Модель - представление - контроллер" . Эта модель представляет данные приложения и связанную с ними бизнес-логику. Модель может быть представлена одним объектом или сложным графом связанных объектов. В приложении для плат­формы Java ЕЕ данные инкапсулируются в объектах предметной области, часто раз­вертываемых в EJB-модуле. Данные передаются в БД и из нее в объектах передачи данных (ОТО), и к ним обращаются с помощью объектов доступа к данным (ОАО).

Представление – это наглядное отображение содержащихся в модели данных. Подмножество модели содержится в отдельном представлении, таким образом, представление действует в качества фильтра для данных модели. Пользователь взаимодействует с данными модели с помощью предлагаемого представлением наглядного отображения и обращается к бизнес логике, которая, в свою очередь, воздействует на данные модели.

Контроллер связывает представление с моделью и управляет потоками данных приложения. Он выбирает, какое представление визуализировать для пользовате­ля в ответ на вводимые им данные и в соответствии с выполняемой бизнес-логикой. Контроллер получает сообщение от представления и пересылает его модели. Мо­дель, в свою очередь, подготавливает ответ и отправляет его обратно контроллеру, где происходит выбор представления и отправка пользователю.

Паттерн MVC логически охватывает клиента и промежуточный уровень мно­гоуровневой архитектуры. В среде Java ЕЕ модель располагается в бизнес-слое, обычно в виде ЕJВ-модуля.

Часто MVC сочетается с другими паттернами, такими как "Команда" (или "Действие"), "Стратегия", "Компоновщик" и "Наблюдатель".

Теперь речь пойдет про такой паттерн как Singletone.

Синглтон (singleton) это класс, у которого экземпляр создаётся только один раз. Хорошим примером такого поведения служит файловая система, система работы с видео и т.д. В Android можно привести примеры с классами OkHttpClient, Retrofit, Gson, SharedPreferences. Для реализации синглтона нужно создать закрытый конструктор и открытый статический член, который и позволяет получить доступ к единственному экземпляру класса.

Область применения сингтона:

* В системе должно существовать не более одного экземпляра заданного класса;
* Экземпляр должен быть легко доступен для всех клиентов данного класса;
* Создание объекта on demand, то есть, когда он понадобится первый раз, а не во время инициализации системы.

В подавляющем большинстве случаев в Singleton классах наследование не нужно и, более того, излишне и является следствием over-design. Да и реализация наследования имеет определенные сложности, учитывая, что и сам instance и метод getInstance() статические. Поэтому, рекомендуется использовать модификатор final и запретить наследование данного класса, если нет особой необходимости в обратном.

Еще одним подводным камнем является то, что каждая виртуальная машина создает свою копию Singleton объекта. И хотя на первый взгляд это выглядит очевидным, во многих распределенных системах, таких как EJB, JINI и RMI все не так просто. Когда промежуточные уровни скрывают (делают прозрачными) распределенные технологи, бывает трудно сказать, где и когда инициализирован объект.

Еще один паттерн, который применяется в данном курсовом проекте это DAO. Шаблон Data Access Object (DAO) является структурным шаблоном, который позволяет изолировать прикладной/бизнес-уровень от постоянного уровня (обычно это реляционная база данных , но это может быть любой другой постоянный механизм) с использованием абстрактного API .

Функциональность этого API заключается в том, чтобы скрыть от приложения все сложности, связанные с выполнением операций CRUD в базовом механизме хранения. Это позволяет обоим слоям развиваться отдельно, ничего не зная друг о друге.

# **3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА**

# **ОСНОВЕ СТАНДАРТА IDEF0**

Методология функционального моделирования IDEFO – это тех­нология описания системы в целом как множества взаимозависимых действий, или функций. Важно отметить функциональную направ­ленность IDEFO – функции системы исследуются независимо от объ­ектов, которые обеспечивают их выполнение. «Функциональная» точ­ка зрения позволяет четко отделить аспекты назначения системы от аспектов ее физической реализации. IDEFO сочетает в себе небольшую по объему графическую со строгими и четко определенными рекомендациями, в совокупности предназначенными для построения качественной и понятной модели системы.

Каждая модель должна иметь контекстную диаграмму верхнего уровня, на которой объект моделирования представлен единственным блоком с граничными стрелками. Эта диаграмма называется А-0. Стрелки на этой

Главный процесс содержит моделирующие каналы, которые передают данные от источника к потребителю. При этом данная диаграмма содержит 4 вида каналов:

– входная стрелка;

– выходная стрелка;

– управляющая стрелка;

– стрелка механизма (рисунок 3.1).

На верхнем уровне моделирования системы аэропорта информация, которая подаётся на вход, является основополагающей. Данный класс стрелок отображает данные, которые преобразуются функцией. Этими данными являются: личные данные клиента,доступные рейсы.

Благодаря корректной семантике разработанного первого уровня модели появилась возможность получить полное, достоверное описание системы. При формулировании названия диаграммы необходимо учитывать, что будет отображаться перечень вопросов, которые необходимо решить данной системы.

Наиболее важные свойства объекта выявляются на верхнем уровне иерархии, по мере декомпозиции функции верхнего уровня и разбиения ее на подфункции эти свойства уточняются. Каждая подфункция декомпозируется на элементы следующего уровня. Декомпозиция происходит до тех пор, пока не будет получена релевантная структура, позволяющая ответить на вопросы, сформулированные в цели моделирования.

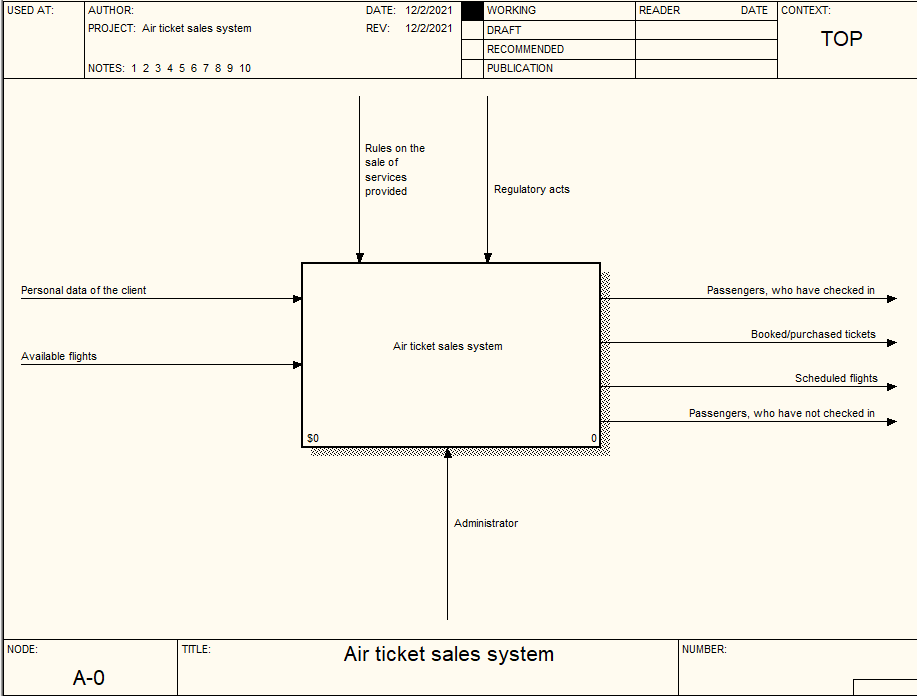


Рисунок 3.1 – Система продажи авиабилетов

Как упоминалось ранее, для полного определения перечня вопросов и структуры необходимо провести декомпозицию главного процесса. Для приобретения более чётких и подробных описаний проектируемой системы необходимо осуществить декомпозицию верхнего уровня диаграммы (рисунок 3.2). Особое внимание следует уделить при моделировании данного уровня диаграмм связыванию функций стрелками для решения обозначенных в цели моделирования задач.

Следующая диаграмма отображает основные цели, преследуемые системой. Так как автоматизация позволяет повысить [производительность труда](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D1%82%D1%80%D1%83%D0%B4%D0%B0), улучшить [качество продукции](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%86%D0%B8%D0%B8), [оптимизировать](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B5) процессы управления, следовательно необходимо автоматизировать основные процессы.

В данной модели необходимо автоматизировать основной из процессов «Бронирование или покупка билетов», где входными параметрами являются личные данные клиента. В свою очередь, это повысит спрос на рынке, так как система будет оснащена удобным интерфейсом для пользователя.

Далее автоматизации подлежит процесс поиска рейсов. Это необходимо для корректного использования ресурсов. При этом необходимо учитывать доступные рейсы, что является информацией на входе.

Последний функциональный блок «Автоматизировать регистрацию пользователя». Выходной информацией данного блока являются пассажиры, которые прошли регистрацию успешно, а также непрошедшие данный этап.

