

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
Ассистент кафедры ЭИ
_____ В.В. Ничепорук
_____._____.2022

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему:
**«РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЗИРОВАННОЙ РАБОТЫ
ПАРИКМАХЕРСКОЙ»**

Выполнил студент группы 024402
Котлярова Яна Игоревна
(подпись студента)
Курсовой проект представлен на
проверку _____._____.2022

(подпись студента)

СОДЕРЖАНИЕ

Введение.....	5
1 Описание работы парикмахерской.....	6
1.1 Обзор работы парикмахерской.....	6
1.2 Обзор программных аналогов	7
2 Постановка задачи и обзор методов её решения	9
2.1 Постановка задачи	9
2.2 Функциональные возможности программного обеспечения	10
2.3 Обоснование решений по использованию технических и программных средств реализации	11
3 Функциональное моделирование на основе стандарта IDEF0.....	12
4 Информационная модель системы и её описание	19
5 Описание алгоритмов, реализующих бизнес-логику серверной части системы автоматизации работы парикмахерской	22
5.1 Алгоритмы, реализующие бизнес-логику серверной части.....	22
5.2 Диаграмма вариантов использования	26
5.3 Диаграмма состояний	26
5.4 Диаграмма последовательности	26
5.5 Диаграммы классов.....	27
5.6 Диаграммы компонентов и развёртывания.....	31
6 Руководство пользователя.....	33
6.1 Руководство для клиента.....	33
6.2 Руководство для сотрудника.....	37
7 Проверка работоспособности приложения	47
Заключение	49
Список использованных источников	50
Приложение А (обязательное) Диаграмма вариантов использования	51
Приложение Б (обязательное) Диаграмма состояний	52
Приложение В (обязательное) Диаграмма последовательности.....	53
Приложение Г (обязательное) Диаграммы классов.....	54
Приложение Д (обязательное) Скрипт создания базы данных	58
Приложение Е (обязательное) Листинг программного кода.....	61

ВВЕДЕНИЕ

Автоматизация является логичным направлением научно-технического процесса, подразумевающим использование автоматических и автоматизированных устройств, работающих частично или полностью без участия человека.

Человечество давно решило перейти от работы, выполняемой вручную, на использование машинной техники, так как автоматизация позволяет повысить производительность труда, улучшить качество продукции, оптимизировать процессы управления, отстранить человека от производств, опасных для здоровья. Сейчас создаётся всё больше программ, сервисов, интернет-магазинов, различных систем и подобного программного обеспечения для упрощения жизни людей и их удобства. Причиной этого становится превосходство машинных программ над человеком в объёмах обрабатываемой информации за одну и ту же единицу времени, точности расчётов, скорости обработки информации.

Поэтому целью курсового проекта является улучшение работы парикмахерской путём автоматизации организационных процессов, таких как запись на услугу и составление расписаний.

Для реализации цели курсового проекта были сформулированы следующие задачи:

- исследование основных аспектов деятельности предприятия;
- определить задачу и рассмотреть методы для её реализации;
- выполнить функциональное моделирование процессов работы парикмахерской;
- выполнить информационное моделирование системы и описать её;
- описать алгоритмы, реализующие бизнес-логику серверной части проектируемой системы;
- разработать руководство пользователя;
- разработать приложение для работы программы;
- разработать интерфейс окна, для решения задачи выбора операции;
- организовать работу с базой данных;
- протестировать программное средство.

Конечная программа позволит автоматизировать и оптимизировать работу парикмахерской, позволяя клиентам записываться на необходимые процедуры, а работникам отслеживать данные и составлять расписания.

1 ОПИСАНИЕ РАБОТЫ ПАРИКМАХЕРСКОЙ

1.1 Обзор работы парикмахерской

Парикмахерская – представляет собой государственное или частное предприятие, в котором осуществляется уход за волосами в оборудованном специально для этого помещении.

К услугам парикмахера относятся стрижка, завивка, плетение кос, дредов и создание сложных причёсок, укладка волос (совокупность операций, позволяющих придать волосам ту или иную форму с тем или иным рисунком на непродолжительное время), окрашивание, мелирование (окрашивание волос отдельными прядями) и другие виды работ с красителями, бритьё и стрижка бород и усов, лечение волос, наращивание (парикмахерская процедура добавления к естественным волосам человека дополнительных прядей для придания длины и объема), ламинирование (механическое воздействие на поверхность волоса с помощью специального выпрямляющего состава), коррекция формы причёски. Некоторые парикмахеры специализируются на постижерных работах – изготовлении париков. В парикмахерских дополнительно оказываются следующие виды услуг: маникюр (уход за пальцами рук, чистка, полировка, покрытие лаком ногтей), педикюр (уход за ступнями и пальцами ног, чистка и полировка ногтей, удаление мозолей), косметические услуги (услуга по уходу за кожей лица и тела путем физического и химического воздействия с использованием средств декоративной косметики) и услуги визажиста.

Парикмахер может быть специалистом по женским, мужским или детским прическам, или парикмахером-универсалом. Но больше всего ценятся парикмахеры-стилисты. Они не просто приводят волосы в порядок, но и меняют образ, они способны с помощью прически подчеркнуть достоинства и скрыть недостатки клиента. Парикмахер владеет современными парикмахерскими технологиями. При необходимости может проконсультировать клиента по вопросам ухода за новой прической (например, африканскими косичками, дредами), посоветовать возможные способы укладки новой стрижки, а также средства для оздоровления волос и кожи головы. Парикмахер легко ориентируется в многообразии своих профессиональных инструментов. Например, при создании одной модельной стрижки он может применять разные виды ножниц, филировочную (опасную) бритву, машинку для стрижки волос. В арсенале парикмахера есть десятки

видов расчесок, бигуди, различные электрические приборы, заколки и декоративные элементы.

1.2 Обзор программных аналогов

Одним из аналогов создаваемой программы является «Beauty Pro».

«Beauty Pro» – программа для салона красоты. Позволяет создать эффективную систему повторных продаж. Использовать информацию о прошлых визитах клиента, чтобы своевременно предложить актуальные для него услуги и товары. Увеличивать показатели по продажам и свою прибыль за счет мотивирования персонала на продажи сопутствующих товаров. Внешний вид программы представлен на рисунке 1.2.

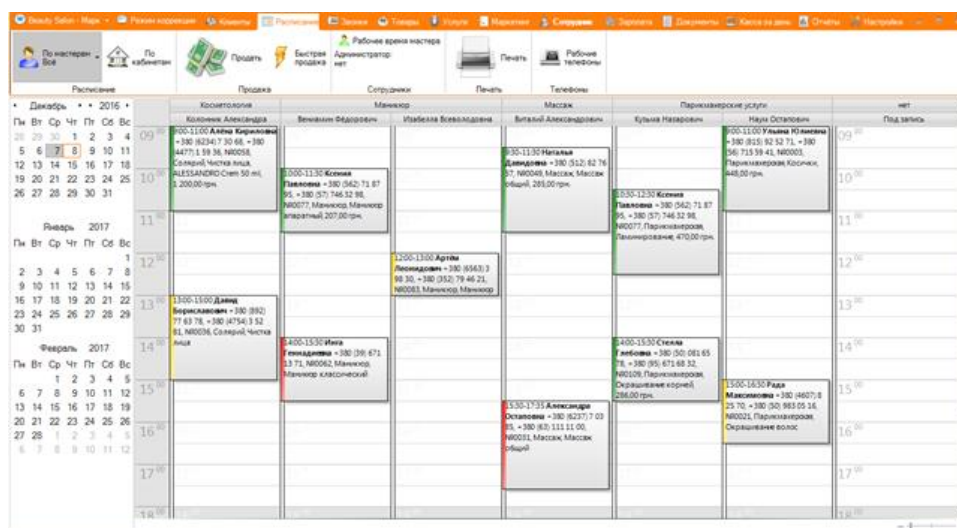


Рисунок 1.2 – Внешний вид программы «Beauty Pro»

Данная программа предоставляет возможность:

- контролировать рентабельность продаж;
- отслеживать наиболее рентабельные услуги;
- продавать больше услуг с помощью абонементов;
- учитывать расходные материалы;
- увеличивать количество повторных продаж;
- применять перекрестные продажи услуг;
- создавать акции и эксклюзивные предложения.

Другим аналогом является программа «YCLIENTS» – программа для автоматизации, рассчитанная на большое количество сфер применения:

медицина, спорт, обучение, индустрия красоты, отдых и досуг, автоуслуги и многие другие. Внешний вид программы представлен на рисунке 1.3.

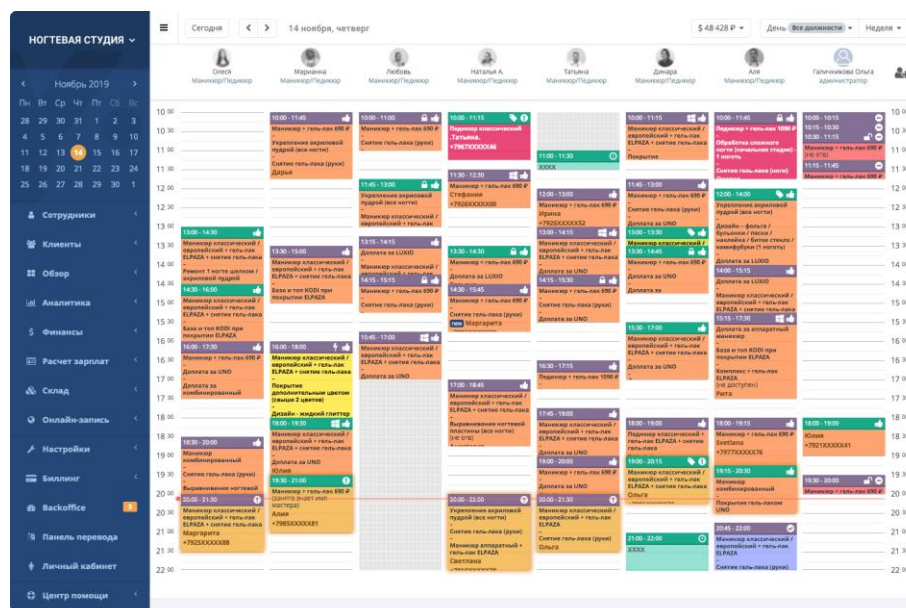


Рисунок 1.3 – Внешний вид программы «YCLIENTS»

Данная программа позволяет исключить ошибки человеческого фактора - история взаимодействия с программой покажет, как ведут учёт сотрудники, предоставит отчет по удаленным записям, поможет выявить недобросовестный персонал и факты мошенничества. Дает возможность удалённого контроля - все финансовые потоки в облачной базе данных в виде наглядных отчетов и схем дают возможность вести дела из любого места при наличии подключения к Интернету. Позволяет получить важнейшие отчёты без ошибок - программа показывает неоплаченные записи и расхождения, а аналитические и финансовые отчеты формируются автоматически. Также трудоемкие задачи, такие, как расчет зарплаты, оповещения клиентов, сведение отчетов, поиск критических остатков программа делает в автоматическом режиме. С программой можно оценить квалификацию персонала не будучи парикмахером. Процент возвратов к мастерам, количество клиентов, выручка, отзывы, отмены, сравнение по мастерам помогут понять, как работают сотрудники. Предусмотрена функция скрытия данных клиентов - можно скрыть в настройках номера телефонов, и будет возможность увидеть только символы XXX, вместо цифр.

