Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Распределенные системы обработки информации»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  ассистент кафедры экономической информатики  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. П. Лыщик |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«АВТОМАТИЗИРОВАННАЯ СИСТЕМА РАБОТЫ ТУРИСТИЧЕСКОГО АГЕНТСТВА»**

|  |  |
| --- | --- |
|  | Выполнил студент группы 074002  Апанович Ангелина Александровна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| [Введение ……………………………………………………………………….](#_heading=h.4d34og8) 6  [1 Анализ и моделирование предметной области программного средства ..](#_heading=h.2s8eyo1) 7  [1.1 Описание предметной области ………………………………………...](#_heading=h.17dp8vu) 7  [1.2 Разработка функциональной модели предметной области ………….](#_heading=h.2et92p0) 8  [1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований ……………………….. 1](#_heading=h.2et92p0)2  [1.4 Разработка информационной модели предметной области ……….... 14](#_heading=h.2et92p0)  [1.5 Модели представления программного средства и их описание ……. 1](#_heading=h.2et92p0)7  [2 Проектирование и конструирование программного средства ………….](#_heading=h.3rdcrjn) 22  [2.1 Постановка задачи …………………………………………………… 2](#_heading=h.17dp8vu)2  [2.2 Архитектурные решения ……………………………………………… 2](#_heading=h.2et92p0)3  [2.3 Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства …………………………..](#_heading=h.2et92p0) 24  [2.4 Проектирование пользовательского интерфейса …………………….](#_heading=h.2et92p0) 26  [2.5 Обоснование выбора компонентов и технологий для реализации программного средства ……………………………………………….](#_heading=h.2et92p0) 26  2.5.1 Spring Framefork ………………………………………………… 26  2.5.2 Hibernate …………………………………………………………. 28  2.5.3 PostgreSQL...……………………………………………………....29 |  |

# **ВВЕДЕНИЕ**

# **Анализ и моделирование предметной области программного средства**

* 1. **Описание предметной области**

Туристическое агентство предоставляет услуги, которые позволяют удовлетворить естественный интерес людей, познакомиться с образом жизни, менталитетом и культурой других стран. Предоставляемые агентством услуги ориентированы, главным образом, на индивидуальных туристов или очень небольшие группы, состоящие из трех-четырех человек, которые заинтересованы в том, чтобы получить более глубокое представление о реальной жизни не только в Беларуси, но и за рубежом.

Автоматизация турагентства в наше время сводится к выбору облачной CRM-системы. Современный рынок программных продуктов подобного класса достаточно разнообразен и широк. Успешность выбора подходящего IT-решения для автоматизации турагентства будет определяться четким представлением о своих потребностях, а также знанием существующих сегодня систем [2].

Увеличивающиеся год от года клиентские базы вместе с все более ожесточающейся конкуренцией на рынке требовали от турагентств новых решений, которые позволяли бы работать проще, быстрее и эффективнее. Ими стали системы автоматизации.

Сейчас на рынке программного обеспечения для туристического бизнеса представлено немало различных CRM систем (сокр. «customer relationship management») для автоматизированного управления деятельностью турагентств. Подобные системы представляют собой компьютерные программы, устанавливаемые на персональные компьютеры пользователей или предоставляемые в аренду в виде облачных веб-приложений, выполняющие все основные операции, с которыми имеют дело менеджеры по туризму.

Внедрить необходимую автоматизированную систему в компании можно, используя различные способы приобретения.

Различают несколько способов приобретения информационных систем:

* разработка, то есть способ, подразумевающий проектирование и программирование собственными силами средствами компании;
* покупка готового решения, то есть один из способов приобретения ИС, который подразумевает покупку готовой ИС у компании, разрабатывающей и/или реализующей разного рода ИС, и ее использование;
* покупка готового решения и его доработка, то есть один из способов приобретения ИС, который подразумевает покупку готового ядра требуемой ИС у компании, разрабатывающей и/или реализующей разного рода ИС, для того, чтобы внести в продукт свои значительные изменения и только, после этого, использовать [3].

Самостоятельное изготовление АРМ – самый дешевый способ. Имеет как преимущества, так и недостатки. Преимущества в том, что исключены лишние затраты на оплату готовых решений, проект делается исключительно для своих нужд и своими силами – что увеличивает приспособленность системы под бизнес-процессы предприятия. Недостаток же только в том, что надежность и время создания системы своими силами зависит от компетенции и профессионализма программистов.

Создание системы силами собственных сотрудников гарантирует: уникальность кода, собственные методы защиты от взлома, создание оптимизированных участков кода, масштабируемость системы в будущем, так как собственную программу человеку менять намного легче, чем разбираться в чужой программе, сСопровождение и улучшение программы возможно все время, пока на фирме есть хоть один ИТ-сотрудник, обладающий соответствующими знаниями.

Обычно в число основных функций, которые должны быть представлены в CRM системах, входят следующие функции:

* управление клиентами;
* управление продажами;
* управление бухгалтерией;
* управление документооборотом;
* управление аналитикой и отчетами.

Наиболее популярные сегодня на рынке программного обеспечения русскоязычные облачные CRM системы, предназначенные сугубо для туристической отрасли: Селена, Columbis, CRM Tour, CRM Travels, ERP.travel, ITERIOS Travel Agent (ITA), MAG.Travel, TourControl.

* 1. **Разработка функциональной модели предметной области**

Для моделирования процесса работы интернет-магазина воспользуемся методологией SADT (IDEF0). Данная методология при описании функционального аспекта информационной системы конкурирует с методами, ориентированными на потоки данных (DFD). В отличие от них IDEF0 позволяет описывать любые системы, а не только информационные (DFD предназначена для описания программного обеспечения) и создавать описание системы и ее внешнего окружения до определения окончательных требований к ней. Иными словами, с помощью данной методологии можно постепенно выстраивать и анализировать систему даже тогда, когда трудно еще представить ее воплощение [4].

Таким образом, IDEF0 может применяться на ранних этапах создания широкого круга систем. В то же время она может быть использована для анализа функций существующих систем и выработки решений по их улучшению.

Основу методологии IDEF0 составляет графический язык описания процессов. Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Модель может содержать 4 типа диаграмм:

* контекстную диаграмму;
* диаграммы декомпозиции;
* диаграммы дерева узлов;
* диаграммы только для экспозиции (FEO).

Контекстная диаграмма (или же диаграмма верхнего уровня), являясь вершиной древовидной структуры диаграмм, показывает назначение системы (основную функцию) и ее взаимодействие с внешней средой. В каждой модели может быть только одна контекстная диаграмма. После описания основной функции выполняется функциональная декомпозиция, т.е. определяются функции, из которых состоит основная.

Далее функции делятся на подфункции и так до достижения требуемого уровня детализации исследуемой системы. Диаграммы, которые описывают каждый такой фрагмент системы, называются диаграммами декомпозиции. После каждого сеанса декомпозиции проводятся сеансы экспертизы – эксперты предметной области указывают на соответствие реальных процессов созданным диаграммам. Найденные несоответствия устраняются, после чего приступают к дальнейшей детализации процессов.

Диаграмма дерева узлов показывает иерархическую зависимость функций (работ), но не связи между ними. Их может быть сколько угодно, поскольку дерево можно построить на произвольную глубину и с произвольного узла [5].

Диаграммы для экспозиции строятся для иллюстрации отдельных фрагментов модели с целью отображения альтернативной точки зрения на происходящие в системе процессы (например, с точки зрения руководства организации).

Контекстная диаграмма работы туристического агентства ООО «TripAgent» приведена на рисунке 1.1.