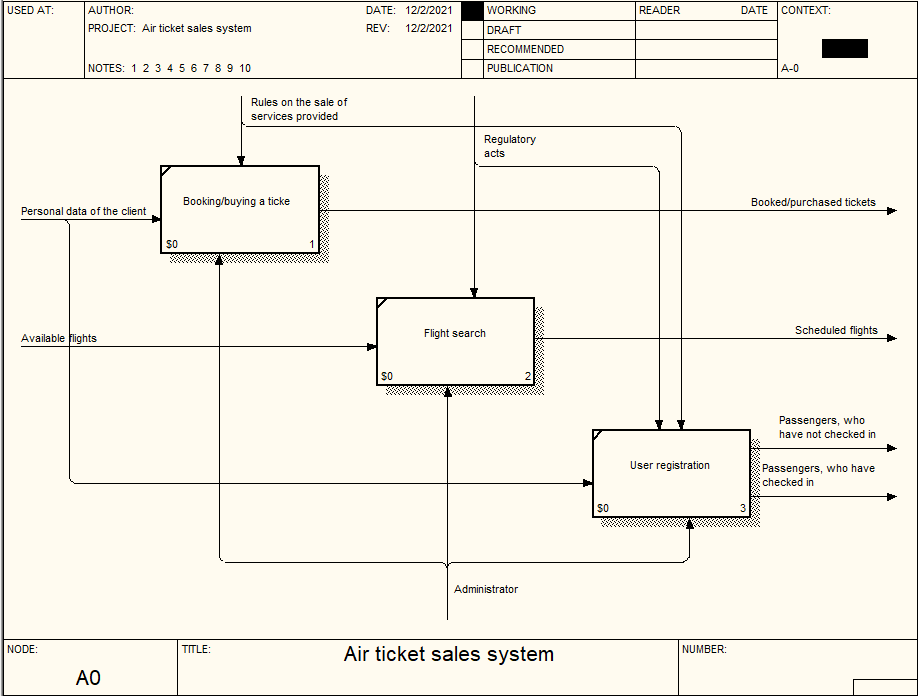


Рисунок 3.2 – Диаграмма декомпозиции системы продажи авиабилетов

Функциональный блок «Забронировать билет» является доминирующим, так как определяет действие остальных функций и выполняется в первую очередь.

По информационным запросам пользователя, поступающим в блок «Просмотреть билет», формируется параметрический запрос просмотр уже забронированного билета. Сформированные данные оплаченного билета поступают на вход блока «Оплатить билет». На основе этого формируется сообщение о проданных билетах. Диаграмма декомпозиции блока «Автоматизация бронирования билета» отображен на рисунке 3.3.

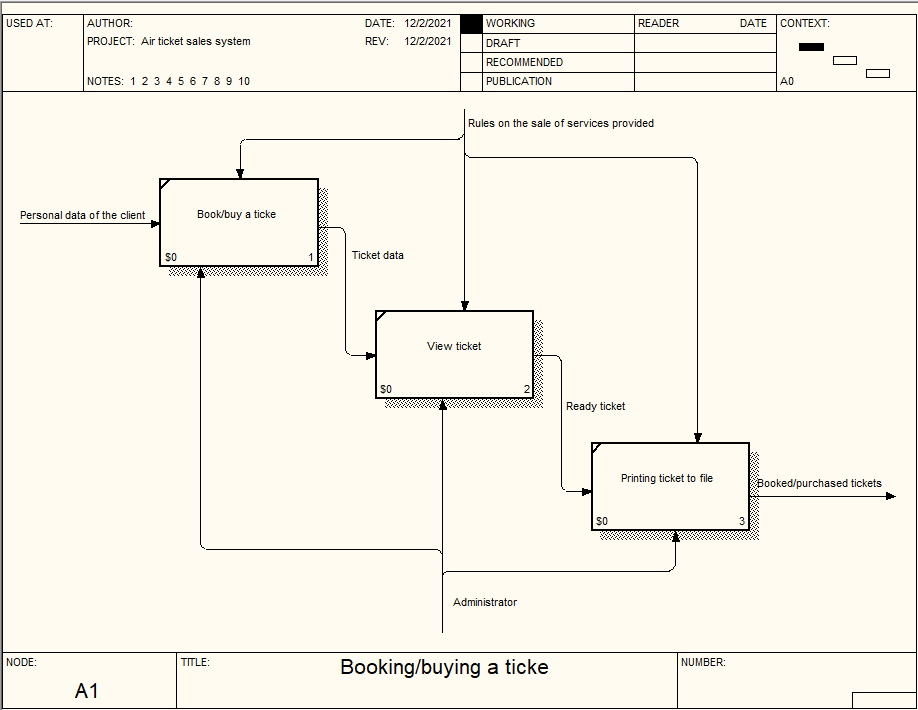


Рисунок 3.3 – Диаграмма декомпозиции блока бронирования/покупки билета

Главным блоком в данной диаграмме является «Заключить договор». Управление данным процессом осуществляет администратор, который составляет статистику предложенных вариантов от авиакомпаний. После заключения договора необходимо составить план рейсов, которые будут осуществляться в намеченных направлениях. Следующий блок «Осуществить план рейсов» является заключительным. На вход подаётся информация об утверждённом расписании данных рейсов. Выходной информацией является запланированные рейсы, которые утвердили сотрудники данной организации (рисунок 3.4).

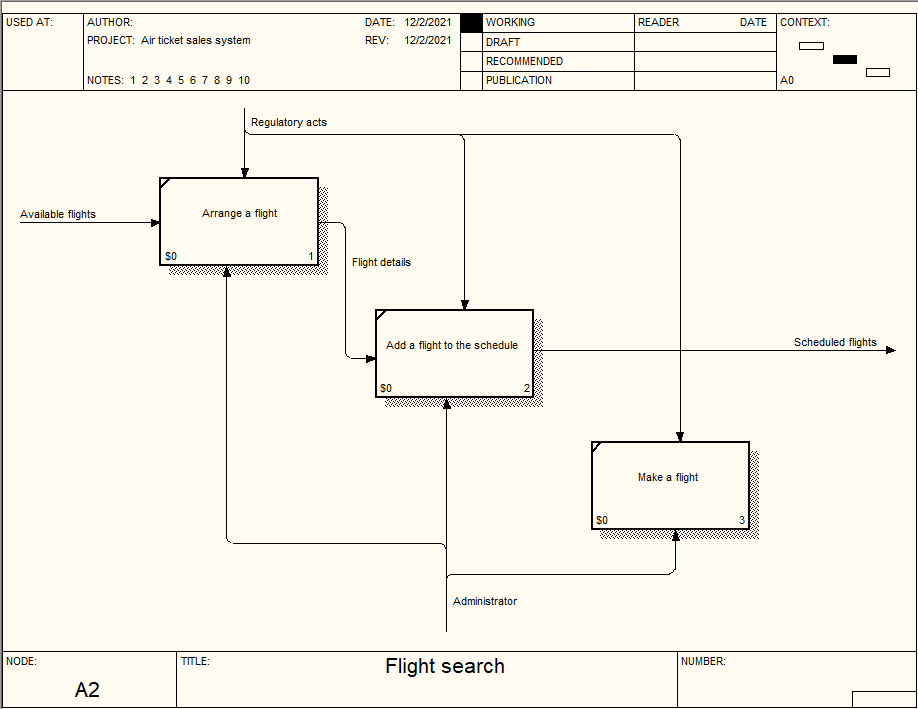


Рисунок 3.4 – Декомпозиция блока составления рейсов

Проектирование автоматизированной информационной системы имеет ряд особенностей. Они связаны с наличием в большинстве информационных систем базы данных, системы аутентификации (идентификации пользователя), системы взаимодействия с пользователем посредством интерфейсов или сообщений. Кроме этого, подавляющее большинство информационных систем предназначено для формирования отчетов по информационным запросам пользователей.

Поэтому были включены в модель автоматизированной информационной системы следующие функциональные блоки «Зарегистрировать пользователя», «Провести идентификацию пользователя», «Осуществить посадку» (рисунок 3.5).

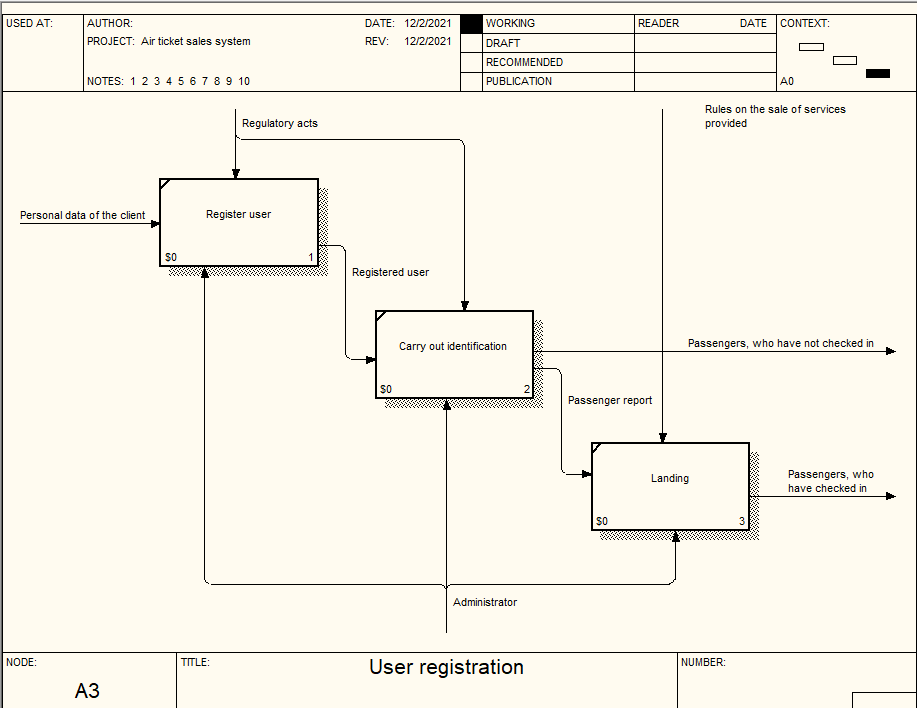


Рисунок 3.5 – Диаграмма декомпозиции блока регистрации пользователя

Следующая диаграмма моделирует регистрацию пользователя в сети. Пассажиру необходимо зарегистрироваться и авторизоваться в системе, чтобы осуществить бронирование билета. При этом пользователю необходимо ввести уникальный логин, который будет являться основным при регистрации. После подбора логина, пользователю необходимо обеспечить безопасный вход в базу. Пароль вводится пользователем для завершения регистрации. Если регистрация завершена успешно, то пользователь добавляется автоматически в базу данных с уникальным логином и паролем (рисунок 3.6).