Наличие программных аналогов даёт понять, что область автоматизации востребована и продолжает развиваться.

2 ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЁ РЕШЕНИЯ

2.1 Постановка задачи

Для реализации цели курсового проекта были сформулированы следующие задачи:

1 Исследование основных аспектов деятельности предприятия. Исследование основных аспектов деятельности предприятия было проведено в первой главе курсового проекта «Описание работы парикмахерской», поэтому не требует дополнительных пояснений в данном пункте.

2 Определить задачу и рассмотреть методы для её реализации. В данной главе будут рассмотрены задачи, сформулированные для реализации цели курсового проекта, а также методы, для решения этой задачи.

3 Выполнить функциональное моделирование процессов работы парикмахерской. Для функционального описания модели работы программы была выбрана нотация IDEF0. В данной главе находятся созданные схемы, которые описывают работу парикмахерской.

4 Выполнить информационное моделирование системы и описать её. Моделирование данных – это средство, относящееся к бизнес-процессу организации, для формального сбора данных. Моделирование является одним из главных на сегодняшний день приемов анализа, основанием, на котором строятся реляционные базы данных. Для данной главы курсового проекта была создана и описана физическая модель базы данных.

5 Описать алгоритмы, реализующие бизнес-логику серверной части проектируемой системы. В пятой главе курсового проекта были описаны алгоритмы, реализующие бизнес-логику серверной части проектируемой системы.

6 Разработать руководство пользователя. Глава руководство пользователя описывает взаимодействие разных типов пользователей с проектируемой системой и предоставляет наглядный графический материал, иллюстрирующий описываемый функционал.

7 Разработать приложение для работы программы.

8 Разработать интерфейс окна, для решения задачи выбора операции. Разработанный интерфейс окон предоставлен в главе руководство пользователя.

9 Организовать работу с базой данных. Для реализации базы данных была выбрана свободная реализационная система MySQL, реализуемая через программное средство HeidiSQL.

10 Протестировать программное средство. Глава о результатах тестирования разработанной системы предоставляет информацию о проведённых программных тестах с созданным программным средством и о результатах данных тестов, представленных в графическом виде.

2.2 Функциональные возможности программного обеспечения

Работники разделяются на администраторов и специалистов. Администратор должен иметь прямой доступ к базе данных для внесения необходимых изменений и контролирования хранящихся там данных. Специалисты же смогут просматривать свои расписания, составляющиеся при проведении клиентом записи на процедуру, предлагаемую каждым из специалистов.

Для регистрации клиента нужно ввести правильно следующую информацию: логин, пароль, ФИО, номер телефона.

Для регистрации сотруднику необходимо ввести следующую информацию: логин, пароль, ФИО, номер телефона, адрес электронной почты, специализация.

Зарегистрироваться самостоятельно может только клиент, добавление новых сотрудников, как администраторов, так и специалистов производится администратором.

Если сотрудник или клиент уже регистрировался в системе, то он сможет зайти в систему снова, указав логин и пароль при входе в систему.

Администратор в автоматизируемой системной области будет обладать следующими основными возможностями:

- 1 Просмотр и изменение личной информации.
- 2 Просмотр данных о всех пользователях.
- 3 Удаление пользователя из базы данных.
- 4 Добавление сотрудника в базу данных.
- 5 Просмотр расписания специалистов.
- 6 Добавление, удаление и просмотр услуг в базе данных.
- 7 Аналитика по данным.

Специалист в автоматизируемой системной области будет обладать следующими основными возможностями:

- 1 Просмотр и изменение личной информации.
- 2 Добавление, редактирование, просмотр и удаление услуг.
- 3 Просмотр личного расписания.

Клиент в автоматизируемой системной области будет обладать следующими основными возможностями:

- 1 Просмотр и изменение личной информации.
- 2 Добавление, удаление, редактирование и просмотр личных записей.

2.3 Обоснование решений по использования технических и программных средств реализации

Для решения поставленных задач в курсовом проекте используются нижеописанные программные и технические средства.

MySQL – свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle.

MySQL позволяет хранить целочисленные значения со знаком и беззнаковые, длиной в 1, 2, 3, 4 и 8 байтов, работает со строковыми и текстовыми данными фиксированной и переменной длины, позволяет осуществлять SQL-команды SELECT, DELETE, INSERT, REPLACE и UPDATE, обеспечивает полную поддержку операторов и функций в SELECT- и WHERE- частях запросов, работает с GROUP BY и ORDER BY, поддерживает групповые функции COUNT(), AVG(), STD(), SUM(), MAX() и MIN(), позволяет использовать JOIN в запросах, в т.ч. LEFT OUTER JOIN и RIGHT OUTER JOIN, поддерживает репликацию, транзакции, работу с внешними ключами и каскадные изменения на их основе, а также обеспечивает многие другие функциональные возможности.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей.

JavaFX – платформа на основе Java для создания приложений с насыщенным графическим интерфейсом. С его помощью можно создавать программы для различных операционных систем: Windows, MacOS, Linux и для самых различных устройств: десктопы, смартфоны, планшеты, встроенные устройства, ТВ. Приложение на JavaFX будет работать везде, где установлена исполняемая среда Java (JRE).

JavaFX предоставляет большие возможности по сравнению с рядом других подобных платформ, в частности, по сравнению со Swing. Это и большой набор элементов управления, и возможности по работе с мультимедиа, двухмерной и трехмерной графикой, декларативный способ описания интерфейса с помощью языка разметки FXML.

3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF0

Для функционального описания модели работы программы была выбрана нотация IDEF0. Данная функциональная модель представлена в виде набора блоков, каждый из которых в свою очередь представлен в виде чёрных прямоугольных областей с входами и выходами, управлением и механизмами, которые уточняются и раскрываются до необходимого уровня сложности и детальности. Данная модель даёт возможность описать все основные виды процессов, представленные в данной программе [1].

На рисунке 3.1 изображена контекстная диаграмма верхнего уровня модели «Работа парикмахерской», а также определены входные и выходные потоки данных, механизмы ограничения и управления данными.

Входной поток включает в себя добавление нового специалиста, добавление новой услуги, запрос на услугу и приём клиента. Выходной поток включает в себя расписание специалиста, созданную услугу, запись на услугу и оказание услуги. Механизмами управления являются администратор, специалист, клиент и автоматизированная система. Механизмами ограничения являются имеющиеся услуги, имеющиеся записи, свободные дата и время, клиент пришёл вовремя, принятие нового специалиста.

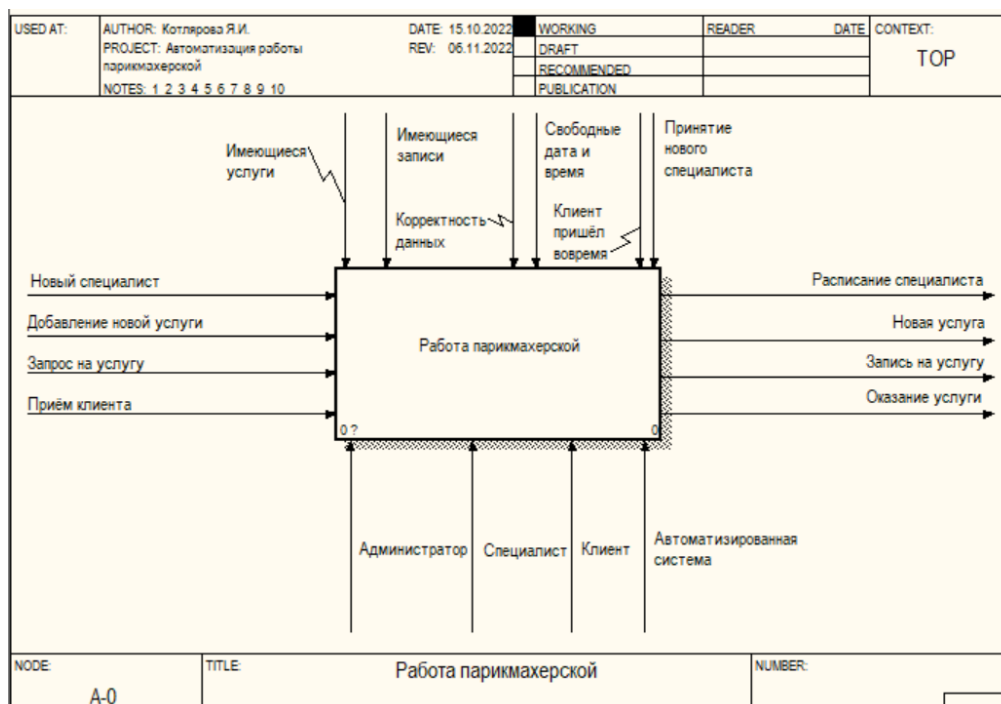


Рисунок 3.1 – Контекстная диаграмма верхнего уровня

На рисунке 3.2 изображена декомпозиция контекстной диаграммы верхнего уровня, состоящая из четырёх блоков: «Добавить специалиста в систему», «Добавить услугу в систему», «Необходимость в услуге» и «Выполнить услугу».