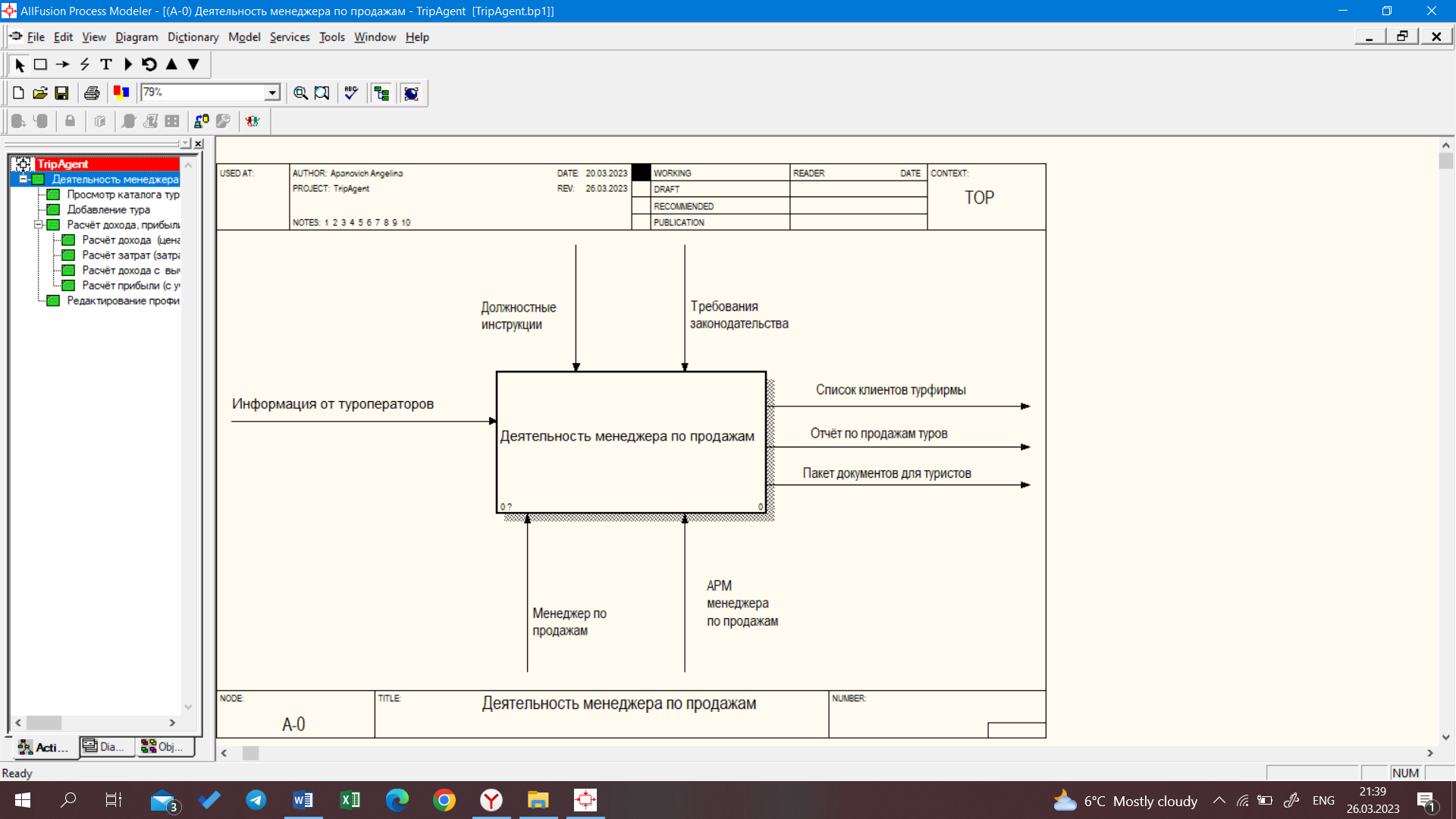


Рисунок 1.1 – Контекстная диаграмма верхнего уровня

Входными данными является информация от туроператоров. Результатом работы программы будет являться полученная отчетность по продажам туров, список клиентов турфирмы, пакет необходимых документов для туристов. В качестве механизма выступают менеджер по продажам и АРМ менеджера по продаже туров. Управлением будут являться должностные инструкции и требования законодательства.

После декомпозиции верхнего уровня диаграммы процесс раскладывается на следующие этапы:

* просмотр каталога туров;
* добавление тура;
* произвести ремонрасчёт дохода, прибыли;
* редактирование профиля.

Эти этапы представлены на рисунке 1.2.

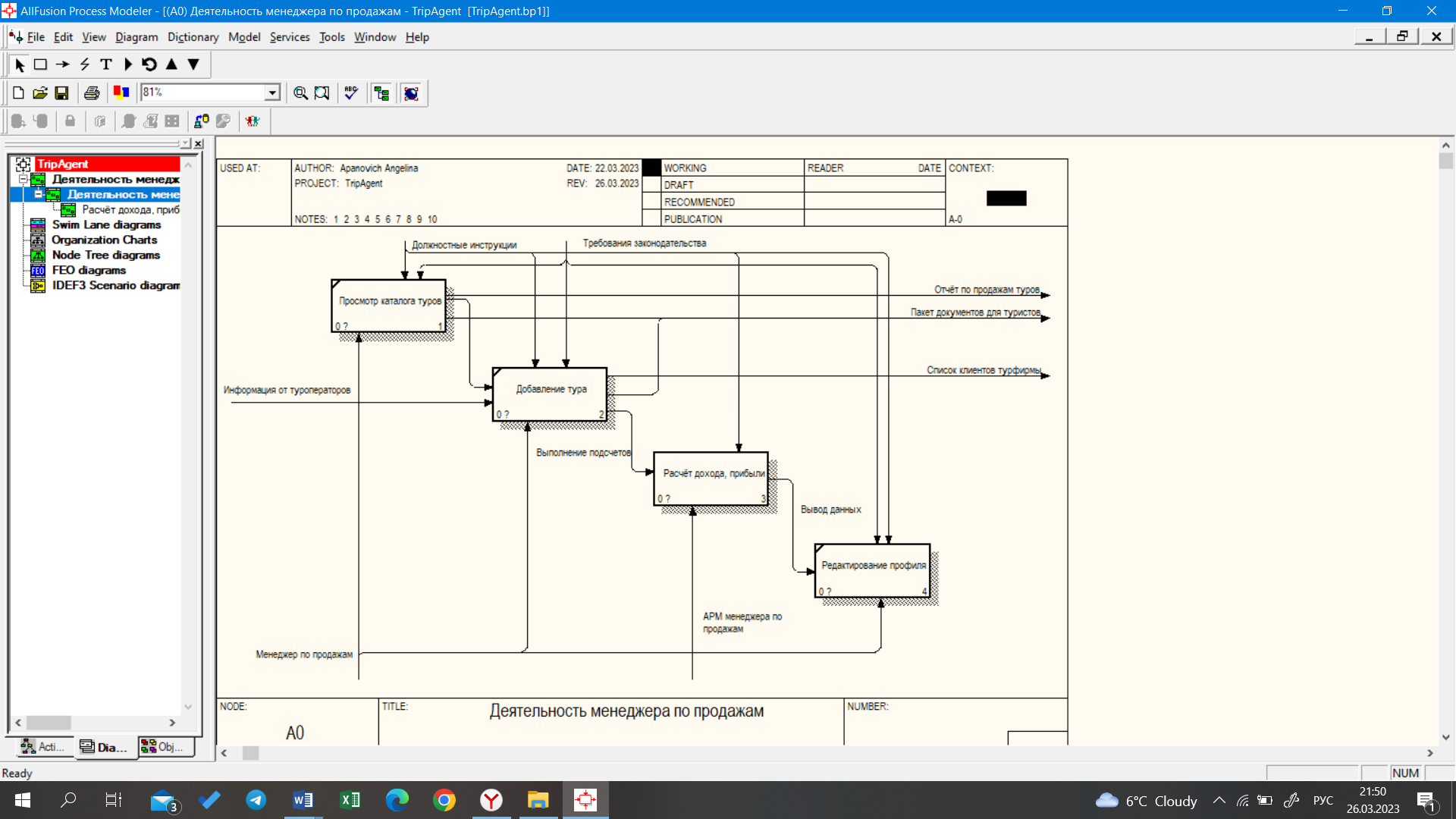


Рисунок 1.2 - Декомпозиция контекстной диаграммы «Деятельность менеджера по продажам»

На рисунке представлена работа функционального блока, выполненная при помощи декомпозиции и построена диаграмма второго уровня. На ней представлено четыре функциональных блока декомпозиции, распределение входных данных и управляющих механизмов.