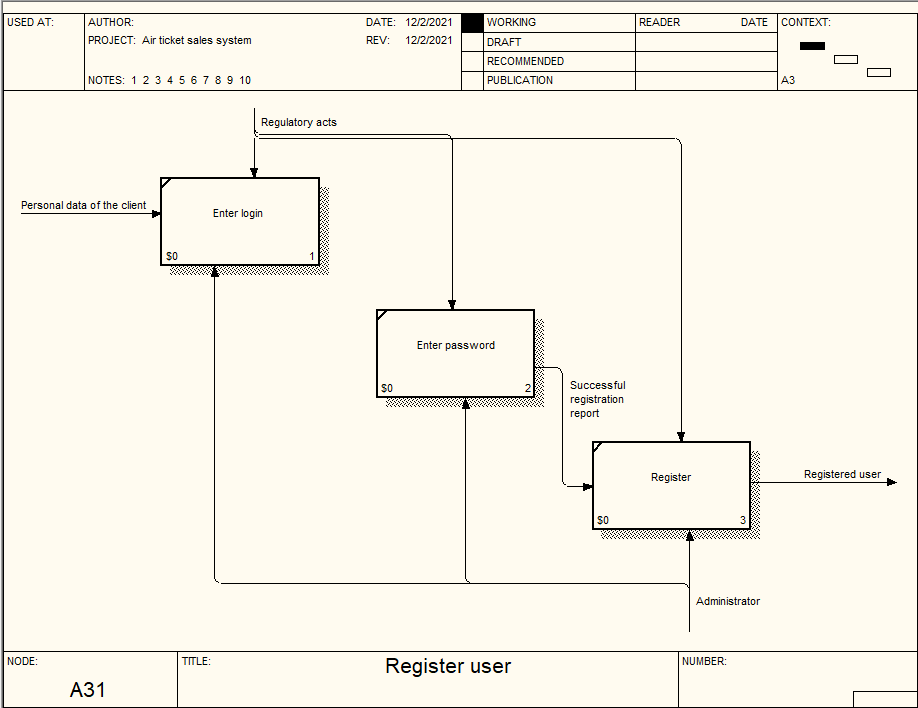


Рисунок 3.6 – Декомпозиция блока регистрации пользователя

Целью построение функциональной модели является описание всех необходимых процессов с точностью, достаточной для однозначного моделирования деятельности системы.

Одним из положительных результатов построения функциональ­ной модели оказывается прояснение границ моделирования системы в целом и ее основных компонентов. Хотя и предполагается, что в процессе работы над моделью будет происходить некоторое измене­ние границ моделирования, их вербальное (словесное) описание должно поддерживаться с самого начала для обеспечения координа­ции работы участвующих в проекте аналитиков. Как и при определе­нии цели моделирования, отсутствие границ затрудняет оценку степе­ни завершенности модели, поскольку границы моделирования имеют тенденцию к расширению с ростом размеров модели.

# **4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЕ ОПИСАНИЕ**

Информационная система – это коммуникационная и вычислительная система по сбору, хранению, обработке и передаче информации, снаб­жающая работников различного ранга той информацией, которая необходима им для реализации функций управления.

Так же информационная модель системы отражает информацию о предметной области, так называемой части реального мира, данные о которой должны храниться в проектируемой базе данных. В данном случае предметной областью является система продажи авиабилетов. При этом информация, необходимая для описания информационной системы, зависит от типа самой инфраструктуры.

Для того, чтобы обеспечить минимальную избыточность данных и физического объёма данных, а также ускоренный доступ, необходимо привести информационную модель к нормальной форме. При этом система называется нормализованной только в том случае, если она способна удовлетворять следующим требованиям: надежное хранение и обновление данных. Это поможет сократить риск потери данных или их искажения при внесении в БД.

Для моделирования системы используются три этапа проектирования:

– концептуальное проектирование;

– логическое проектирование;

– физическое проектирование;

Концептуальное (инфологическое) проектирование – анализ предметной области и ее описание. Этот этап осуществляется без ориентации на какие-либо конкретные программные или технические средства .

Даталогическое (логическое) проектирование – описание логической структуры данных средствами системы управления базами данных (СУБД), для которой проектируется БД. Описание данной модели основывается на построении концептуальной модели. Даталогическое проектирование включает в себя:

− описание таблиц;

− описание связей между таблицами;

− описание атрибутов.

На данном этапе проектирования информационной модели осуществляется приведение базы к нормальным формам, что позволяет на этапе физического проектирования осуществить коррекциюю работы всей проектируемой системы. Модель системы изображена на рисунке 4.1:

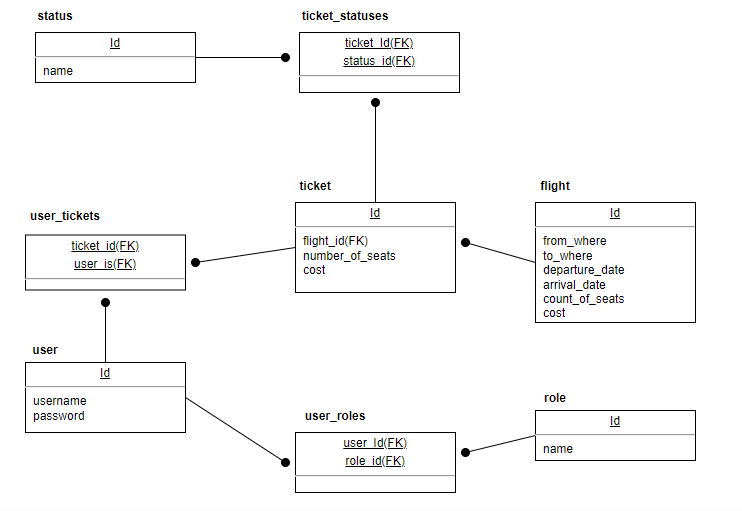


Рисунок 4.1 – Логическая модель системы

Физическое проектирование – описание физической структуры БД, т.е. ее размещения на запоминающем устройстве (рисунок 4.2):

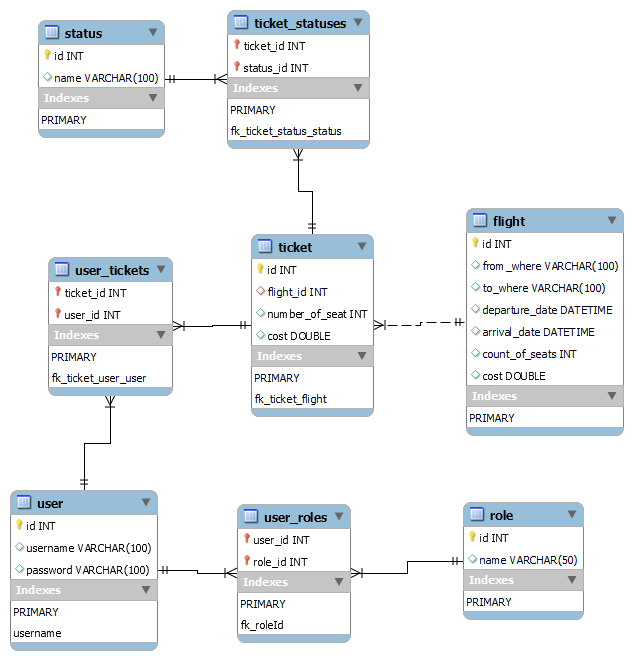


Рисунок 4.2 –Физическая модель системы

В данной информационной модели содержатся различные сущности. Под сущностями следует понимать, что это объект предметной области, который исследуется и моделируется. Иными словами, сущность – это любой различимый объект, о котором необходимо хранить информацию в базе данных.

Каждая сущность содержит атрибуты, которые определяют и отображают свойства и качества объекта. Например, сущность «ticket» имеет следующие атрибуты: «id», «flight\_id», «number\_of\_seats», «cost». При этом атрибут «id» является первичным ключом, а атрибут «flight\_id» внешним ключом.

Следующая сущность описывает рейсы, которые будет осуществлять аэропортовая структура в определённом направлении. Атрибутами данного объекта являются «id», «from\_where», «to\_where», «departure\_date», «arrival\_date», «count\_of\_seats», «cost». Первичный ключ – «id».

Следующая сущность «ticket\_statuse» соединяет две сущности: «ticket» и «status». Даная сущность имеет составной первичный ключ, который включает в себя «ticket\_id» и «status\_id», которые в свою очередь являются внешними ключами. Потенциальными первичными ключами могут быть и другие атрибуты данной сущности, однако необходимо выполнение следующих условий: уникальность и минимальность.