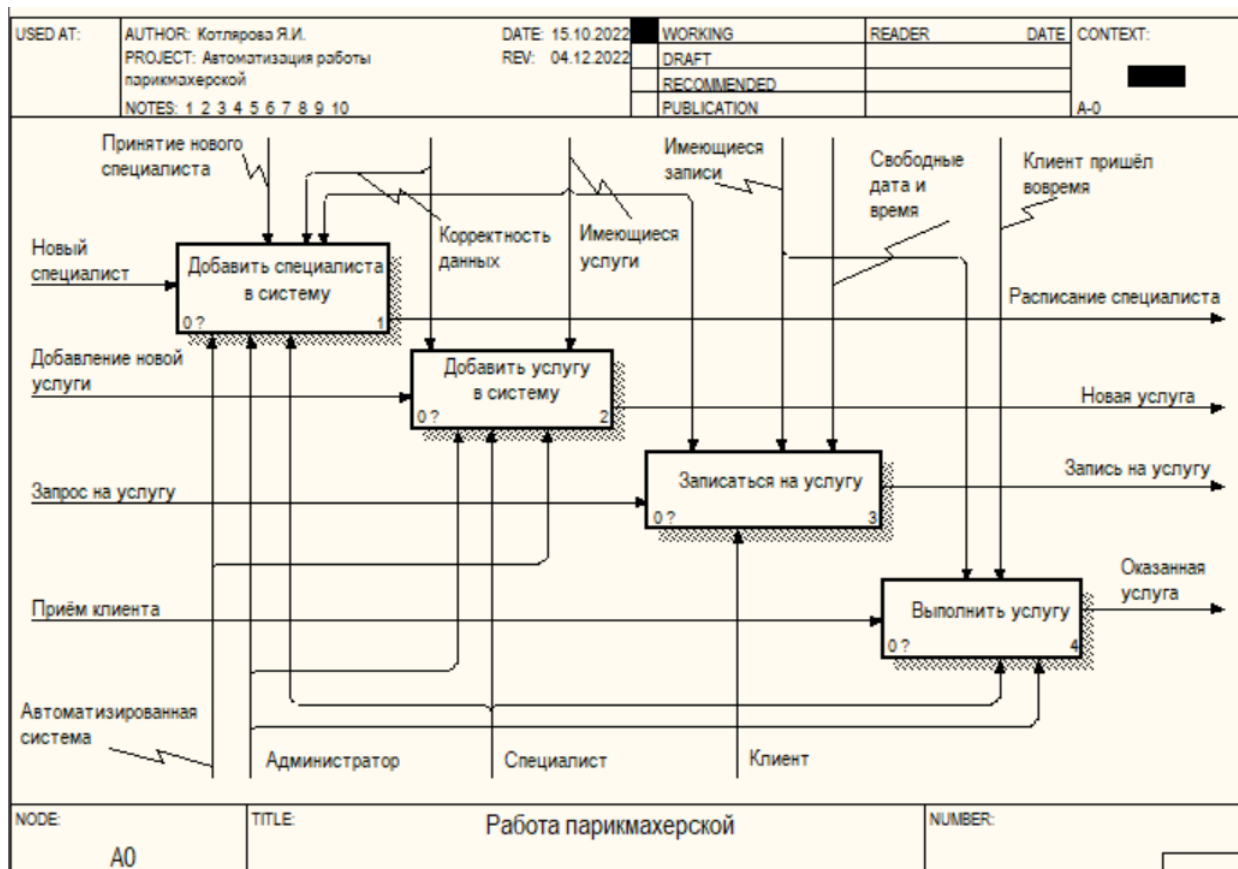


Рисунок 3.2 – Декомпозиция контекстной диаграммы

Первый компонент данной декомпозиции, «Добавить специалиста в систему», подразумевает добавление новых специалистов в базу данных. Данный процесс разделяется на подачу специалистом необходимых данных, после проверки корректности этих данных и принятия специалиста на работу данные вносятся в базу данных, а затем используются для составления расписания для нового сотрудника. Декомпозиция блока представлена на рисунке 3.3.

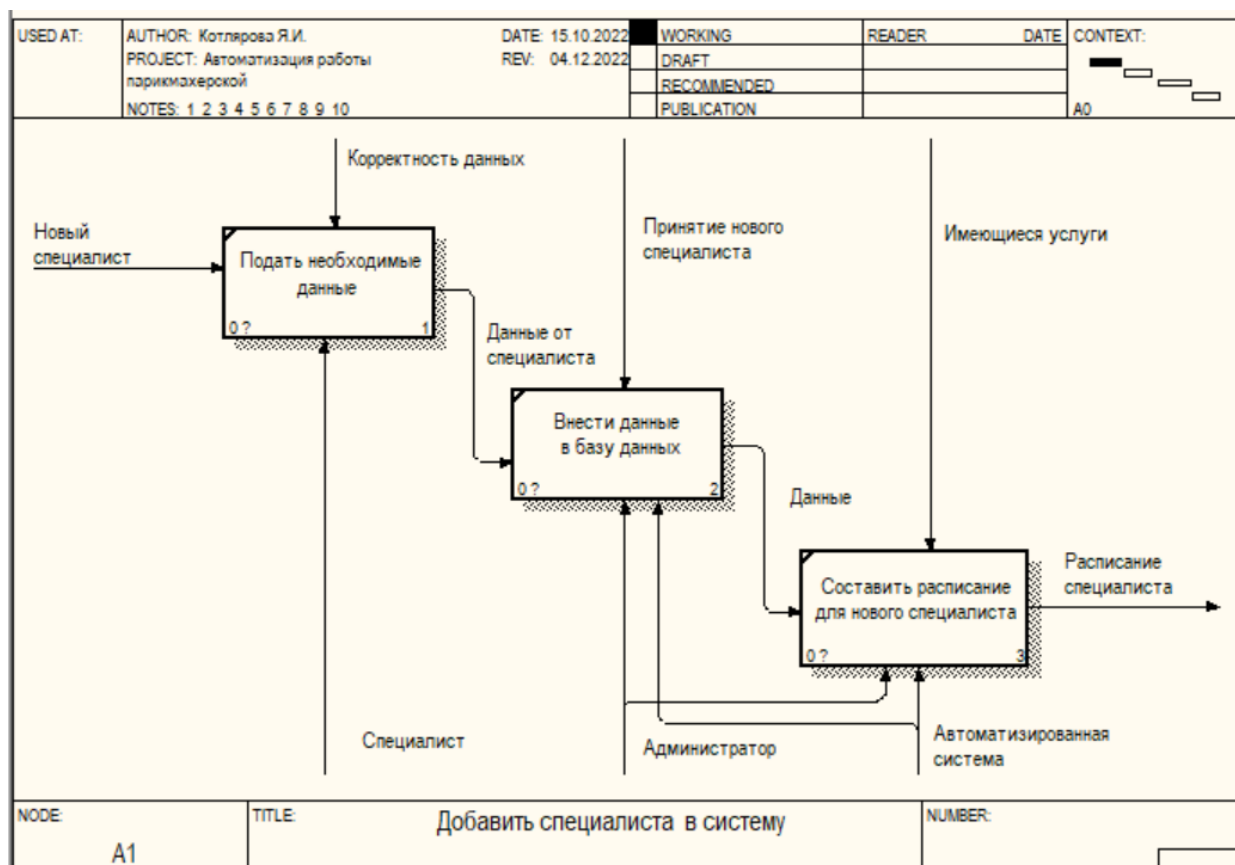


Рисунок 3.3 – Декомпозиция блока «Добавить специалиста в систему»

Второй компонент декомпозиции, «Добавить услугу в систему», подразумевает собой процесс, при котором от специалиста поступает запрос на создание новой услуги, при получении одобрения от администратора происходит внесение необходимых данных, которые после проверки на корректность заносятся в базу данных. Декомпозиция блока представлена на рисунке 3.4.

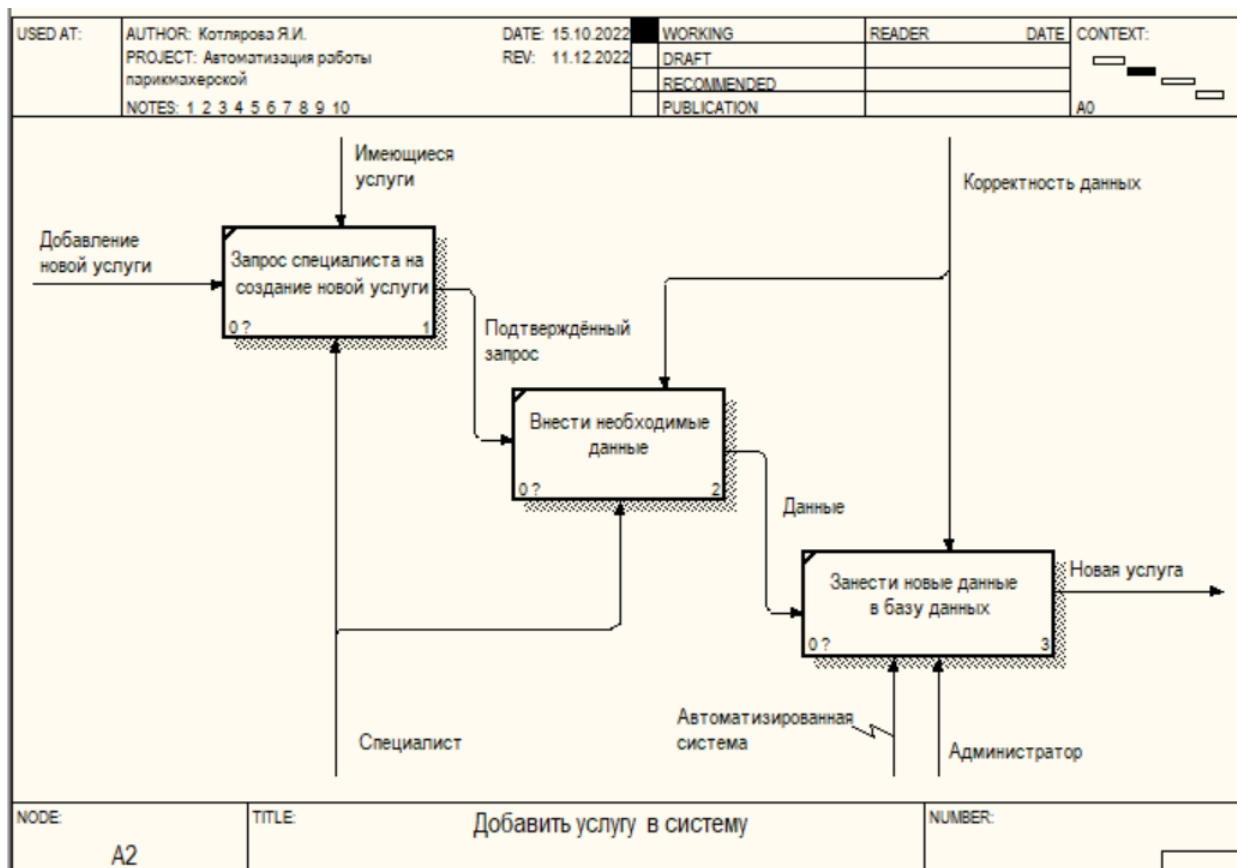


Рисунок 3.4 – Декомпозиция блока «Добавить услугу в систему»

Третий компонент представлен блоком «Записаться на услугу». В этом процессе происходит составление запроса от клиента. Запрос включает в себя выбор услуги, подходящего специалиста и даты и времени. Каждый выбор проверяется соответствующими механизмами ограничения. Декомпозиция блока «Записаться на услугу» представлена на рисунке 3.5.