Данный уровень включает в себя:

* расчёт дохода (цена тура \* кол-во продаж);
* расчёт затрат (затраты на 1 клиента \* количество продаж);
* расчёт дохода с вычетом затрат;
* расчёт прибыли (с учётом подоходного налога).

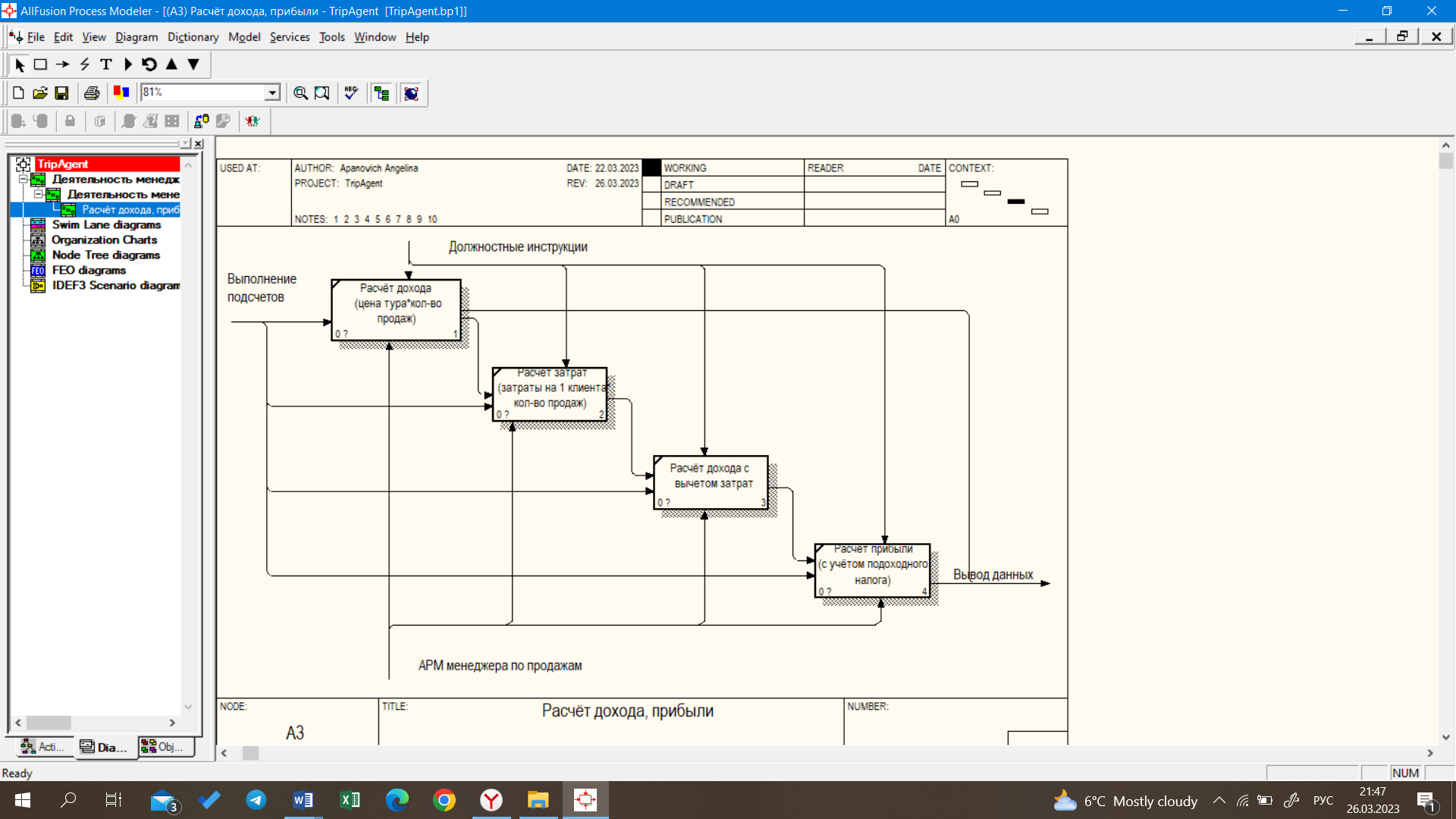


Рисунок 1.3 – Декомпозиция процесса «Расчёт дохода, прибыли»

Таким образом, весь процесс функционирования разбит на 4 подпроцесса, в свою очередь один из них разбивается на дополнительные подпроцессы. В итоге IDEF0 предоставляет преобразование одного сложного процесса на систему подпроцессов.

* 1. **Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований**

Функциональные требования (functional requirements) определяют функциональность ПО, которую разработчики должны построить, чтобы пользователи смогли выполнить свои задачи в рамках бизнес-требований. Иногда они называются требованиями поведения (behavioral requirements), они содержат положения с традиционным «должен» или «должна»: например, «система должна по электронной почте отправлять пользователю подтверждение о заказе».

Диаграммы Use Case языка UML – это важный и ценный метод анализа требований, который широко используется в современной разработке программного обеспечения.

Диаграммы вариантов использования описывают взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующими в процессе.

Вариант использования представляет собой типичное взаимодействие пользователя и проектируемой системы. Варианты использования характеризуются рядом свойств:

* вариант использования охватывает некоторую очевидную для пользователей функцию;
* вариант использования может быть как небольшим, так и достаточно крупным;
* вариант использования решает некоторую дискретную задачу пользователя.

В простейшем случае вариант использования создается в процессе обсуждения с пользователями тех вещей, которые они хотели бы получить от системы. При этом каждой отдельной функции, которую они хотели бы реализовать, присваивается некоторое имя и записывается ее краткое текстовое описание [6].

В данном курсовом проекте были определены два основных действующих лица: зарегистрированный пользователь, он же менеджер, и не зарегистрированный пользователь.

Менеджеру и незарегистрированному пользователю представлены некоторые одинаковые функции, такие как:

* возможность создать аккаунт;
* просмотр информации о компании.

Помимо этого, менеджер может осуществлять следующие действия:

* просмотр каталога реализованных туров;
* добавление нового реализованного тура;
* удаление тура из католога;
* просмотр описания тура;
* просмотр затрат по туру;
* расчёт дохода по туру;
* расчёт прибыли по туру;
* просмотр статистики прибыльности туров.

Диаграмма вариантов использования представлена на рисунке 1.4.