Сущность «user\_tickets» аналогична сущности «ticket\_statuse» . Она также в атрибутах содержит два внешних ключа, которые в свою очередь составляют первичный ключ, и соединяет сущности «user» и «ticket» . Данная структура необходима для того, чтобы органировать связь много-ко многим и обеспечить третью нормальную форму базы данных.

Еще одна сущнось, которая аналогична двум предыдущим – «user\_roles». Ее атрибутами являются «user\_id», «\_role\_id». Эта таблица содержит в себе идентификационные номера пользователя и его роли. Таким образом данная сущность соединяет две таблицы «user» и «role». Атрибуты «user\_id» и «role\_id» являются внешними ключами и составляют первичный ключ.

Следующая сущность «user» содержит данные о потенциальных клиентах аэропорта со следующими полями: «id», «username», «password». Первичный ключ в данной сущности – «id». Поля «username» и «password» в данной системе являются логином и паролем пользователя. Это необходимо для авторизации пользователя в системе для выполнения им различных действий: отмена билетов, бронирование, хранения, а также их печать.

Оставшиеся две сущности представлены таблицами «status» и «role». Они описываюс состояние билета(куплен или забронирован) и роль прользователя (админ или юзер). Первичный ключ у обеих сущностей – «id», а поле – «name».

**5 ОПИСАНИЕ АЛГОРИТМА, РЕАЛИЗУЮЩЕГО**

**БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ СИСТЕМЫ**

Приложение разработано на основе архитектуры «клиент-сервер». Для создания насыщенного графического интерфейса была использована библиотека JavaFX. Работа с информационной моделью системы была осуществлена с помощью объектно-реляционной системой управления базами данных MySql.   
 Для начала работы необходимо осуществить идентификацию для базы данных. Пользователь вводит данные на стороне сервера. При этом необходимо подключить клиента по уникальному порту. После обработки данных в консоли сервера выводится сообщения об успешном установлении соединения. Затем сервер отправляет сообщение клиенту о результатах выполнения своей работы. Выведенное сообщение отображает результат выполненных работ.

Далее пользователю необходимо авторизоваться. Для этого в соответствующей форме необходимо ввести уникальные пароль и логин. После авторизации данные передаются на сервер. Для корректной идентификации пользователя был разработан метод «initialize» класс «LoginController» в клиентской части приложения. В этом классе собраны обработчики кнопок и полей, которые необходимы пользователю в процессе авторизвции. В случае успешной авторизации выводится пользлватель попадает в другое окно приложения. Также пользователь может зарегистрироваться в системе. Для этого используется тоже метод «initialize», но уже другого класса, класса «RegistrationController. В методах данных классов реализовано сообщение с серверной частью приложения, где и происходит обработка всех данных.

В автоматизированной системе продажи авиационных билетов основным алгоритмом является алгоритм продажи билета.

После того, как пользователь зарегистировался или авторизировался, на экране появляется главное меню, где пользователь может просмотреть все свои купленные ранее билеты, ранее забронированные билеты, а также может найти нужный ему рейс. Для нахождения рейса предусмотрен метод «initialize» класса «FindFlightClient».

После нажатия кнопки «найти рейс» в главном меню, срабатывает обработчик этой кнопки и появляется следующее окно, которое предлагает пользователю ввести направления рейса и дату дату отправления. Когда пользователь ввел данные в текстовые поля, происходит их считавание в переменные «fromWhere», «toWhere» и «date». Если поля направлений заполнены, то серверу отправляется команда FIND\_FLIGHT\_USE\_PARAMS. Далее, если пользователь заполнил поле для даты, то серверу отправляется полная информация рейса, включающая направления и дату. Сервер в свою очередь принимает команду FIND\_FLIGHT\_USE\_PARAMS и полную информацию рейса и преобразовывает ее в объект типа FlightDto. Далее идет преобразование в тип Flight. Далее, если дата введена верно, то идет поиск данного рейса в базе данных. После успешного находжения рейса, объект опять преобразуется в тип FlightDto и отправляется клиенту. Если же пользователь изначально задал только направление рейса и не указал дату, то поиск будет производиться только по направлению и в дальнейшем в таблицу выведутся все рейсы данного направления.

Алгоритм покупки билета представлен на рисунке 5.1.

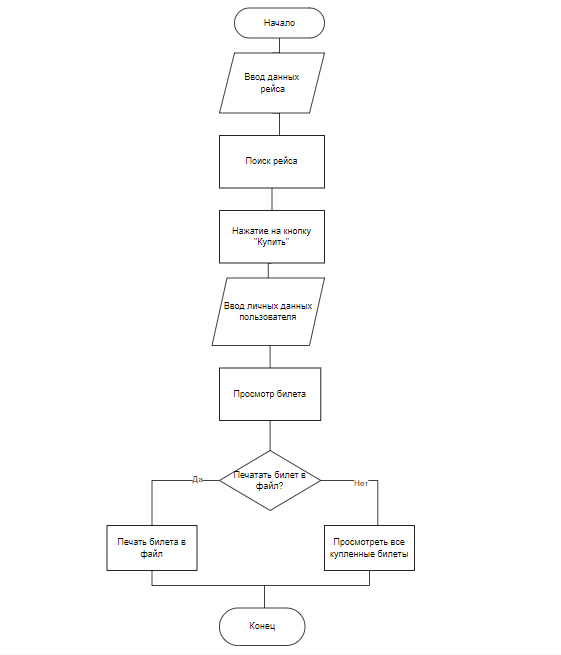
****

Рисунок 5.1 – Блок-схема алгоритма покупки билета

**6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

* 1. **Алгоритм работы пользователя**

Для корректного запуска системы, компьютер или рабочая станция, на которой разворачиваются клиентская и серверная части приложения должен обладать следующими минимальными требованиями

* 32х разрядная система ОС Windows 7 и выше;
* СУБД MySql;
* Java Development Environment (IntelliJ IDEA Ultimate).

После успешного запуска серверной части приложения появляется окно авторизации. В форме необходимо ввести логин и пароль существующего пользователя или нажать кнопку «Зарегистрироваться», если пользователь новый. Пример окна авторизации приведен на рисунке 6.1.

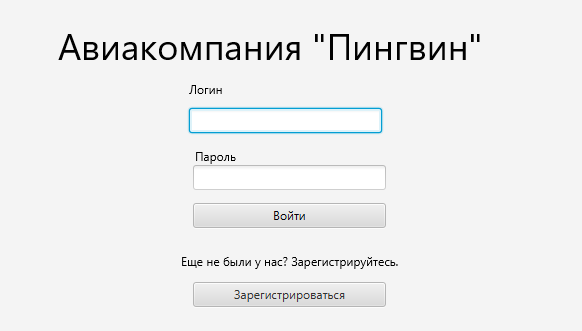


Рисунок 6.1 – окно авторизации

Если пользователь хочет зарегистрироваться, то для него всплывает форма, представленная на рисунке 6.2. Пользователю необходимо ввести уникальный логин и два раза повторить пароль.

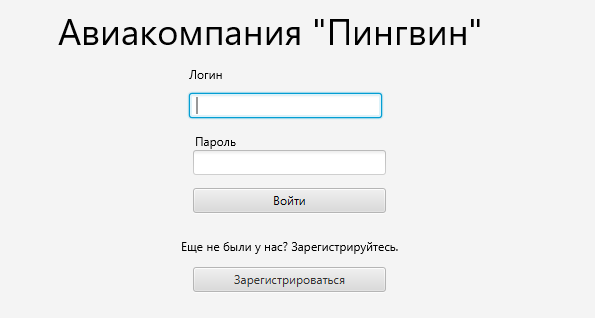


Рисунок 6.2 – Окно регистрации

После успешной авторизации или регистрации пользователь попадает в главное меню приложения. Ранее сущестующий пользователь может посмотреть уже купленные и забронированные прежде билеты,а также предусмотрена возможность их возврата, у зарегистрированного только что пользователя эти списки будут пусты. Также, пользователи могут найти рейс или выйти из главного меню.

Форма главного меню изображена на рисунке 6.3.

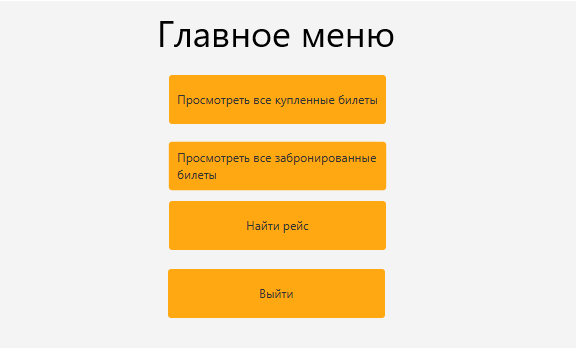


Рисунок 6.3 – Форма главного меню

Форма просмотра всех купленных билетом приведена на рисунке 6.4

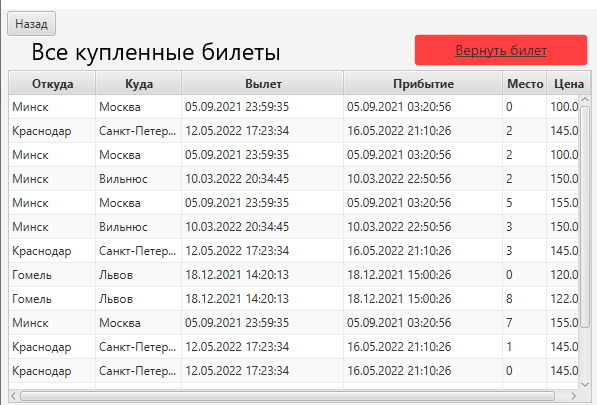


Рисунок 6.4 – Форма главного меню

Если пользователь хочет найти рейс, то он попадает в окно поиска рейса. В окне необходимо заполнить данные рейса для дальнейшей работы программы.