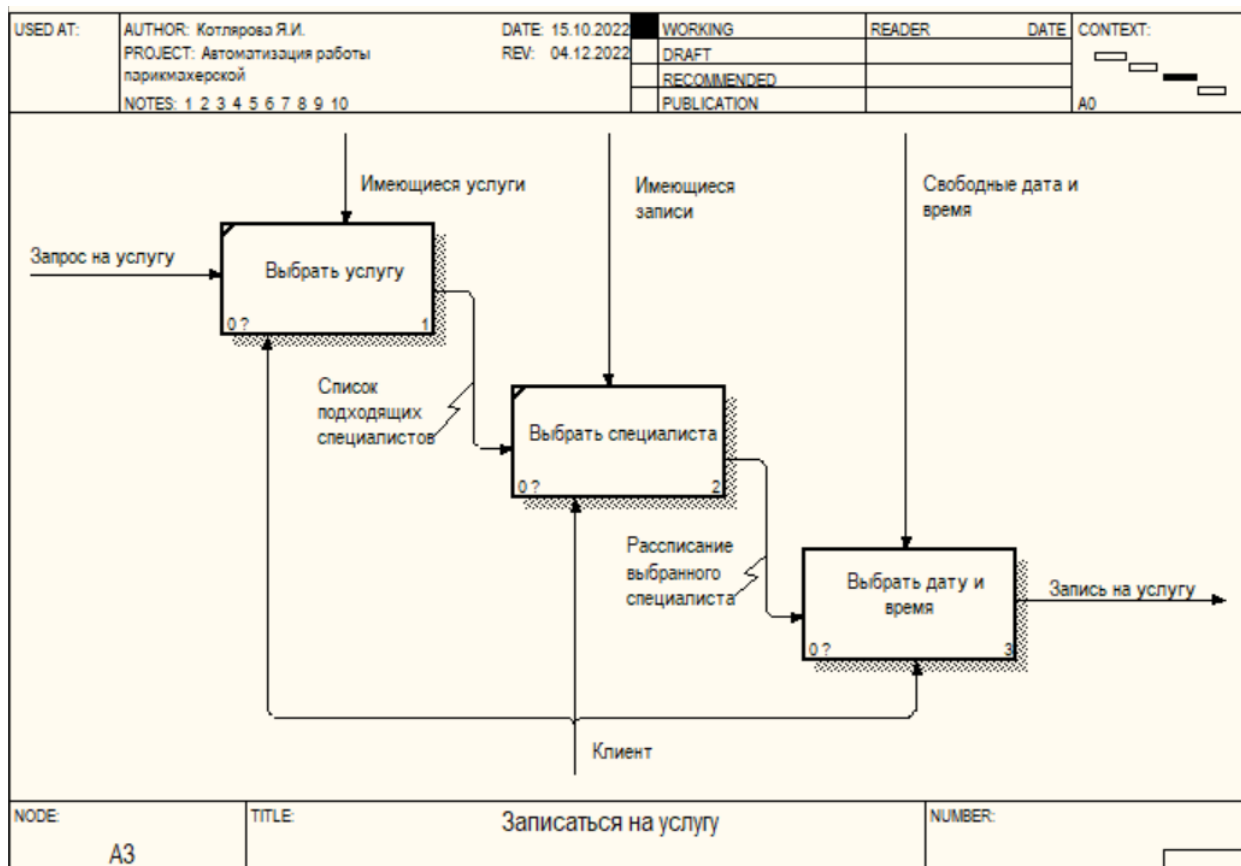


Рисунок 3.5 – Декомпозиция блока «Записаться на услугу»

Последний четвёртый компонент представлен блоком «Выполнить услугу». Данный процесс включает в себя подтверждение записи посредством связи администратора с клиентом, после получения подтверждения и прибытия клиента в нужное время у специалиста появляется возможность оказать услугу в соответствии с записью клиента. Декомпозиция блока «Выполнить услугу» представлена на рисунке 3.6.

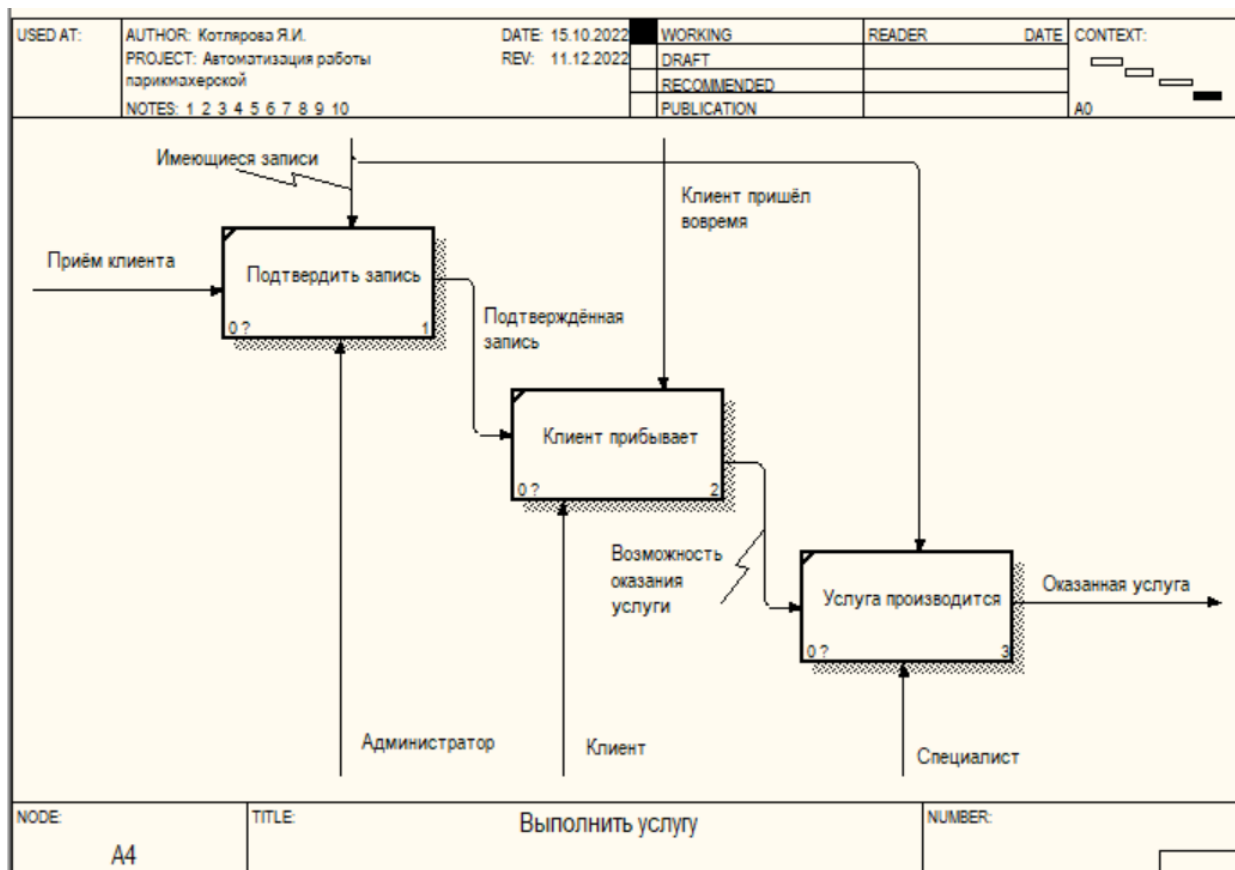


Рисунок 3.6 – Декомпозиция блока «Выполнить услугу»

Четвёртый уровень декомпозиции представлен блоком «Выбрать дату и время». Данный процесс включает в себя просмотр клиентом расписания специалиста и выбор подходящих даты и времени, предоставленными в выведенном расписании выбранного специалиста: при выборе доступной даты выводится доступное на эту дату время. Декомпозиция блока «Выбрать дату и время» представлена на рисунке 3.7.