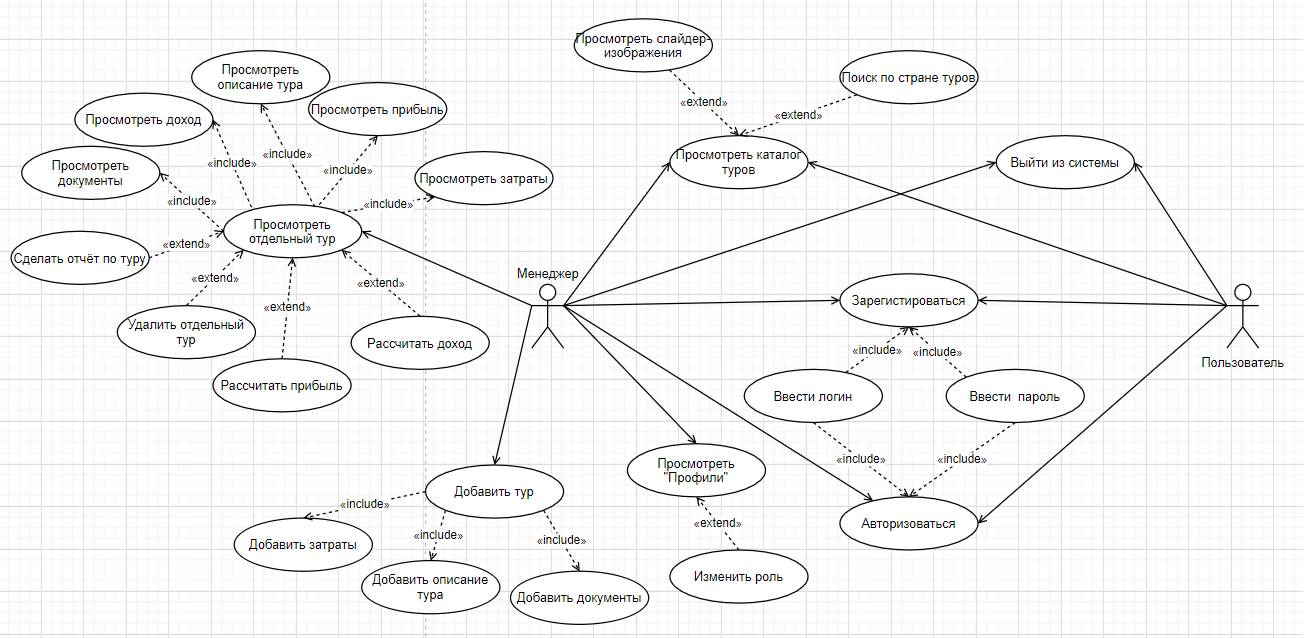


Рисунок 1.4 – Диаграмма вариантов использования (Use Case)

* 1. **Разработка информационной модели предметной области**

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 аэропорт. 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 [7].

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 [8].

Общая схема связанных сущностей базы данных представлена на рисунке 1.5.

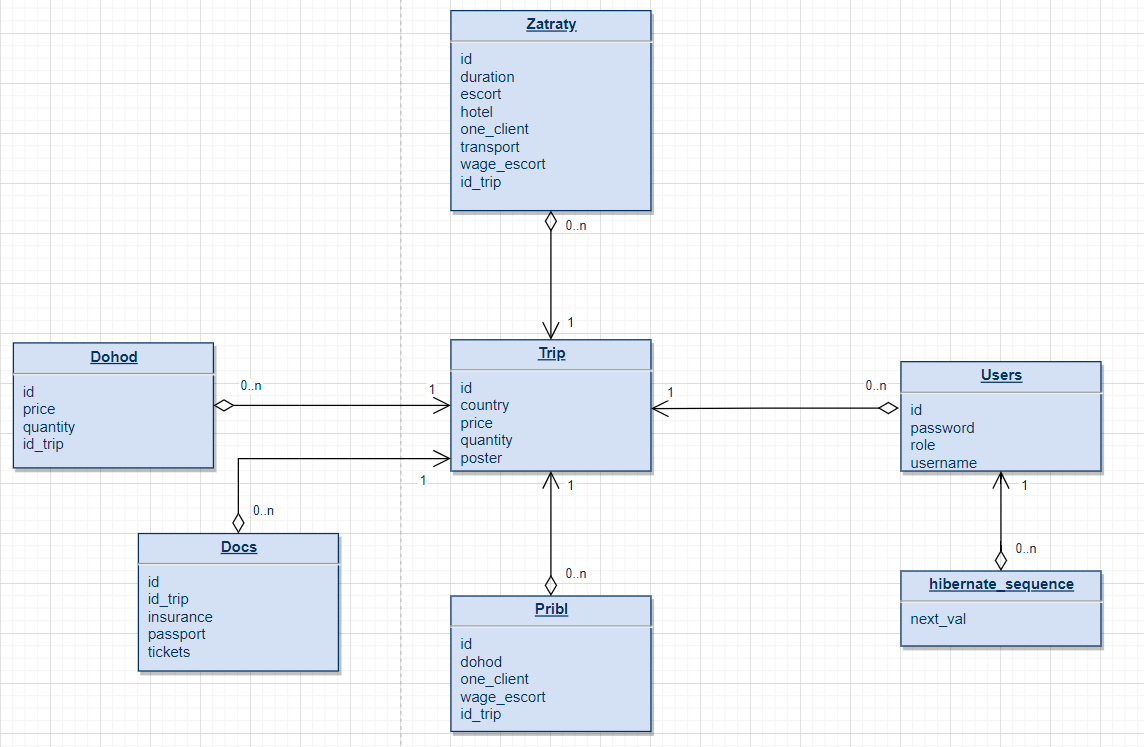


Рисунок 1.5 – Информационная модель базы данных

Данная модель состоит из 6 взаимосвязанных таблиц:

* *zatraty;*
* *trip;*
* *dohod;*
* *users;*
* *docs;*
* *hibernate\_sequence;*
* *pribl.*

Таблица «*Zatraty*» необходима для хранения информации o затратах, связана с таблицей «*Trip*» связью 1: М и содержит в себе:

* *id* – данное поле хранит идентификационный номер затрат;
* *duration* – данное поле хранит продолжительность тура;
* *escort* – данное поле хранит ФИО сопровождающего;
* *hotel* – данное поле хранит отель;
* *one\_client* – данное поле хранит затраты на одного клиента;
* *wage\_escort* – данное поле хранит ЗП сопровождающего;
* *id\_trip* – данное поле хранит идентификатор тура;
* *transport* – данное поле хранит транспорт.

Таблица «*trip*» необходима для хранения информации о турах, связана со всеми таблицами связью 1: М и содержит в себе:

* *id* – данное поле хранит идентификационный номер тура;
* *country* – данное поле хранит страну;
* *price* – данное поле хранит цену тура;
* *quontity* – данное поле хранит количество реализованных туров;
* *price* – данное поле хранит цену тура.

Таблица «*Pribl*» необходима для хранения информации о прибыли тура, связана с таблицой «*trip»* связью 1: М и содержит в себе:

* *id* – данное поле хранит идентификационный номер прибыли;
* *one\_client* – данное поле хранит затраты на одного клиента;
* *dohod* – данное поле хранит сумму заказа в системе;
* *wage\_escort* – данное поле хранит ЗП сопровождающего;
* *id\_trip* – данное поле хранит идентификатор тура.

Таблица «*Users*» необходима для хранения информации о всех пользователях, связана с таблицей «*Trip*» связью 1: М и содержит в себе:

* *id* – данное поле хранит идентификационный номер пользователя в системе;
* *password* – данное поле хранит пароль пользователя в системе;
* *role* – данное поле хранит тип роли пользователя в системе;
* *username* – данное поле хранит логин пользователя в системе.

Таблица «*Hibernate\_sequence*» хранит уникальные ключи ко всем таблицам базы данных.

Таблица «*Docs*» необходима для хранения информации о документах, связана с таблицей «*Trip*» связью 1: М и содержит в себе:

* *id* – данное поле хранит идентификационный номер документов;
* *ensurance* – данное поле хранит страховку клиента;
* *passport* – данное поле хранит пасспорт клиента;
* *tickets* – данное поле хранит билеты;
* *id\_trip* – данное поле хранит идентификатор тура.

Таблица «*Dohod*» необходима для хранения информации о прибыли тура, связана с таблицой «*trip»* связью 1: М и содержит в себе:

* *id* – данное поле хранит идентификационный номер дохода;
* *price* – данное поле хранит цену тура;
* *quantity* – данное поле хранит количество туров в системе;
* *id\_trip* – данное поле хранит идентификатор тура.

Между таблицами могут существовать следующие связи:

* многие ко многим;
* один ко одному;
* один ко многим.

Связь один к одному образуется, когда ключевой столбец присутствует в другой таблице, в которой тоже является ключом либо одно и тоже значение не может повторяться в разных строках.