Форма поиска рейса приведена на рисунке 6.5.

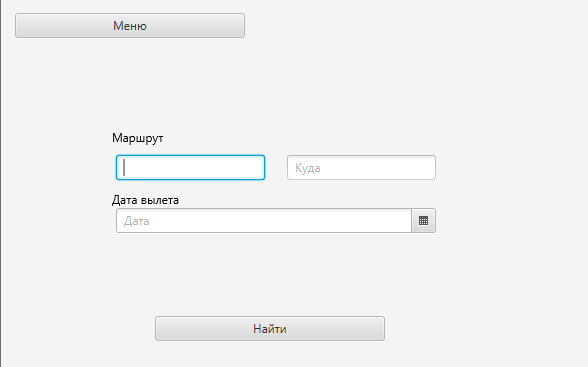


Рисунок 6.5 – Форма поиска рейса

Если пользователь не ввел данные рейса, то ему представятся все имеющиеся рейсы. В окне, где представлены все рейсы, можно купить или забронировать билет. Данная форма представлена на рисунке 6.6.

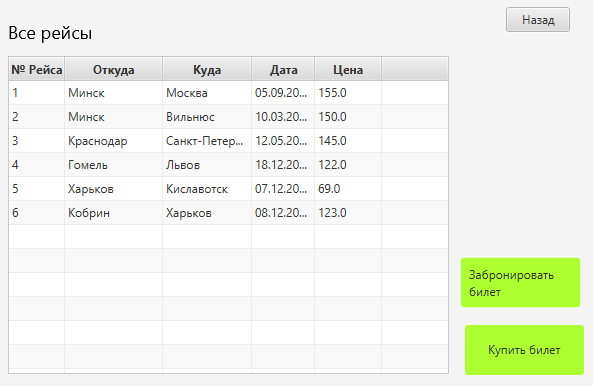


Рисунок 6.6 – Форма поиска рейса

После того, как пользователь выберет необходимый ему рейс и нажмет кнопку «купить», появится форм для заполнения личных данных. Форма заполнения личных данных пользователя приведена на рисунке 6.7.

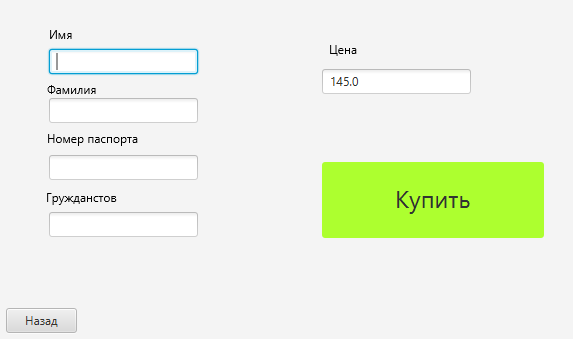


Рисунок 6.7 – Форма заполнения личных данных пользователя

После того, как пользователь ввел данные, перед ним появляется заказанный билет и предлагается напечатать его в файл или просмотреть все купленные билеты. Форма представлена на рисунке 6.8.

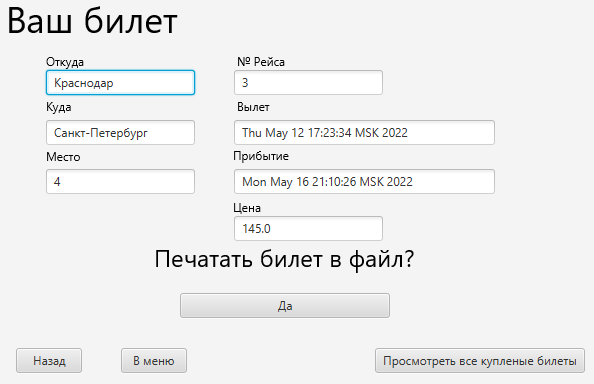


Рисунок 6.8 – Форма заполнения личных данных пользователя

Форма напечатанного в файл билета приведена на рисунке 6.9

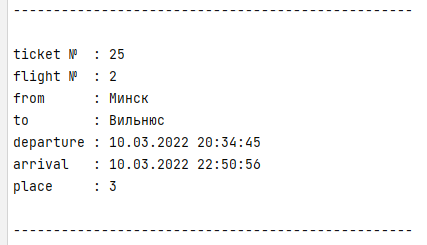


Рисунок 6.9 – Форма заполнения личных данных пользователя

**6.2 Алгоритм работы администратора**

Для корректной работы информационной системы была предусмотрена возможность управления процессами в качестве администратора.

Администратору, как и обычному пользователю, необходимо

авторизоваться. После успешной авторизации появляется главная форма, которая позволяет данному пользователю управлять процессами системы.

Администратор имеет возможность просматривать информацию о рейсах и билетах. Также для удобства сортировки информации была предусмотрена функция для поиска (рисунок 6.10).

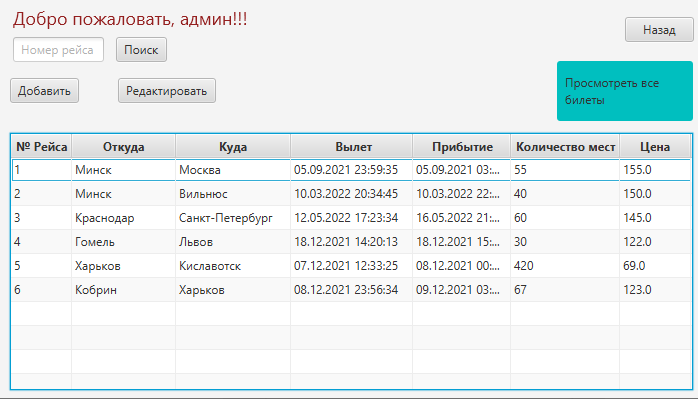


Рисунок 6.10 – Форма администратора

Администратору предоставляется возможность редактирования и добавления рейса. Для того, чтобы отредактировать данные необходимо нажать на кнопку «Редактировать». После этого появляется форма для редактирования данных ( рисунок 6.11). После нажатия на кнопку «Добавить» появляется окно для добавления рейса (рисунок 6.12). Также, администратор может просматривать все билеты и изменять цену свободных билетов (рисунок 6.13).

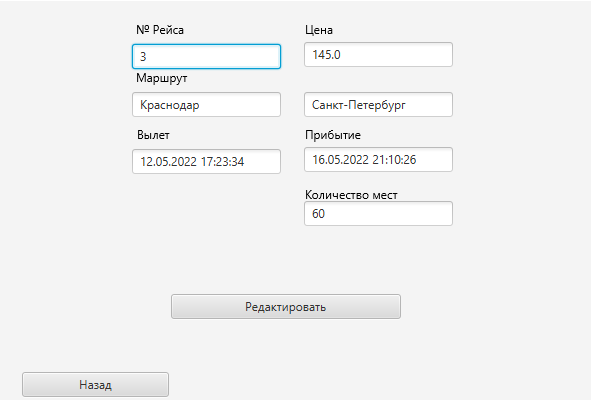


Рисунок 6.11 – Форма редактирования рейса

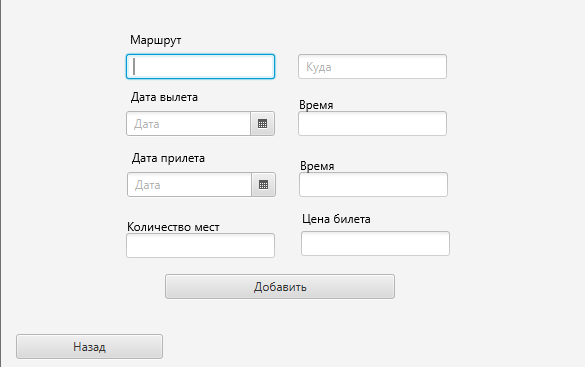


Рисунок 6.12 – Форма редактирования рейса

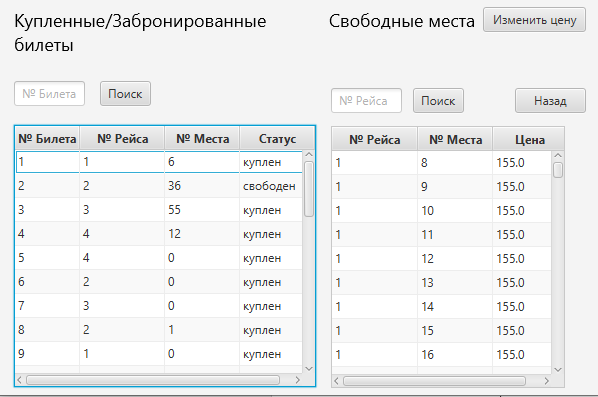


Рисунок 6.13 – Форма редактирования рейса

1. **РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ**

В ходе работы с приложением могли возникнуть исключительные ситуации. Разработанная система направлена на широкий круг пользователя, так как услугой покупки билета пользуется большое количество людей.