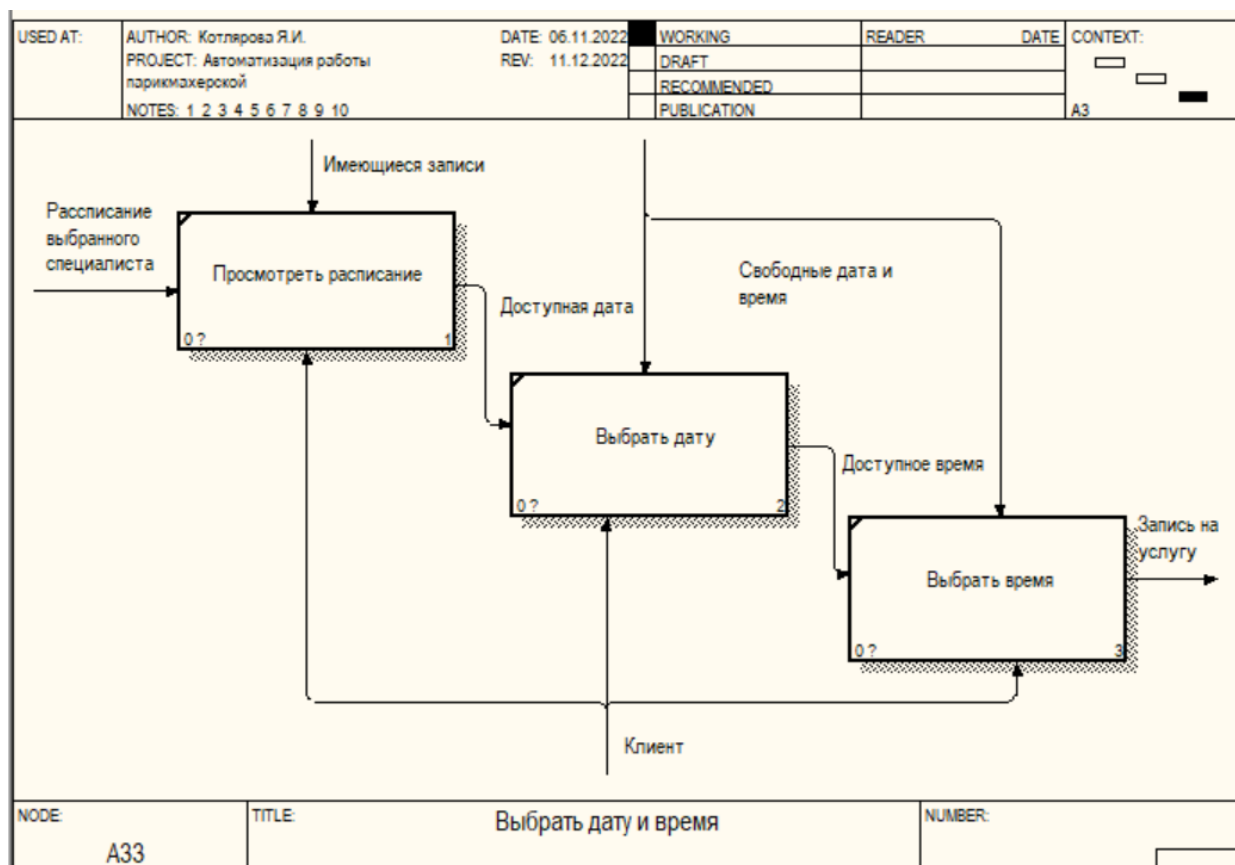


Рисунок 3.7 – Декомпозиция блока «Выбрать дату и время»

Данный подход моделирования позволяет наглядно видеть, как будет работать уже автоматизированная система для парикмахерской. Можно узнать на каком этапе происходит добавление нового специалиста и его услуг, а также запись клиента на необходимую ему услугу и реализацию этой записи.

4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЁ ОПИСАНИЕ

Моделирование данных – это средство, относящееся к бизнес-процессу организации, для формального сбора данных. Моделирование является одним из главных на сегодняшний день приемов анализа, основанием, на котором строятся реляционные базы данных.

Модель данных – графическое или лексическое представление данных, устанавливающее их свойства, структуры и взаимосвязи. Модель данных включает все объекты, их атрибуты и отношения, требуемые данным отдельным проектом.

Для моделирования базы данных применялась такая методология, как IDEF1.X, служащая для разработки реляционных баз данных. Для реализации методологии применялось программное средство Allfusion Erwin Data Modeler.

Методология IDEF1X - один из подходов к семантическому моделированию данных, основанный на концепции «Сущность – Отношение», это инструмент для анализа информационной структуры систем различной природы. Информационная модель, построенная с помощью IDEF1X-методологии, представляет логическую структуру информации об объектах системы. Эта информация является необходимым дополнением функциональной IDEF0-модели, детализирует объекты, которыми манипулируют функции системы.[2]

Концептуально IDEF1X-модель можно рассматривать как проект логической схемы базы данных для проектируемой системы.

При построении информационной модели используется алгоритм:

1. Определение сущностей и связей между ними.
2. Определение первичных и вторичных ключей для сущностей.
3. Определение атрибутов сущностей.
4. Переход к физическому описанию. [3]

Сущность – это объект предметной области, которая исследуется и моделируется. Другими словами сущность – это любой различимый объект, о котором необходимо хранить информацию в базе данных.

Атрибут отражает определенное свойство, качество, признак сущности. Например, для сущности студент атрибутами могут быть: № зачетной книжки, № группы, фамилия, имя, отчество, год рождения, адрес, телефон и т.д.

Связь – это ассоциация между сущностями, при которой, как правило, каждый экземпляр одной сущности, называемой родительской сущностью, ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, называемой сущностью-потомком, а каждый

экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя. Таким образом, экземпляр сущности-потомка может существовать только при существовании сущности родителя. [4]

После проведения информационного моделирования были определены следующие сущности:

- пользователь;
- клиент;
- сотрудник;
- услуга;
- запись.

Для более подробного описания построена информационная модель базы данных, которая представлена на рисунке 4.1.

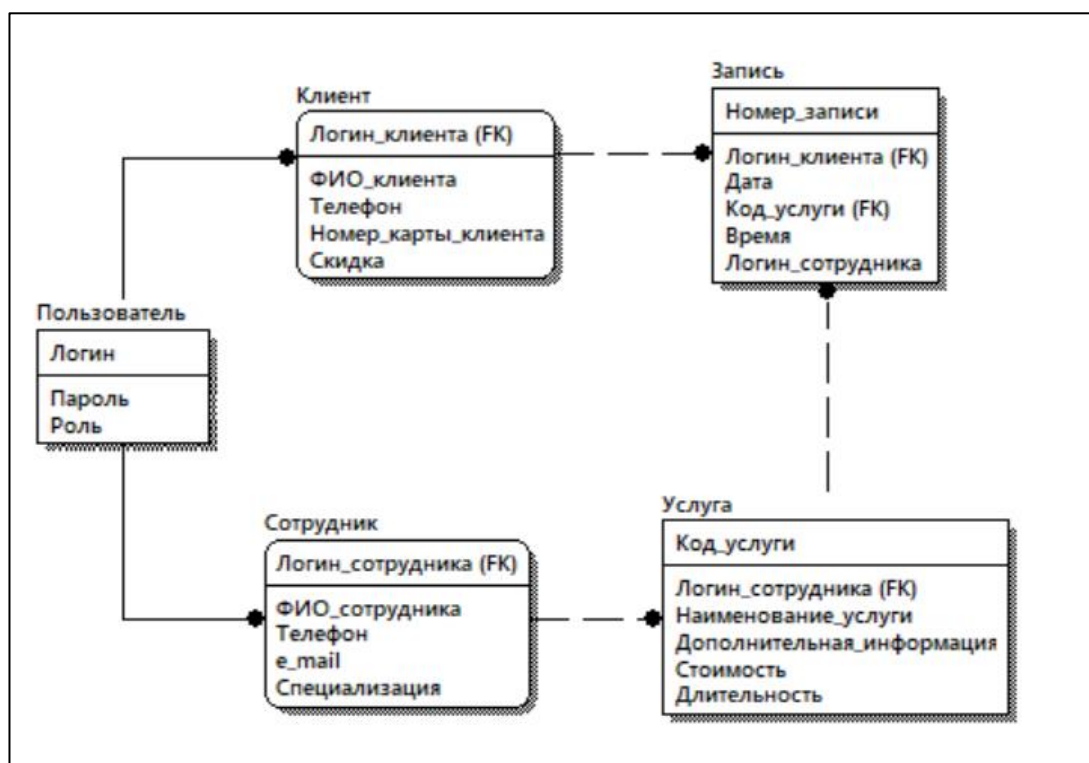


Рисунок 4.1 – Информационная модель базы-данных

Сущность «Пользователь» необходима для входа в программу и регистрации новых пользователей. Так как при обоих действиях фигурируют такие данные, как логин, пароль и роль пользователя, то всё это необходимо хранить. Всё перечисленное является атрибутами сущности «Пользователь».

Сущность «Клиент» хранит информацию, необходимую для отделения клиентов от работников. К ней относятся ФИО клиента, номер карты клиента,

который совпадает с номером его телефона также являющимся полем сущности, предоставляемая скидка и логин клиента, передаваемый из сущности «Пользователь». Данная сущность связана с сущностями «Пользователь», связь типа 1:1, и «Запись», связь типа 1:M.

Сущность «Сотрудник» хранит информацию о работниках. К ней относятся ФИО сотрудника, его специализация, телефон, электронная почта и логин сотрудника, передаваемый из сущности «Пользователь». Данная сущность связана с сущностями «Пользователь», связь типа 1:1, и «Услуга», связь типа 1:M.

Сущность «Услуга» содержит атрибуты код услуги, являющийся первичным ключом, наименование услуги, дополнительная информация, стоимость и длительность, а также логин сотрудника, оказывающего эту услугу, передаваемый из сущности «Сотрудник». Она также связана с сущностью «Запись», которая имеет атрибуты номер записи, являющийся первичным ключом, и дата и время получает от неё код услуги, на которую сделана запись, а также логин клиента, записавшегося на эту услугу, из сущности «Клиент».

Поскольку в представленной модели все атрибуты сущностей являются простыми (логически неделимыми на составные части), а также отсутствуют повторы атрибутов, то модель считается приведённой к первой нормальной форме. Вторая нормальная форма требует, чтобы сущность находилась в первой нормальной форме и имела исключительно простой первичный ключ, состоящий из одного атрибута. Так как данная модель отвечает вышеперечисленным требованиям, то она находится во второй нормальной форме. Третья нормальная форма запрещает зависимости между неключевыми атрибутами. Поскольку неключевые атрибуты сущностей данной модели зависят только от первичного ключа, а сущности уже находятся во второй нормальной форме, то сущности считаются приведёнными к третьей нормальной форме.

Последовательное приведение сущностей к третьей нормальной форме гарантирует отсутствие избыточности данных, которая может возникнуть за счёт содержания в сущностях разнородной информации. Нормализация данных позволяет избежать аномалий вставки, обновления и удаления при работе с базой данных.

5 ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ СИСТЕМЫ АВТОМАТИЗАЦИИ РАБОТЫ ПАРИКМАХЕРСКОЙ

5.1 Алгоритмы, реализующие бизнес-логику серверной части

Описание работы алгоритма добавления услуги в систему.

В ходе выполнения поставленной задачи, был разработан алгоритм, позволяющий специалисту добавить оказываемую им услугу. Блок-схема данного алгоритма представлена на рисунке 5.1.