В типе связей один ко многим одной записи первой таблицы соответствует несколько записей в другой таблице.

Если нескольким записям из одной таблицы соответствует несколько записей из другой таблицы, то такая связь называется «многие ко многим» и организовывается посредством связывающей таблицы.

После проведения построений таблиц и распределения связей между ними, мы получили приведенную к третьей нормальной форме базу данных.

* 1. **Модели представления программного средства и их описание**

Под моделью программного обеспечения понимают формализованное описание системы программного обеспечения на уровне абстракции. Каждая модель определяет конкретный аспект системы, использует набор диаграмм и документов заданного формата, а также отражает точку зрения и является объектом деятельности различных людей с конкретными интересами, ролями или задачами. Далее будут рассмотрены диаграммы последовательности, состояния, классов, компонентов и развертывания компонентов.

Диаграмма последовательности наглядно отображает временной аспект взаимодействия. Она имеет два измерения. Одно измерение (слева-направо) указывает на порядок вовлечения экземпляров сущностей во взаимодействие. Крайним слева на диаграмме отображается экземпляр актера или объект, который является инициатором взаимодействия. Правее отображается другой экземпляр сущности, который непосредственно взаимодействует с первым и т.д. Второе измерение (сверху-вниз) указывает на порядок обмена сообщениями. Начальному моменту времени соответствует самая верхняя часть диаграммы. Масштаб на оси времени не указывается, поскольку диаграмма отображает лишь временную упорядоченность взаимодействия типа «раньше-позже».

Диаграмма последовательности данного курсового проекта представлена на рисунке 1.6.

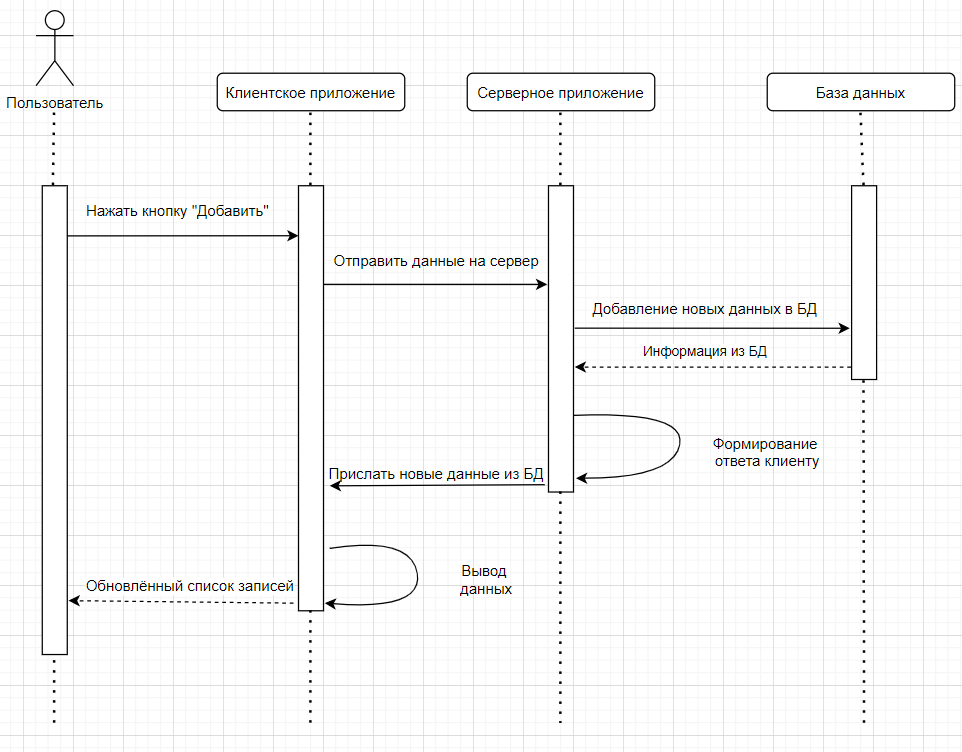


Рисунок 1.6 – Диаграмма последовательности

Диаграмма состояний при входе пользователя в систему представлена на рисунке 1.7. На ней отображено взаимодействие между клиентом и сервером в ходе использования программы.

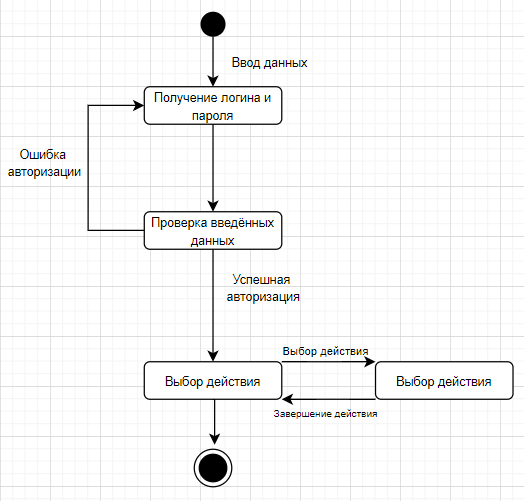


Рисунок 1.7 – Диаграмма состояний

Диаграмма состояний при входе пользователя в систему дает подробный отчет о том, как все происходит от ввода данных, до обработки данных на сервере. Рассмотренная выше диаграмма классов представляет собой логическую модель статического представления моделируемой системы.

Диаграмма классов представлена на рисунке 1.8.

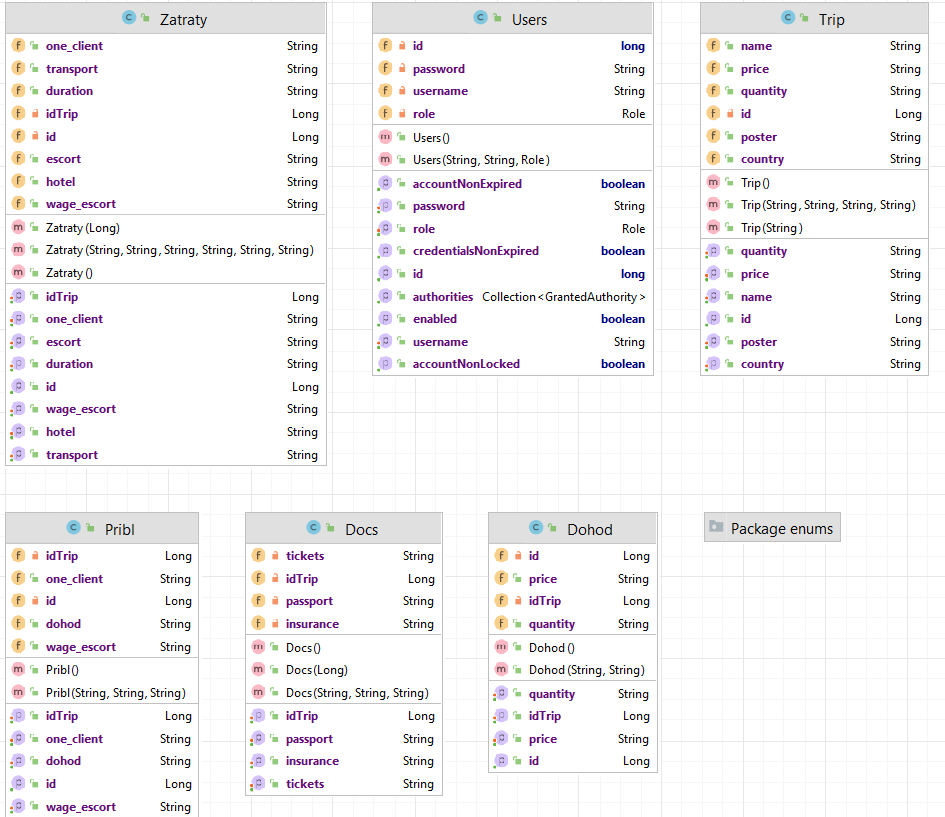


Рисунок 1.8 – Диаграмма классов

Диаграмма используется во многих случаях, особенно может помочь при поиске ошибок. Диаграмма классов занимает центральное место в проектировании объектно-ориентированной системы. Нотация классов используется на разных этапах проектирования и строится с различной степенью детализации.