На этапе регистрации пользователя разработана исключительная ситуация, позволяющая выводить сообщения о некорректном вводе (рисунок 7.1).

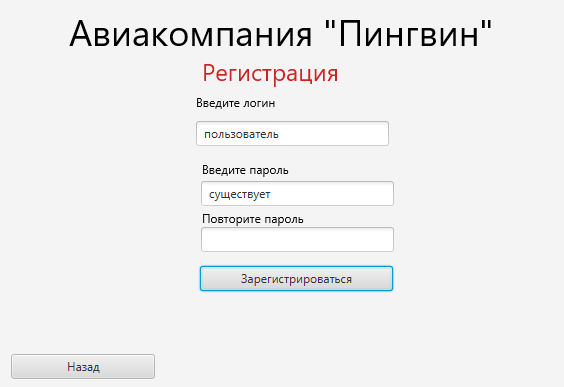


Рисунок 7.1 – Валидация при регистрации

Также, если при регистрации пользователь ввел два разных пароля, то возникнет исключитальная ситуация, сообщающая, что пользователь должен ввести два одинаковых пароля(рисунок 7.2).

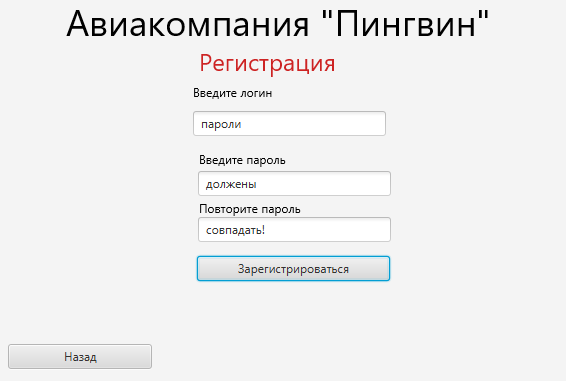


Рисунок 7.2 – Валидация при вводе неверных паролей

При авторизации пользователя предусмотрена возможность обработки ситуации, когда пользователь вводит неверные данные (рисунок 7.3).

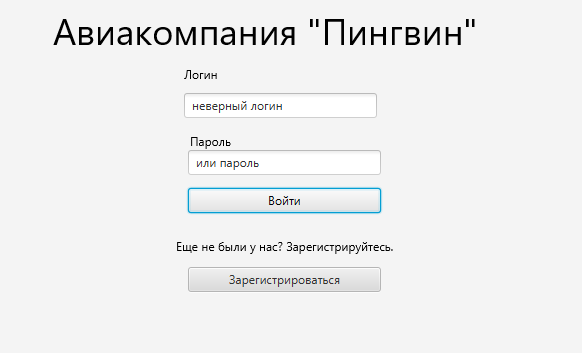


Рисунок 7.3 – Некорректный ввод

При успешной авторизации пользователя или администратора разработаны исключительные ситуации, позволяющие обеспечить корректную и эффективную работу пользователя.

Если авторизация происходит в качестве пользователя, то открывается главное меню, которая позволяет выбрать кнопку, реализующую поиск необходимого рейса. При этом, если пользователь указал неверные данные, то выводится просто пустая таблица(рисунок 7.4).

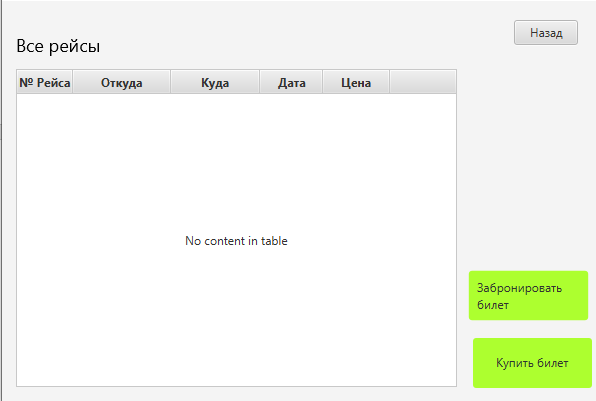


Рисунок 7.4 – Исключительная ситуация при поиске рейса

Также на данном этапе была предусмотрена ситуация, когда пользователь не заполнил все соответствующие поля. Например, при заполнении личных данных при покупке билета, если пользователь оставит какое-то поле пустым, то программа просто не пустит его дальше и не оформит купленный билет до тех пор, пока пользователь не введет все необходимые данные.

# 

# **ЗАКЛЮЧЕНИЕ**

Главной целью курсового проекта была разработка автоматизированной системы продажи авиационных билетов. При решении поставленной задачи был выбран ряд соответствующих технологий и инструментов, которые позволили осуществить корректную работу данного приложения.

При функциональном моделировании было создано точное описание проектируемой системы, а также интерпретация полученного описания для определения оценочных знаний некоторых характеристик системы.

Информационная модель системы была создана для корректной работы с данными: хранение, обработка, удаление. При этом данная модель способна предоставлять доступ к информации, необходимой пользователю.

Проектирование UML-диаграмм позволило сформулировать основные требования к информационной системе, а также обеспечить объективность в выработке требований к проектированию системы. Данные модели описания требований к информационной системе преобразуются в систему моделей, описывающих концептуальный проект информационной системы. При этом сформировались модели архитектуры информационной системы, требования к программному обеспечению и информационному обеспечению.

В рамках курсового проекта была продумана логика действий, спектр возможностей и функционал администратора и пользователя.

При выполнении курсовой работы были рассмотрены основные требования и задачи проектированной системы. Автоматизация аэропортовой структуры была выполнена в полной мере, что позволяет увеличить эффективность и конкурентоспособность системы. Автоматизация была проведена в основных процессах покупки и бронирования авиационных билетов. Благодаря этому появилась возможность обеспечить комфортное и своевременное взаимодействие с пользователем.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Автоматизация оценки эффективности аэропортовой инфраструктуры [Электронный ресурс]. – Режим доступа: http://cyberleninka.ru

[2] Классификация средств механизации и автоматизации процессов обслуживания авиапассажиров [Электронный ресурс]. – Режим доступа: <https://cyberleninka.ru/article/n/klassifikatsiya-sredstv-mehanizatsii-i-avtomatizatsii-protsessov-obsluzhivaniya-aviapassazhirov>

[3] Построение контекстной диаграммы [Электронный ресурс]. – Режим доступа: <file:///D:/pz_1.html>

[4] Черемных, С. Моделирование и анализ систем. IDEF-технологии / Черемных С.В., Семенов И.О., Ручкин В.С, Москва: Россия. – 2006. – 192 с.

[5] Батин, Н.В. Проектирование баз данных / Батин Н.В., Минск: Беларусь -2007. – 56 с.

[6] Базы данных [Электронный ресурс]. – Режим доступа: <file:///D:/EORD_bazy_dannykh.html>

# **ПРИЛОЖЕНИЕ A**

**( обязательное )**

**Листинг скрипта базы данных**

**Файл BD.sql:**

drop

database if exists course\_work;

create

database course\_work;

use

course\_work;

create table user

(

id int primary key auto\_increment,

username varchar(100) unique,

password varchar(100)

);

create table role

(

id int primary key auto\_increment,

name varchar(50)

);

create table user\_roles

(

user\_id int,

role\_id int,

constraint fk\_userId foreign key (user\_id) references user (id),

constraint fk\_roleId foreign key (role\_id) references role (id),

constraint pk\_user\_roles primary key (user\_id, role\_id)

);

create table flight

(

id int primary key auto\_increment,

from\_where varchar(100),

to\_where varchar(100),

departure\_date datetime,

arrival\_date datetime,

count\_of\_seats int,

cost double

);

create table ticket

(

id int primary key auto\_increment,

flight\_id int,

number\_of\_seat int,

cost double,

constraint fk\_ticket\_flight foreign key (flight\_id) references flight (id)

);

create table status

(

id int primary key auto\_increment,

name varchar(100)

);

create table ticket\_statuses

(

ticket\_id int,

status\_id int,

constraint fk\_ticket\_status\_ticket foreign key (ticket\_id) references ticket (id),

constraint fk\_ticket\_status\_status foreign key (status\_id) references status (id),

constraint pk\_ticket\_status primary key (ticket\_id, status\_id)

);

create table user\_tickets

(

ticket\_id int,

user\_id int,

constraint fk\_ticket\_user\_ticket foreign key (ticket\_id) references ticket (id),

constraint fk\_ticket\_user\_user foreign key (user\_id) references user (id),

constraint pk\_ticket\_user primary key (ticket\_id, user\_id)

);

insert into user(username, password)

values ('user', 'user'),

('admin', 'admin');

insert into role(name)

values ('USER'),

('ADMIN');

insert into user\_roles(user\_id, role\_id)

values (1, 1),

(2, 2);

insert into flight(from\_where, to\_where, departure\_date, arrival\_date, count\_of\_seats, cost)

values ('Минск', 'Москва', '2021-09-05 23:59:45', '2021-09-05 03:30:56', 56, 100),