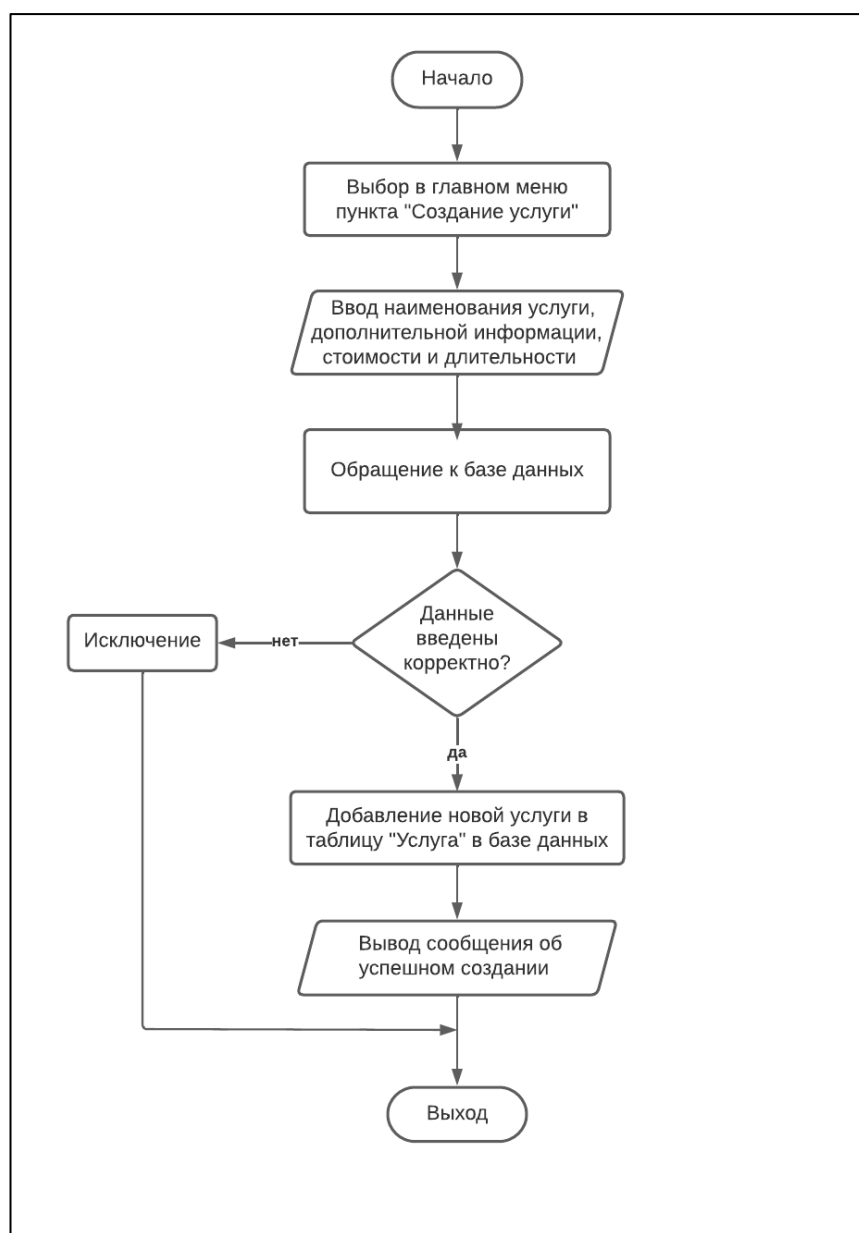


Рисунок 5.1 – Блок-схема алгоритма добавления услуги в систему

Для того, чтобы выполнить добавление зарегистрированному специалисту необходимо выбрать подходящий пункт меню правильно подать требуемые данные:

- наименование услуги;
- дополнительная информация;
- стоимость;
- длительность.

После правильного введения данных сервер обращается к базе данных для добавления услуги, и при отсутствии ошибок при вводе услуга будет добавлена в таблицу «Услуга», где автоматически заполнится ключевое поле код услуги и добавится логин создавшего услугу сотрудника.

Также у сотрудника есть возможность просмотреть созданные им услуги, редактировать некорректно введенные или подверженные изменению данные и, при необходимости, удалить услугу из базы данных посредством выбора соответствующих пунктов меню на клиентской части.

Описание работы алгоритма записи на услугу.

Также был разработан алгоритм, позволяющий клиенту записаться на необходимую ему услугу к любому из оказывающих её специалистов при наличии свободных даты и времени. Блок-схема данного алгоритма представлена на рисунке 5.2.

Запись на услугу осуществляется следующим образом: клиент выбирает соответствующий пункт меню, сервер обращается к базе данных и выводит клиенту список оказываемых услуг. После того, как клиент определится с необходимой ему услугой, сервер снова обратится к базе данных, для получения логина сотрудника, предоставляющего данную услугу. После получения логина сервер сможет снова обратиться к базе данных для получения расписания выбранного сотрудника, которое отобразится клиенту, чтобы он смог выбрать устраивающие его дату и время. По окончании выбора сервер снова обратится к базе данных для внесения записи в таблицу, где автоматически заполнится ключевое поле номер записи, запишутся код выбранной услуги, логин клиента, совершившего запись и выбранные им дата и время.

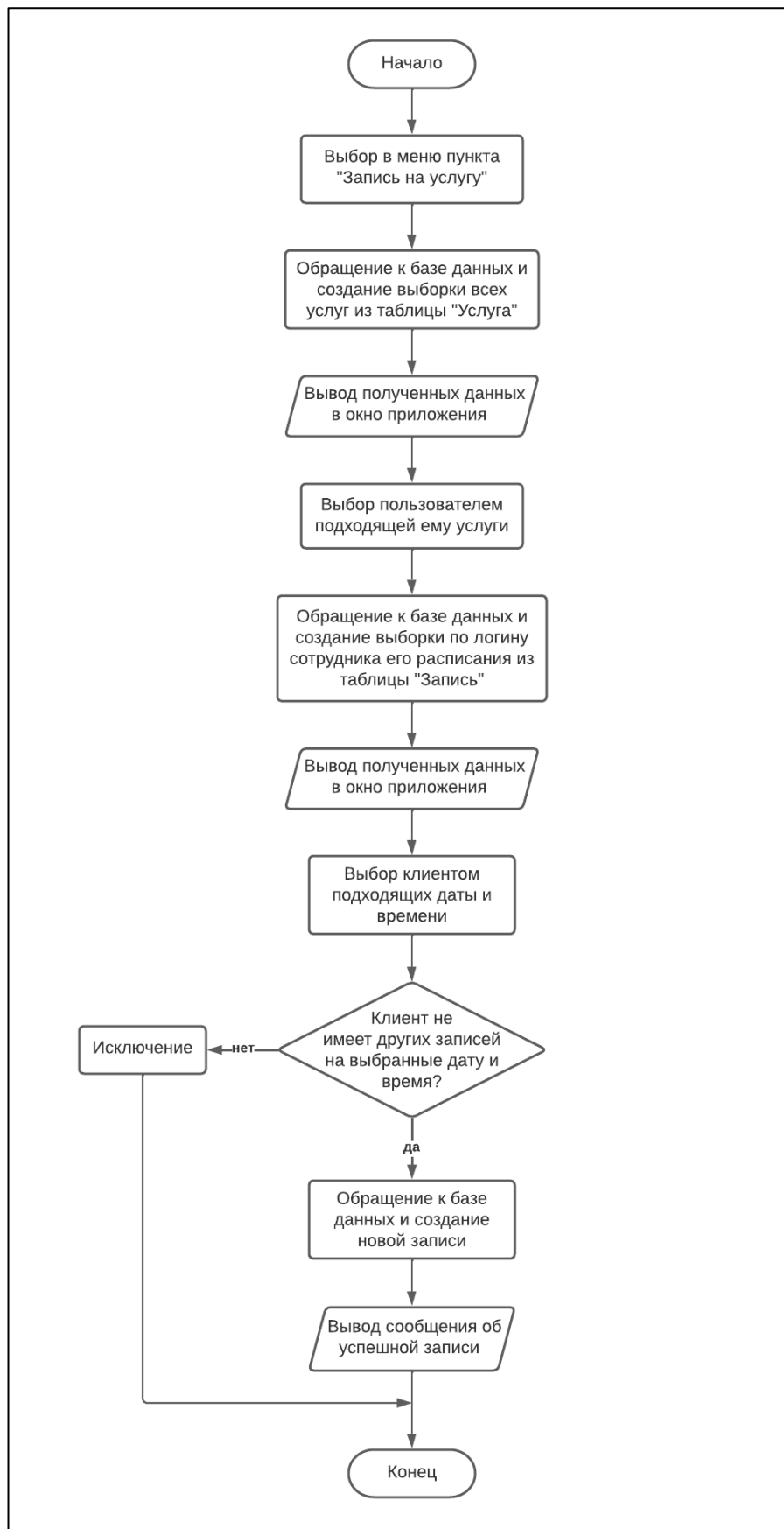


Рисунок 5.2 – Блок-схема алгоритма записи на услугу

Описание работы алгоритма составления расписания.

Также был разработан алгоритм, составляющий расписание специалиста, которое полезно как для самого специалиста, так и для клиента, при записи на услугу. Блок-схема данного алгоритма представлена на рисунке 5.3.

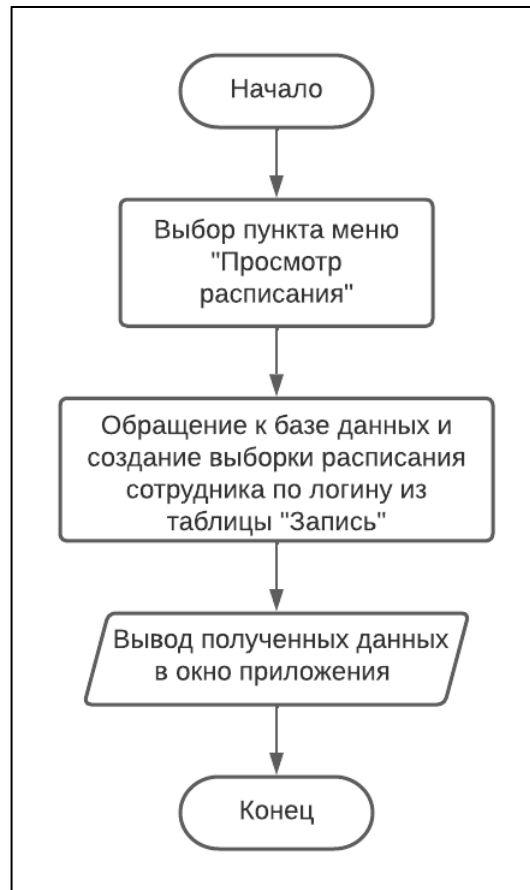


Рисунок 5.3 – Блок-схема алгоритма составление расписания

Составление расписания происходит следующим образом: при поступлении на сервер запроса на получение расписания определённого специалиста от него самого, администратора или же от клиента при выборе ими соответствующих пунктов меню, сервер отправляет запрос базе данных на составление выборки по логину специалиста из таблицы «Запись», после чего полученные данные формируют календарную таблицу на неделю с датами, временем и уже существующими записями.