Диаграмма компонентов позволяет определить состав программных компонентов, в роли которых может выступать исходный, бинарный и исполняемый код, а также установить зависимости между ними. Диаграмма компонентов представлена на рисунке 1.9.

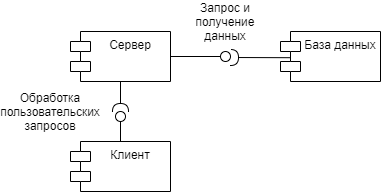


Рисунок 1.9 – Диаграмма компонентов

Диаграмма развертывания представлена на рисунке 1.10. Графическое представление ИТ-инфраструктуры может помочь более рационально распределить компоненты системы по узлам сети, от чего зависит в том числе и производительность системы.

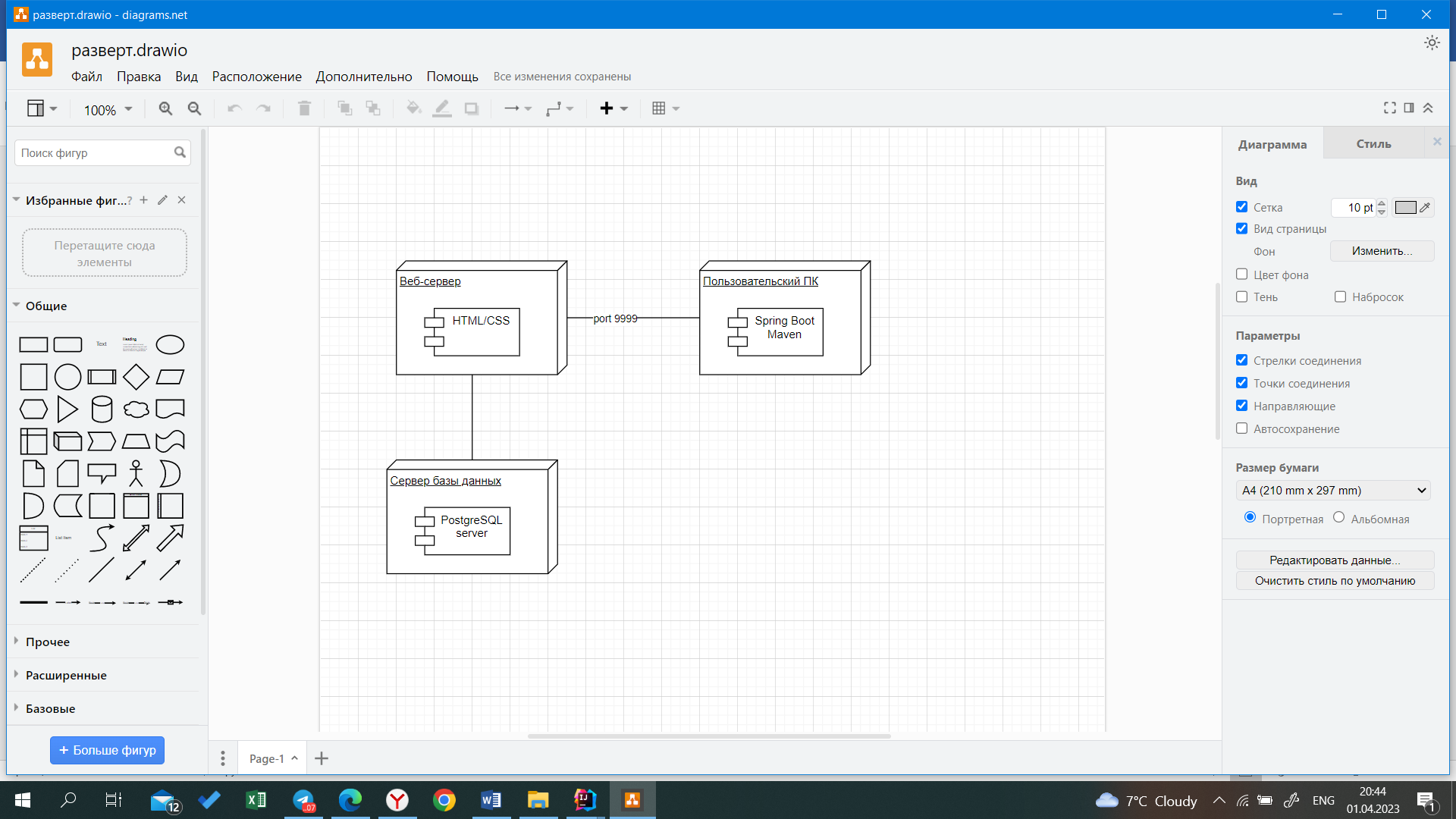


Рисунок 1.10 – Диаграмма развертывания компонентов

Рассматривая компоненты системы и их взаимодействие можно сказать, что структура приложения представляет собой:

− клиентскую часть (страница в веб-браузере);

− серверную часть (программа на сервере);

− приложение СУБД (отдельная СУБД на сервере).

1. **ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**
   1. **Постановка задачи**

Целью курсового проекта является разработка АРМ для оптимизации деятельности турфирмы за счет автоматизации процесса учёта и контроля туров, клиентов, ресурсов, а также процесса составления статистики о реализованных турах.

Для достижения поставленной цели будет разработано программное обеспечение, которое поможет менеджерам быстро искать записи и составлять отчёт о реализованных турах. Программное обеспечение будет представлено в виде web-сайта, предоставляющее соответствующий функционал в зависимости от роли вошедшего пользователя.

Любое из аналогичных приложение должно разделять пользователей как администратора и обычного пользователя. Поэтому на первом этапе стоят задачи, такие как:

– регистрация пользователей;

– выбор роли пользователя при регистрации;

– авторизация пользователей;

Далее, исходя из полученной информации можно определить, что информационная система должна предоставлять возможность администратору:

– создание реализованных туров;

– редактирование реализованных туров;

– удаление реализованных туров;

– просмотр статистики.

Пользователи в свою очередь должны иметь возможность просматривать список всех реализованных туров.

Задача данной программы заключается в обеспечении оптимизации деятельности менеджера турфирмы. Необходимо, чтобы программа была построена на основе клиент-серверной архитектуры.

В приложении необходимо реализовать:

–  авторизация/регистрация пользователей (для разделения прав доступа администратору и пользователю);

–  пользователь должен видеть всю имеющуюся информация о нужных ему турах;

–  администратор должен иметь возможность добавлять, редактировать и удалять уже хранящуюся в базе данных информацию, а также составлять статистические отчёты.

* 1. **Архитектурные решения**

Архитектура программного обеспечения системы отображает организацию или структуру системы и дает объяснение того, как она ведет себя. Система представляет собой набор компонентов, выполняющих определенную функцию или набор функций. Другими словами, архитектура программного обеспечения обеспечивает прочную основу, на которой может быть построено программное обеспечение.

Ряд архитектурных решений и компромиссов влияют на качество, производительность, ремонтопригодность и общий успех системы. Неспособность учесть общие проблемы и долгосрочные последствия может подвергнуть систему риску.

Существует множество архитектурных схем и принципов высокого уровня, обычно используемых в современных системах. Их часто называют архитектурными стилями. Архитектура программной системы редко ограничивается одним архитектурным стилем. Вместо этого, комбинация стилей часто составляет полную систему.

Одно из популярных архитектурных схем является клиент-серверная архитектура, которая и была использована в курсовом проекте.

Архитектура «клиент-сервер» определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций (сервисов) и клиенты (потребители этих функций). Практические реализации такой архитектуры называются клиент-серверными технологиями.