('Минск', 'Москва', '2021-05-09 13:59:45', '2021-05-09 15:10:46', 56, 50),

('Минск', 'Москва', '2021-07-06 20:49:45', '2021-07-07 01:08:36', 56, 100),

('Минск', 'Вильнюс', '2022-03-10 20:34:45', '2022-03-10 22:50:56', 40, 150),

('Минск', 'Вильнюс', '2022-04-15 20:34:45', '2022-04-16 01:13:34', 30, 100),

('Минск', 'Вильнюс', '2022-04-10 12:24:45', '2022-04-10 17:50:56', 40, 160),

('Минск', 'Вильнюс', '2022-05-30 10:56:25', '2022-05-30 14:05:13', 40, 150),

('Краснодар', 'Санкт-Петербург', '2022-05-12 17:23:34', '2022-05-12 21:10:26', 60, 145),

('Краснодар', 'Санкт-Петербург', '2022-06-12 13:23:34', '2022-06-12 16:20:26', 60, 135),

('Гомель', 'Львов', '2021-12-18 14:20:13', '2021-12-18 15:00:26', 30, 120),

('Гомель', 'Львов', '2021-12-20 14:20:13', '2021-12-20 15:00:26', 30, 110),

('Гомель', 'Львов', '2021-12-28 10:20:13', '2021-12-28 11:40:26', 30, 100);

insert into ticket(flight\_id, number\_of\_seat, cost)

values (1, 6, 100),

(2, 36, 100),

(3, 55, 100),

(4, 12, 100);

insert into status(name)

values ('куплен'),('забронирован'),('свободен');

insert into ticket\_statuses(ticket\_id, status\_id)

values (1, 1),(2, 3),(3, 2),(4, 1);

insert into user\_tickets(ticket\_id, user\_id)

values (1, 1),(3, 1),(4, 1);

# **ПРИЛОЖЕНИЕ Б**

**( обязательное )**

**Листинг основных элементов**

**Файл DBConnection.java:**

package by.victoria.connection;  
  
import java.io.IOException;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;  
import java.net.Socket;  
  
public class ConnectionHandler {  
 private final Socket socket;  
 private final ObjectInputStream inputStream;  
 private final ObjectOutputStream outputStream;  
  
 public ConnectionHandler(Socket socket) {  
 this.socket = socket;  
 try {  
 this.outputStream = new ObjectOutputStream(socket.getOutputStream());  
 this.inputStream = new ObjectInputStream(socket.getInputStream());  
 } catch (IOException e) {  
 throw new RuntimeException("Can't create streams for send/receive message", e);  
 }  
 }  
  
 public void writeObject(Object object) {  
 try {  
 outputStream.writeObject(object);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public Object readObject() {  
 Object object = null;  
 try {  
 object = inputStream.readObject();  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 return object;  
 }  
  
  
 public void close() {  
 try {  
 inputStream.close();  
 outputStream.close();  
 socket.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

**Файл ClientHandler.java:**

package by.victoria.app;  
  
import by.victoria.command.Command;  
import by.victoria.connection.ConnectionHandler;  
import by.victoria.dao.FlightDao;  
import by.victoria.dao.TicketDao;  
import by.victoria.dao.UserDao;  
import by.victoria.mapper.FlightMapper;  
import by.victoria.mapper.TicketMapper;  
import by.victoria.mapper.UserMapper;  
import by.victoria.model.dto.FlightDto;  
import by.victoria.model.dto.FreePlaceDto;  
import by.victoria.model.dto.TicketDto;  
import by.victoria.model.dto.UserDto;  
import by.victoria.model.entity.Flight;  
import by.victoria.model.entity.Role;  
import by.victoria.model.entity.Ticket;  
import by.victoria.model.entity.User;  
  
import java.net.Socket;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.stream.Collectors;  
  
import static by.victoria.command.Command.\*;  
  
public class ClientHandler implements Runnable {  
 private final ConnectionHandler connectionHandler;  
  
 public ClientHandler(Socket socket) {  
 this.connectionHandler = new ConnectionHandler(socket);  
 }  
  
 @Override  
 public void run() {  
 UserDao userDao = new UserDao();  
 TicketDao ticketDao = new TicketDao();  
 FlightDao flightDao = new FlightDao();  
  
 UserMapper userMapper = new UserMapper();  
 TicketMapper ticketMapper = new TicketMapper();  
 FlightMapper flightMapper = new FlightMapper();  
  
 while (true) {  
 Command command;  
 try {  
 command = (Command) connectionHandler.readObject();  
 } catch (RuntimeException e) {  
 command = *EXIT*;  
 }  
 System.*out*.println(command);  
 switch (command) {  
 case *LOGIN* -> {  
 UserDto userDto = (UserDto) connectionHandler.readObject();  
  
 User user = userDao.find(userDto.getUsername(), userDto.getPassword());  
 if (user == null) {  
 connectionHandler.writeObject(*UNAUTHORIZED*);  
 } else {  
 List<String> roles = user.getRoles()  
 .stream()  
 .map(Role::getName)  
 .collect(Collectors.*toList*());  
 if (roles.contains("USER")) {  
 connectionHandler.writeObject(*USER*);  
 } else {  
 connectionHandler.writeObject(*ADMIN*);  
 }  
 connectionHandler.writeObject(userMapper.userToUserDto(user));  
 }  
 }  
 case *REGISTRATION* -> {  
 UserDto userDto = (UserDto) connectionHandler.readObject();  
 User user = userMapper.userDtoToUser(userDto);  
 connectionHandler.writeObject(userDao.create(user));  
 }  
 case *BUY\_TICKET* -> {  
 UserDto userDto = (UserDto) connectionHandler.readObject();  
 FlightDto flightDto = (FlightDto) connectionHandler.readObject();  
  
 User user = userMapper.userDtoToUser(userDto);  
 Flight flight = flightMapper.flightDtoToFlight(flightDto);  
  
 Ticket ticket = new Ticket(flight, flight.getCost());  
 ticket = ticketDao.buyTicket(user, ticket);  
 TicketDto ticketDto = ticketMapper.ticketToTicketDto(ticket);  
 connectionHandler.writeObject(ticketDto);  
 }  
 case *BUY\_TICKET\_AFTER\_BOOKING* -> {  
 TicketDto ticketDto = (TicketDto) connectionHandler.readObject();  
 Ticket ticket = ticketMapper.ticketDtoToTicket(ticketDto);  
 ticket = ticketDao.buyTicketAfterBooking(ticket);  
 ticketDto = ticketMapper.ticketToTicketDto(ticket);  
 connectionHandler.writeObject(ticketDto);  
 }  
 case *FIND\_ALL\_TICKETS* -> {  
 List<TicketDto> ticketDtoList = ticketMapper.ticketListToTicketDtoList(ticketDao.findAll());  
 connectionHandler.writeObject(ticketDtoList);  
 }  
 case *FIND\_ALL\_BOUGHT\_TICKETS* -> {  
 Integer userId = (Integer) connectionHandler.readObject();  
 List<TicketDto> ticketDtoList = ticketMapper  
 .ticketListToTicketDtoList(ticketDao.findAllBoughtTickets(userId));  
 connectionHandler.writeObject(ticketDtoList);  
 }  
 case *FIND\_ALL\_BOOKED\_TICKETS* -> {  
 Integer userId = (Integer) connectionHandler.readObject();  
 List<TicketDto> ticketDtoList = ticketMapper  
 .ticketListToTicketDtoList(ticketDao.findAllBookedTickets(userId));  
 connectionHandler.writeObject(ticketDtoList);  
 }  
 case *BOOK\_TICKET* -> {  
 UserDto userDto = (UserDto) connectionHandler.readObject();  
 FlightDto flightDto = (FlightDto) connectionHandler.readObject();  
  
 User user = userMapper.userDtoToUser(userDto);  
 Flight flight = flightMapper.flightDtoToFlight(flightDto);  
  
 Ticket ticket = new Ticket(flight, flight.getCost());  
 ticket = ticketDao.bookTicket(user, ticket);  
 TicketDto ticketDto = ticketMapper.ticketToTicketDto(ticket);  
 connectionHandler.writeObject(ticketDto);  
 }  
 case *CANCEL\_RESERVATION\_TICKET*, *RETURN\_TICKET* -> {  
 TicketDto ticketDto = (TicketDto) connectionHandler.readObject();  
 UserDto userDto = (UserDto) connectionHandler.readObject();  
  
 User user = userMapper.userDtoToUser(userDto);  
 Ticket ticket = ticketMapper.ticketDtoToTicket(ticketDto);  
 ticketDao.delete(user, ticket);  
 }  
 case *FIND\_FLIGHT* -> {  
 Integer id = (Integer) connectionHandler.readObject();  
 Flight flight = flightDao.find(id);  
 FlightDto flightDto = flightMapper.flightToFlightDto(flight);  
 connectionHandler.writeObject(flightDto);  
 }  
 case *FIND\_FLIGHT\_USE\_PARAMS* -> {  
 FlightDto flightDto = (FlightDto) connectionHandler.readObject();  
 Flight flight = flightMapper.flightDtoToFlight(flightDto);  
  