При внесении каких-либо изменений со стороны самого сотрудника или клиента расписание будет автоматически редактироваться.

5.2 Диаграмма вариантов использования

Диаграмма вариантов использования – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей.

Созданная диаграмма представлена в приложении А.

В диаграмме актёрами являются Пользователь, Администратор, Сотрудник и Клиент. После авторизации или регистрации актёр Пользователь получает роль в приложении. Администратор имеет возможность работать со всеми таблицами базы данных, а именно осуществлять создание, чтение, изменение и удаление полей, записывать данные в файл для получения аналитических данных, а также чтение этих данных. Сотрудник имеет возможность взаимодействовать с таблицей услуг, для которой ему доступны добавление, чтение, редактирование и удаление, и таблицей записей, из которых составляется его расписание. Клиент может взаимодействовать с таблицей записей, добавляя новые, редактируя, просматривая и удаляя имеющиеся.

5.3 Диаграмма состояний

Диаграмма состояний описывает все возможные состояния одного объекта и возможные последовательности его переходов из одного состояния в другое, то есть моделирует все изменения состояний объекта как его реакцию на внешние воздействия.

Созданная диаграмма представлена в приложении Б.

В данной диаграмме представлено состояния объекта «Услуга». Начальным его состоянием является «идея новой услуги», которая переходит в «новую предложенную услугу». После получения отклонения предложения объект перестает существовать, а после получения одобрения объект переходит в состояние «новая одобренная услуга». Далее поступает запрос на введение данных, и после их корректного ввода объект добавляется в базу данных и переходит в конечное состояние «новая оказываемая услуга».

5.4 Диаграмма последовательности

Диаграмма последовательности описывает поведение только одного варианта использования. На такой диаграмме отображаются только экземпляры объектов и сообщения, которыми они обмениваются между собой.

Созданная диаграмма представлена в приложении В.

В данной диаграмме пользователь добавляет услугу в систему. При выборе этой функции ему необходимо ввести необходимые данные о добавляемой услуге, которые формируют объект, который потом отправляется серверу, в котором вызывается функция запроса к базе данных.

5.5 Диаграммы классов

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, из коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними.

Пакет authorization содержит в себе класс LogIn, функция которого обеспечивает проверку на совпадения логина и пароля с одной из строк в таблице user. Диаграмма класса представлена на рисунке 5.4.

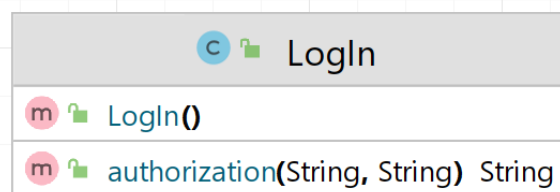


Рисунок 5.4 – Диаграмма класса LogIn

Пакет Client содержит в себе классы BasicWindowAction, Client и ClientApplication, реализующие работу клиентской части в TCP/IP соединении. Диаграммы классов BasicWindowAction и ClientApplication представлены на рисунке 5.5, а диаграмма класса Client представлена в приложении Г.

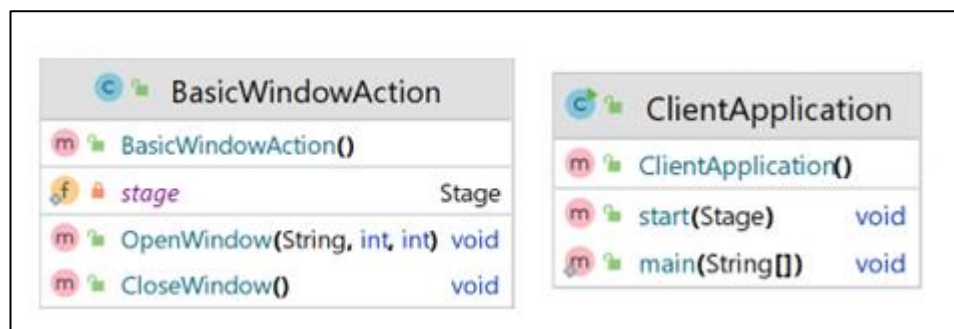


Рисунок 5.5 – Диаграммы классов BasicWindowAction и ClientApplication

Пакет Server содержит в себе классы ClientThread и Server, реализующие работу серверной части в TCP/IP соединении. Диаграммы классов представлены на рисунках 5.6.

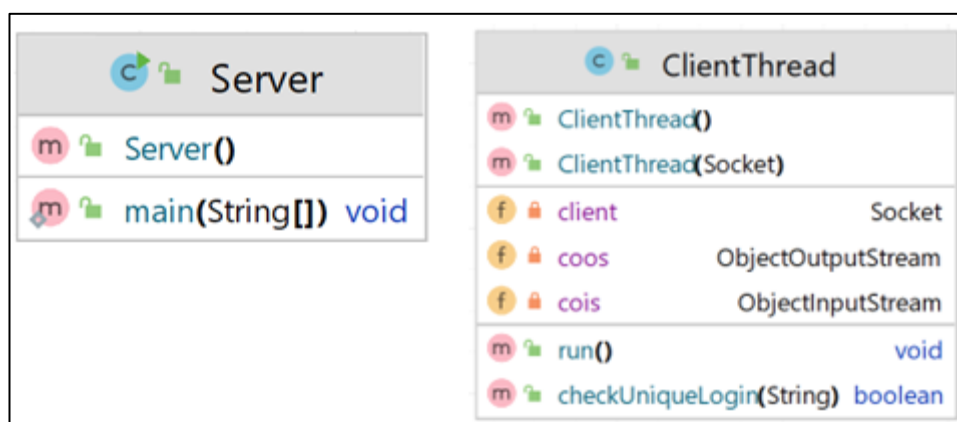


Рисунок 5.6 – Диаграммы классов Server и ClientThread

Пакет model содержит в себе классы Users, Clients, Workers, Service и Register, являющиеся моделями для таблиц в базе данных для сервера. Диаграммы классов представлены на рисунках 5.7 и 5.8.

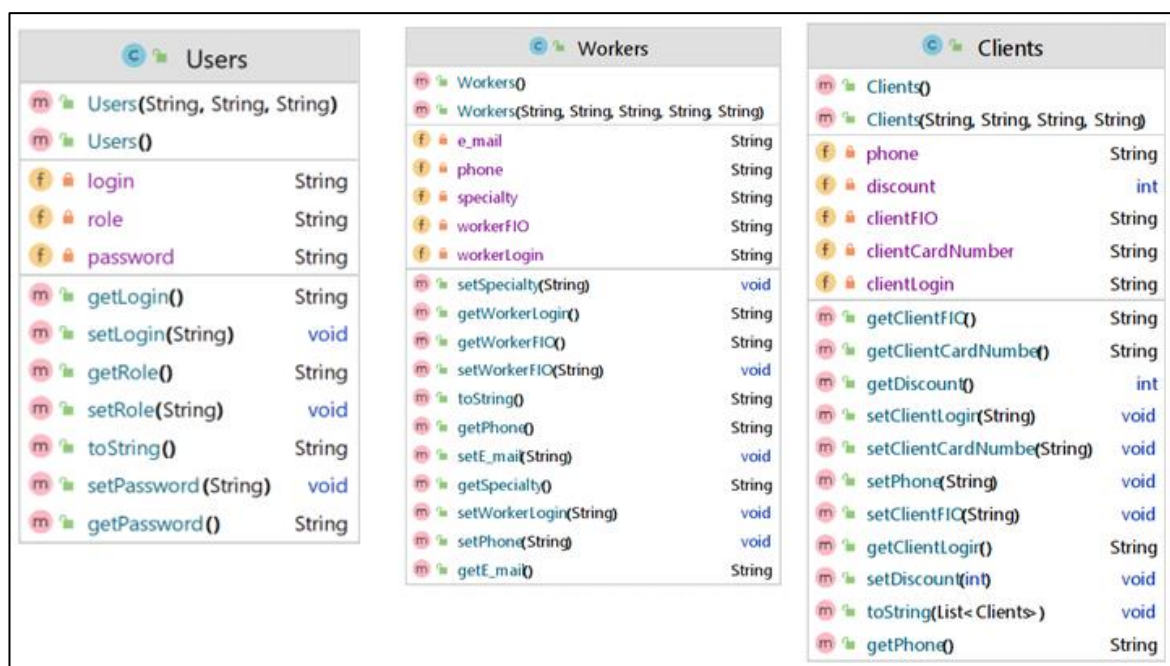


Рисунок 5.7 – Диаграммы классов Users, Workers и Clients



Рисунок 5.8 – Диаграммы классов Service и Register

Пакет modelForClient содержит в себе классы UserModel, WorkerModel, ClientModel, ServiceModel и RegisterModel, являющиеся моделями для таблиц в базе данных для клиента. Диаграммы классов представлены на рисунках 5.9 и 5.10.