Клиент – это браузер, но встречаются и исключения (в тех случаях, когда один веб-сервер (ВС1) выполняет запрос к другому (ВС2), роль клиента играет веб-сервер ВС1). В классической ситуации (когда роль клиента выполняет браузер) для того, чтобы пользователь увидел графический интерфейс приложения в окне браузера, последний должен обработать полученный ответ веб-сервера, в котором будет содержаться информация, реализованная с применением HTML, CSS, JS (самые используемые технологии). Именно эти технологии «дают понять» браузеру, как именно необходимо «отрисовать» все, что он получил в ответе.

Веб-сервер – это сервер, принимающий HTTP-запросы от клиентов и выдающий им HTTP-ответы. Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и непосредственно компьютер, на котором это программное обеспечение работает. Наиболее распространенными видами ПО веб-серверов являются Apache, IIS и NGINX. На веб-сервере функционирует тестируемое приложение, которое может быть реализовано с применением самых разнообразных языков программирования: PHP, Python, Ruby, Java, Perl и другие.

База данных фактически не является частью веб-сервера, но большинство приложений просто не могут выполнять все возложенные на них функции без нее, так как именно в базе данных хранится вся динамическая информация приложения.

База данных (БД) – это организованная структура, предназначенная для хранения информации. Обычно БД представляются в виде совокупности взаимосвязанных файлов или таблиц, предназначенных для решения конкретной задачи. Базы данных функционируют под управлением так называемых систем управления базами данных (далее – СУБД). Самыми популярными СУБД являются MySQL, MS SQL Server, PostgreSQL, Oracle, SQLite.

* 1. **Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства**

Бизнес-логика – это система бизнес-правил, описывающих поведение объектов и процессов предметной области, а также представляющая эти процессы в информационных системах. Бизнес-логика определяет методы и алгоритмы анализа данных, а также способы передачи его результатов пользователям.

Примером бизнес-логики является процесс «встречи» клиента, посетившего сайт компании. Она включает запрос имени и пароля, вывод приветственной надписи, отображение персональных предложений (если есть), вывод поздравления, если посещение происходит в праздник или день рождения клиента, вывод предложения добавить товары в корзину и информации о методах оплаты. В то же время высказывание о необходимости «встречи» клиента является бизнес-правилом.

Можно сказать, что все, что является процессом, процедурой – можно отнести к бизнес-логике. Напротив, все, что процессом и процедурой не является – является бизнес-правилами. Таким образом, бизнес-логика носит процедурный характер, а бизнес-правила – декларативный.

В области разработки программного обеспечения бизнес-логикой также называются реализующие ее программные модули и уровни системы, на которых эти модули расположены.

В фазе бизнес-моделирования и разработки требований бизнес-логика может описываться в виде:

– текста;

– концептуальных аналитических моделей предметной области (онтологии);

– бизнес-правил;

– разнообразных алгоритмов;

– диаграмм деятельности;

– графов и диаграмм перехода состояний;

– моделей бизнес-процессов.

В фазе анализа и проектирования системы бизнес-логика воплощается в различных диаграммах языка UML или ему подобных. В фазе программирования бизнес-логика воплощается в коде классов и их методов в случае использования объектно-ориентированных языков программирования, или процедур и функций в случае применения процедурных языков.

Одним из самых важных процессов в программе является авторизация. Блок-схема этого процесса представлена на рисунке 2.1.

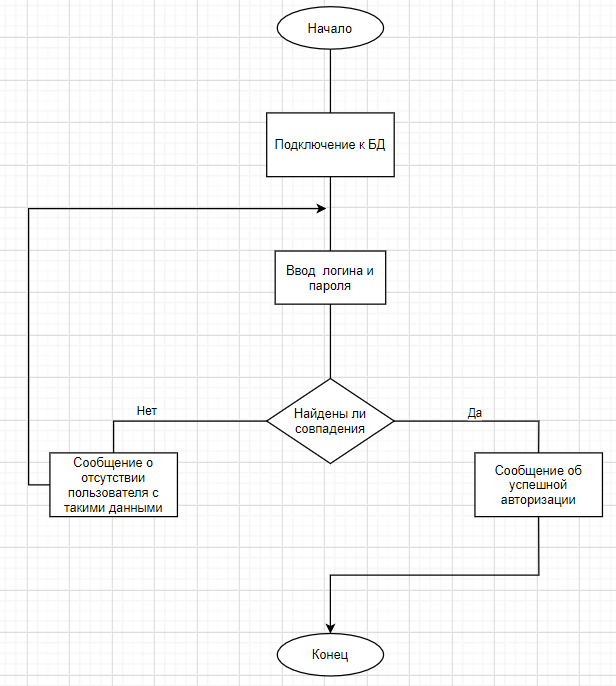


Рисунок 2.1 – Блок-схема алгоритма авторизации

* 1. **Проектирование пользовательского интерфейса**

Цель проектирования пользовательского интерфейса – получить четкое видение того, каким должен быть интерфейс системы. Потому что, это инструмент, который помогает выполнению бизнес-задач и создает конкурентное преимущества, а также даёт разработчикам четкие инструкции по поводу того, как делать систему. Важно ответить на ключевые вопросы: для кого и для чего эта система, кто ее основная аудитория и какие задачи этой аудитории она решает, с какими целями создается продукт и какие задачи стоят перед ним, что является важным для успеха продукта, а что – второстепенным?

Первый шаг проектирования – это предпроектный анализ. Нужно понять, что требуется сделать. После этого можно уже достаточно точно оценить и спланировать работы.

Основной документ, который получается в итоге проектирования – детальные схемы страниц. Они показывают, что представляет из себя каждая страница системы, каковы особенности ее работы. После этого можно начинать работу над дизайном. В дополнение можно подготовить описание работы страниц – сценарии взаимодействия. В них детально описан алгоритм работы пользователя с сайтом.

Но самый лучший результат проектирования – интерактивный прототип системы. Это действующая модель пользовательского интерфейса, в него включены основные страницы и процессы работы системы.

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

**2.5.1 Spring framework**

Spring – один из наиболее широко используемых фреймворков для разработки интерпрайз-приложений, обеспечивающий продуманную модель программирования и конфигурации. Целью создания данного фреймворка способствовало желание упростить разработку приложений на популярном в то время (да и сейчас) Java EE стеке технологий от компании Oracle, который на тот момент был очень сложен и не всегда удобен в использовании.

В отличие от других платформ, Spring фокусируется на нескольких областях функционирования приложений и предоставляет для них широкий спектр дополнительных функций.

Spring не изобретает велосипед, вместо этого он действительно использует некоторые из существующих технологий, такие как несколько фреймворков ORM, таймеры JEE, Quartz и JDK и другие технологии просмотра.

Веб-фреймворк Spring – это хорошо спроектированный веб-фреймворк MVC, который обеспечивает отличную альтернативу таким веб-фреймворкам, как Struts или другим излишне спроектированным или менее популярным веб-фреймворкам.

Spring предоставляет удобный API для преобразования исключений, связанных с конкретной технологией (например, вызванных JDBC, Hibernate или JDO), в согласованные, непроверенные исключения.

Spring предоставляет согласованный интерфейс управления транзакциями, который может масштабироваться до локальной транзакции (например, с использованием одной базы данных) и масштабироваться до глобальных транзакций (например, с использованием JTA).

Технология, с которой больше всего ассоциируется Spring, – это инверсия управления зависимостями (DI). Инверсия управления (IoC) является общей концепцией, и она может быть выражена по-разному. DI – это лишь один конкретный пример инверсии управления.

При написании сложного приложения Java классы приложения должны быть как можно более независимыми от других классов Java, чтобы увеличить возможность повторного использования этих классов и их тестирования независимо от других классов во время модульного тестирования. Инъекция зависимостей помогает склеивать эти классы вместе и в то же время сохранять их независимость.