 List<Flight> flights;  
  
 if (flight.getDepartureDate() != null) {  
 flights = flightDao.findByFromWhereAndToWhereAndDepartureDate(flight);  
 } else {  
 flights = flightDao.findByFromWhereAndToWhere(flight);  
 }  
 List<FlightDto> flightDtos = flightMapper.flightListToFlightDtoList(flights);  
 connectionHandler.writeObject(flightDtos);  
 }  
 case *FIND\_ALL\_FLIGHTS* -> {  
 List<FlightDto> flightDtoList = flightMapper.flightListToFlightDtoList(flightDao.findAll());  
 connectionHandler.writeObject(flightDtoList);  
 }  
 case *FIND\_ALL\_FREE\_PLACES* -> {  
 List<Ticket> ticketList = ticketDao.findAll();  
 List<FreePlaceDto> result = ticketMapper.ticketListToFreePlaceDtoList(ticketList);  
 connectionHandler.writeObject(result);  
 }  
 case *ADD\_FLIGHT* -> {  
 FlightDto flightDto = (FlightDto) connectionHandler.readObject();  
 Flight flight = flightMapper.flightDtoToFlight(flightDto);  
 flightDao.create(flight);  
 }  
 case *EDIT\_FLIGHT* -> {  
 FlightDto flightDto = (FlightDto) connectionHandler.readObject();  
 Flight flight = flightMapper.flightDtoToFlight(flightDto);  
 flightDao.update(flight);  
 }  
 case *EXIT* -> {  
 connectionHandler.close();  
 return;  
 }  
 }  
 }  
 }  
}

**Файл HelloUserController.java:**

package by.victoria.controller;  
  
import by.victoria.app.CurrentFlight;  
import by.victoria.app.CurrentTicket;  
import by.victoria.app.CurrentUser;  
import by.victoria.app.ShowWindowFxml;  
import javafx.fxml.FXML;  
import javafx.scene.control.Button;  
import javafx.stage.Stage;  
public class HelloUserController {  
 @FXML  
 public Button Button\_mainmenu\_bookedtickets;  
 @FXML  
 public Button Button\_mainmenu\_searchflght;  
 @FXML  
 public Button Button\_mainmenu\_logout;  
 @FXML  
 public Button Button\_mainmenu\_boughtickets;  
  
 public void initialize() {  
 CurrentFlight.*setCurrentFlightDto*(null);  
 CurrentFlight.*setCurrentFlightDtoList*(null);  
 Button\_mainmenu\_bookedtickets.setOnAction(event -> {  
 ((Stage) Button\_mainmenu\_bookedtickets.getScene().getWindow()).close();  
 new ShowWindowFxml("allBookedTickets"); });  
 Button\_mainmenu\_searchflght.setOnAction(event -> {  
 ((Stage) Button\_mainmenu\_searchflght.getScene().getWindow()).close();  
 new ShowWindowFxml("findFlightClient");});  
 Button\_mainmenu\_logout.setOnAction(event -> {  
 CurrentUser.*setCurrentUserDtoNull*();  
 CurrentFlight.*setCurrentFlightDto*(null);  
 CurrentTicket.*setCurrentTicketDto*(null);  
 ((Stage) Button\_mainmenu\_logout.getScene().getWindow()).close();  
 new ShowWindowFxml("login"); });  
 Button\_mainmenu\_boughtickets.setOnAction(event -> {  
 ((Stage) Button\_mainmenu\_logout.getScene().getWindow()).close();  
 new ShowWindowFxml("allBoughtTickets");  
 });  
 }

**ПРИЛОЖЕНИЕ В  
 (обязательное)  
 Графический материал**

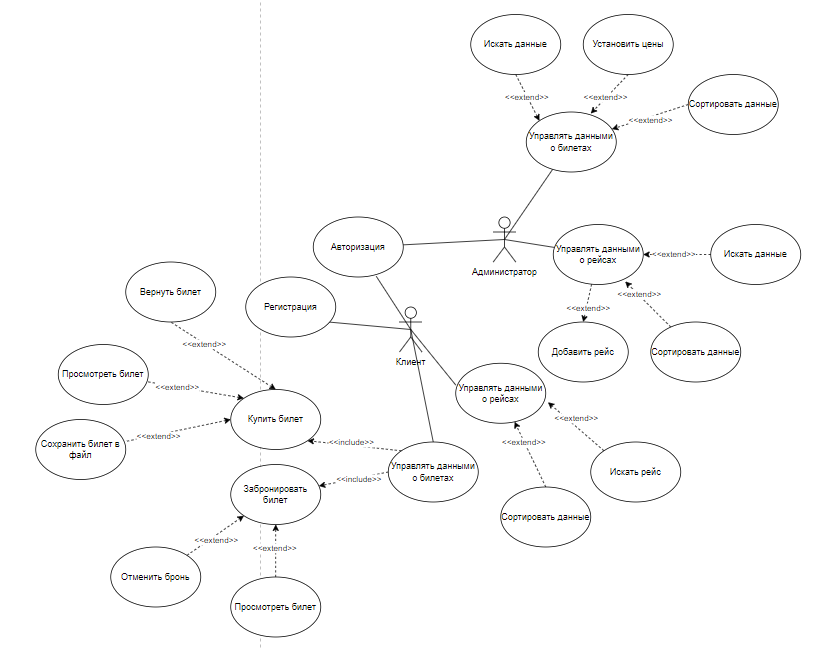


Рисунок В.1 – Диаграмма вариантов использований

Продолжение приложения В

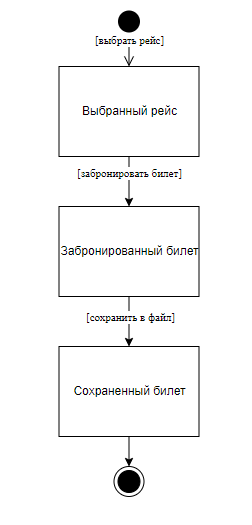


Рисунок В.2 – Диаграмма состояний

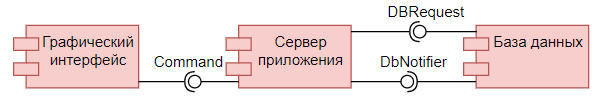


Рисунок В.3 – Диаграмма компонентов

Продолжение приложения В

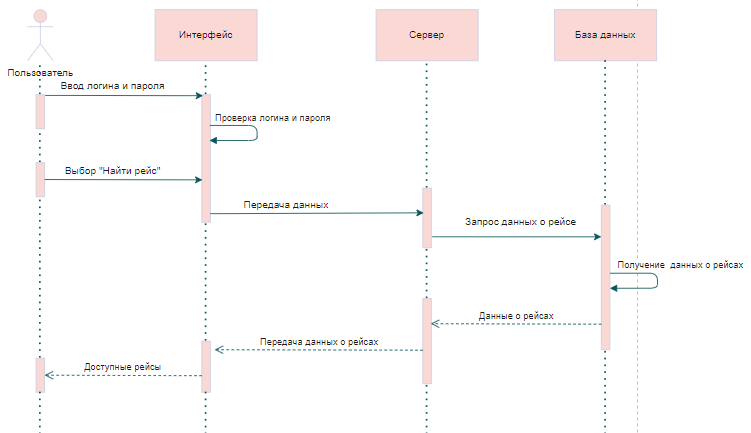


Рисунок В.4 – Диаграмма последовательностей

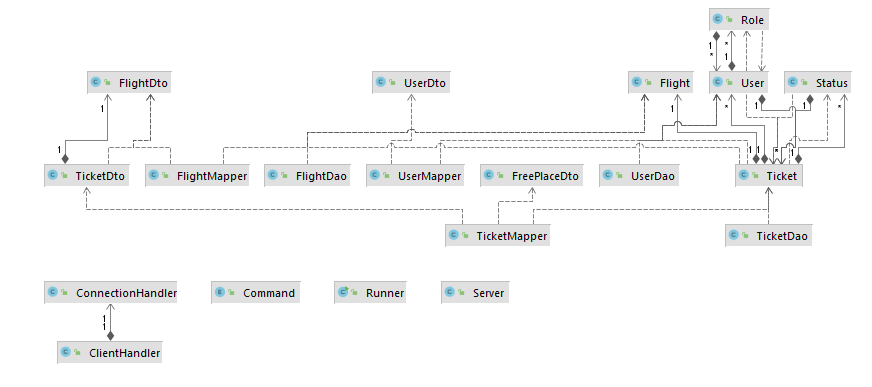


Рисунок В.5 – Диаграмма классов серверной части

Продолжение приложения В

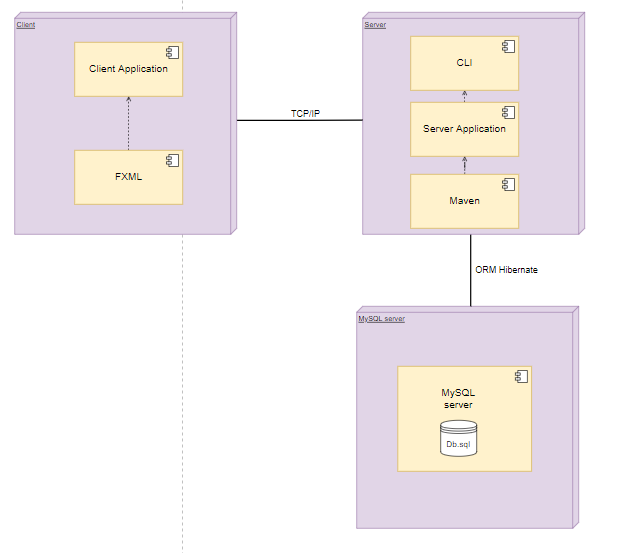


Рисунок В.6 – Диаграмма развертывания