Одним из ключевых компонентов Spring является структура аспектно-ориентированного программирования (АОП). Функции, охватывающие несколько точек приложения, называются сквозными задачами, и эти сквозные задачи концептуально отделены от бизнес-логики приложения. Существуют различные общие хорошие примеры аспектов, включая ведение журнала, декларативные транзакции, безопасность, кеширование и т. д.

Ключевой единицей модульности в ООП является класс, тогда как в АОП единицей модульности является аспект. DI помогает отделить объекты приложения друг от друга, а AOП помогает отделить сквозные проблемы от объектов, на которые они влияют.

Модуль AOП Spring Framework предоставляет реализацию аспектно-ориентированного программирования, позволяющую определять методы-перехватчики для четкого разделения кода, реализующего функциональность, которая должна быть разделена.

**2.5.2 Hibernate**

Hibernate – это ORM фреймворк для Java с открытым исходным кодом. Эта технология является крайне мощной и имеет высокие показатели производительности.

Hibernate создаёт связь между таблицами в базе данных (далее – БД) и Java-классами и наоборот. Это избавляет разработчиков от огромного количества лишней, рутинной работы, в которой крайне легко допустить ошибку и крайне трудно потом её найти.

Какие же преимущества даёт использование Hibernate:

1. Hibernate устраняет множество спагетти кода (повторяющегося), который постоянно преследует разработчика при работе с JDBC. Скрывает от разработчика множество кода, необходимого для управления ресурсами и позволяет сосредоточиться на бизнес логике.
2. Hibernate поддерживает XML также как и JPA аннотации, что позволяет сделать реализацию кода независимой.
3. Hibernate предоставляет собственный мощный язык запросов (HQL), который похож на SQL. Стоит отметить, что HQL полностью объектно-ориентирован и понимает такие принципы, как наследование, полиморфизм и ассоциации (связи).
4. Hibernate – широко распространенный open source проект. Благодаря этому доступны тысячи открытых статей, примеров, а также документации по использованию фреймворка.
5. Hibernate легко интегрируется с другими Java EE фреймворками, например, Spring Framework поддерживает встроенную интеграцию с Hibernate.
6. Hibernate поддерживает ленивую инициализацию используя proxy объекты и выполняет запросы к базе данных только по необходимости.
7. Hibernate поддерживает разные уровни cache, а следовательно, может повысить производительность.
8. Важно, что Hibernate может использовать чистый SQL, а значит поддерживает возможность оптимизации запросов и работы с любым сторонним вендором БД и его фичами.

Hibernate поддерживает все основные СУБД: MySQL, Oracle, PostgreSQL, Microsoft SQL Server Database, HSQL, DB2.  
Hibernate также может работать в связке с такими технологиями, как Maven и J2EE.

Hibernate поддерживает такие API, как JDBC, JNDI, JTA.  
JDBC обеспечивает простейший уровень абстракции функциональности для реляционных БД. JTA и JNDI, в свою очередь, позволяют Hibernate использовать серверы приложений J2EE.

Рассмотрим отдельно каждый из элементов Hibernate:

– *Transaction*: этот объект представляет собой рабочую единицу работы с БД. В Hibernate транзакции обрабатываются менеджером транзакций;

– *SessionFactory*: самый важный и самый тяжёлый объект (обычно создаётся в единственном экземпляре, при запуске приложения). Нам необходима как минимум одна SessionFactory для каждой БД, каждый из которых конфигурируется отдельным конфигурационным файлом;

– *Session*: сессия используется для получения физического соединения с БД. Обычно, сессия создаётся при необходимости, а после этого закрывается. Это связано с тем, что эти объекты крайне легковесны. Чтобы понять, что это такое, модно сказать, что создание, чтение, изменение и удаление объектов происходит через объект Session;

– *Query*: этот объект использует HQL или SQL для чтения/записи данных из/в БД. Экземпляр запроса используется для связывания параметров запроса, ограничения количества результатов, которые будут возвращены и для выполнения запроса;

– *Configuration*: этот объект используется для создания объекта SessionFactory и конфигурирует сам Hibernate с помощью конфигурационного XML-файла, который объясняет, как обрабатывать объект Session;

– *Criteria*: используется для создания и выполнения объектно-ориентированных запроса для получения объектов.

**2.5.3 PostgreSQL**

PostgreSQL - это бесплатная и открытая реляционная база данных с отличной репутацией в сообществе разработчиков. Она доступна для многих операционных систем, включая Linux, Windows, macOS и др. PostgreSQL была выпущена в 1989 году и на данный момент является одной из самых надежных и масштабируемых баз данных, которые можно использовать в различных проектах.

Ниже приведены некоторые преимущества PostgreSQL:

* Открытый и бесплатный исходный код: PostgreSQL имеет открытый и бесплатный исходный код, что позволяет пользователям настраивать и изменять его под свои потребности. Кроме того, существует большое сообщество пользователей и разработчиков, которые могут помочь с проблемами и дополнительными функциями;
* расширяемость: PostgreSQL может быть расширена с помощью множества модулей, которые позволяют использовать дополнительные типы данных, функции и языки программирования, такие как Python, Java и другие;
* высокая надежность: PostgreSQL известна своей высокой надежностью и устойчивостью. Она поддерживает множество функций, таких как транзакции, которые позволяют откатывать изменения, если что-то пошло не так. PostgreSQL также имеет механизмы резервного копирования и восстановления, что позволяет быстро восстанавливать данные в случае сбоев;
* масштабируемость: PostgreSQL может обрабатывать большие объемы данных и масштабироваться на большое количество пользователей и запросов. PostgreSQL поддерживает партиционирование, что позволяет разбивать таблицы на несколько фрагментов для более эффективного управления большими объемами данных;
* поддержка полнотекстового поиска: PostgreSQL имеет встроенную поддержку полнотекстового поиска, что позволяет искать данные в текстовых полях с использованием сложных запросов и функций;
* соответствие стандартам: PostgreSQL соответствует стандартам ANSI SQL и SQL:2008, что означает, что она совместима с большинством стандартных инструментов и приложений, которые используют SQL.

В целом, PostgreSQL является отличным выбором для любого проекта, требующего надежной и масштабируемой базы данных. Она имеет широкий набор функций и возможностей, которые позволяют эффективно управлять данными и обеспечивать высокую производительность.

Кроме того, PostgreSQL поддерживает множество языков программирования и инструментов разработки, что делает ее удобной для использования в различных проектах. Например, PostgreSQL можно использовать с популярными языками программирования, такими как Python, Java, C++, Ruby и другими.

Также стоит отметить, что PostgreSQL имеет хорошо развитое сообщество пользователей и разработчиков, что обеспечивает поддержку и обновления базы данных. Это позволяет пользователям получать поддержку и советы, если возникают проблемы с использованием PostgreSQL.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Антонова, А.С. Автоматизированные информационные системы, базы и банки данных. Вводный курс: Учебное пособие [текст] / А.С. Антонова. - М.: Гелиос АРВ, 2014. - 368 с.

[2] CRM-системы [Электронный ресурс]. – Режим доступа: https://www.tourbc.ru/tehnologii/obzory/537-crm-dlya-turagentstva.html

[3] Приобретение ИС [Электронный ресурс]. – Режим доступа: https://studbooks.net/2028558/informatika/vybor\_obosnovanie\_sposoba\_priobreteniya\_avtomatizatsii\_zadachi

[4] Методология SADT [Электронный ресурс]. – Режим доступа: http://www.kgau.ru/istiki/umk/mbp/ch06s07.html

[5] Методология IDEF0 [Электронный ресурс]. – Режим доступа: https://www.businessstudio.ru/wiki/docs/current/doku.php/ru/csdesign/bpmodeling/idef0

[6] Моделирование и анализ бизнес-процессов [Электронный ресурс]. – Режим доступа: https://analytics.infozone.pro/business-modeling-idef-uml-aris/

[7] Базы данных. Лекции [Электронный ресурс]. – Режим доступа: https://sites.google.com/site/anisimovkhv/learning/pris/lecture

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