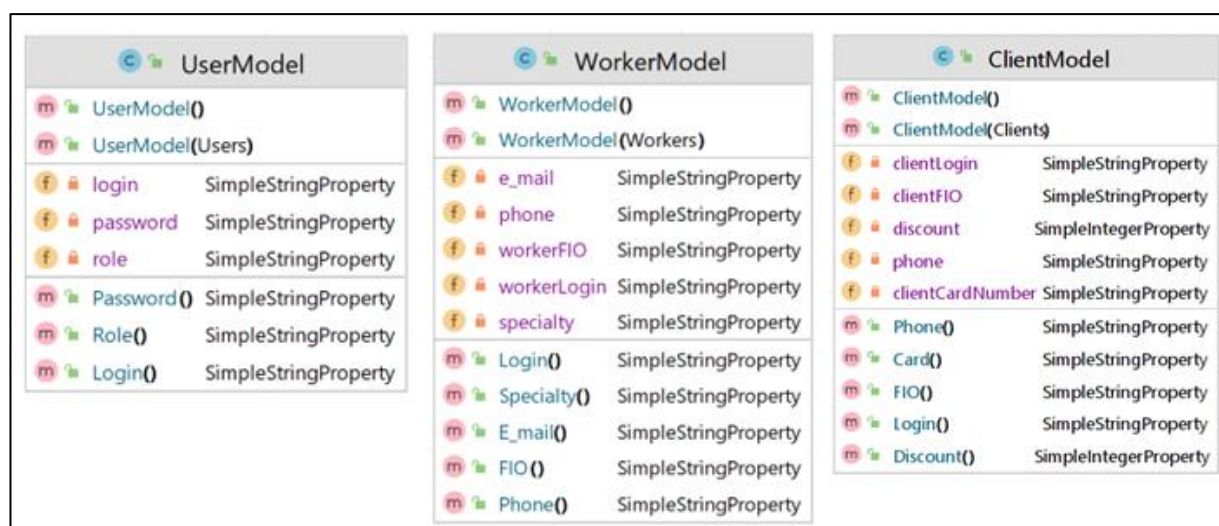


Рисунок 5.9 – Диаграммы классов UserModel, WorkerModel и ClientModel

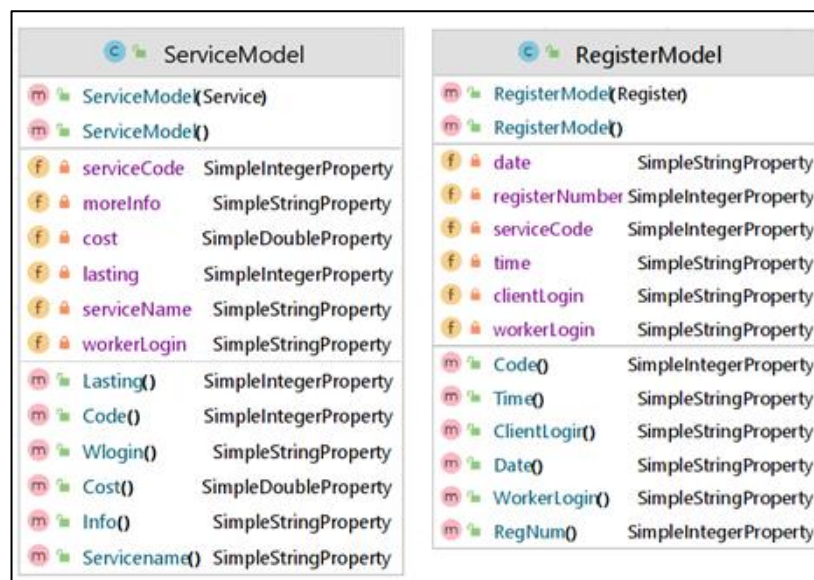


Рисунок 5.10 – Диаграммы классов ServiceModel и RegisterModel

Пакет dao содержит в себе класс ConnectionFactory для связи программы с базой данных, а также классы UserDao, ClientDao, WorkerDao, ServiceDao и RegisterDao, содержащие запросы к таблицам базы данных. Диаграмма класса ConnectionFactory представлена на рисунке 5.11. Диаграммы классов UserDao, ClientDao и WorkerDao представлены на рисунке 5.12, а диаграммы классов ServiceDao и RegisterDao представлены на рисунке 5.13.

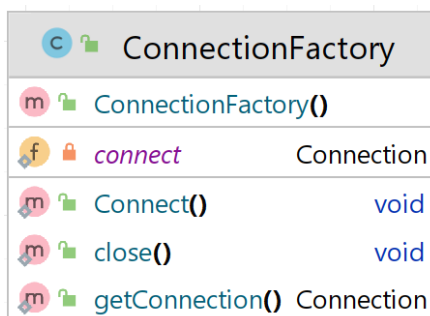


Рисунок 5.11 – Диаграмма класса ConnectionFactory



Рисунок 5.12 – Диаграммы классов UserDao, ClientDao и WorkerDao

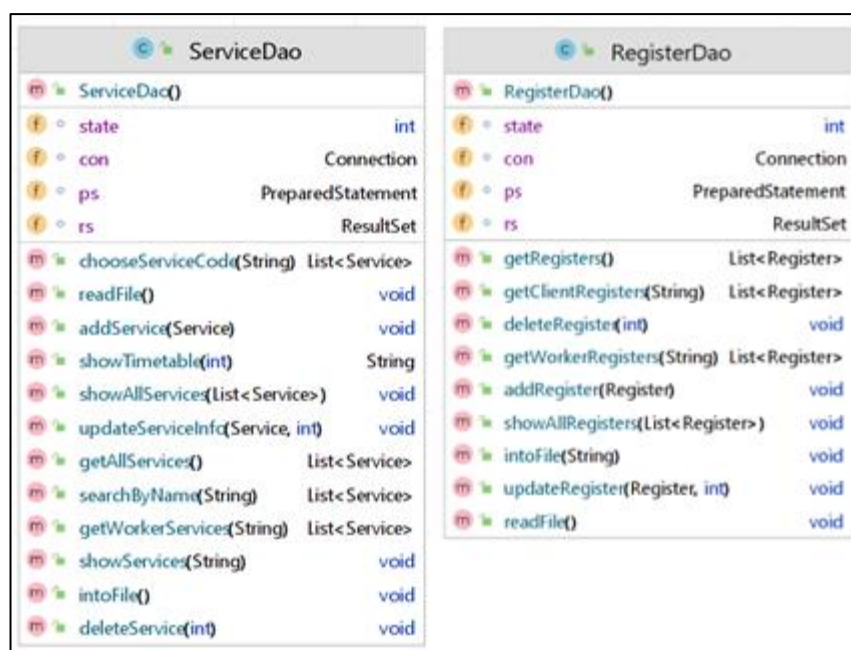


Рисунок 5.12 – Диаграммы классов ServiceDao и RegisterDao

5.6 Диаграммы компонентов и развёртывания

Диаграмма компонентов — элемент языка моделирования UML, статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и тому подобное.

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их.

Диаграммы развертывания обычно используются для визуализации физического аппаратного и программного обеспечения системы. Используя его, вы можете понять, как система будет физически развернута на аппаратном обеспечении.

Данные диаграммы представлены на рисунке 5.4.

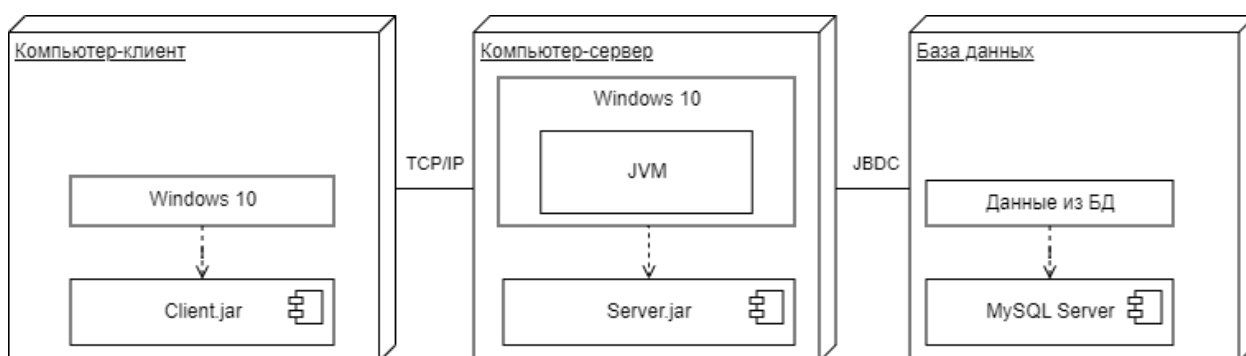


Рисунок 5.4 – Диаграммы компонентов и развёртывания

На данной диаграмме в качестве узлов выступают компьютер-клиент, компьютер-сервер и база данных.

Для запуска разрабатываемого приложения необходимо наличие исполняемой среды JDK версии JDK8 на компьютерах клиента, администратора и сотрудника.

Связь узлов компьютеров администратора, сотрудника и клиента с сервером осуществляется по протоколу TCP/IP, а связь сервера с базой данных осуществляется с помощью JDBC.

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

6.1 Руководство для клиента

В программе присутствуют две части: серверная и клиентская, которые реализованы отдельно. Для начала работы следует запустить серверную часть, а затем клиентскую. Программа будет работать только при наличии подключения к базе данных.

После запуска, программа предоставляет пользователю окно авторизации, предоставленное на рисунке 6.1. При наличии учётной записи можно войти в программу, а при её отсутствии – зарегистрироваться.

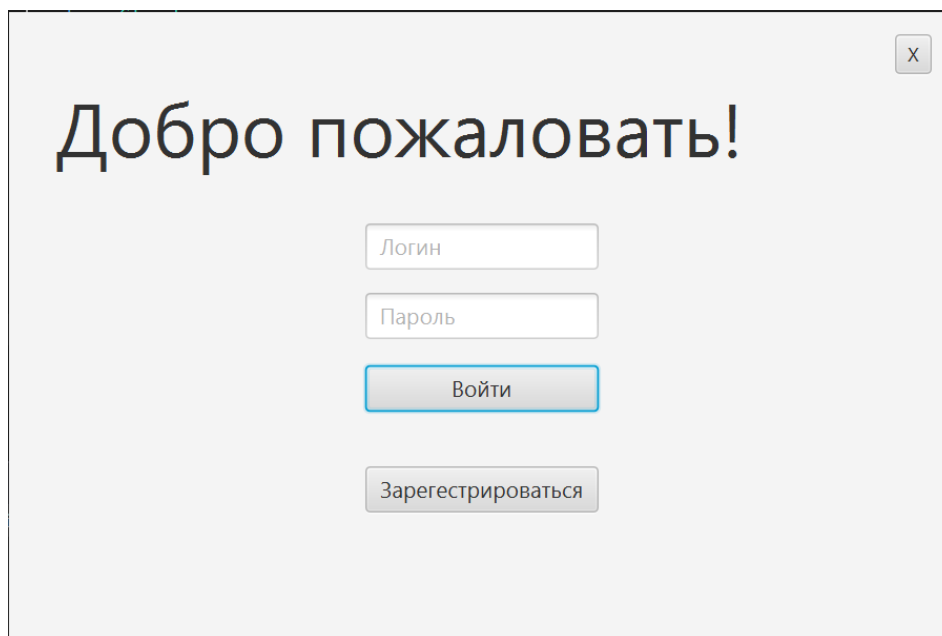
The image shows a software window with a light gray background. At the top right is a small square button with an 'X' icon. The main text 'Добро пожаловать!' is centered in a large, dark font. Below it are four elements stacked vertically: a text input field with the placeholder 'Логин', another text input field with the placeholder 'Пароль', a button with the text 'Войти' (highlighted with a blue border), and a button with the text 'Зарегистрироваться'.

Рисунок 6.1 – Окно авторизации

На этапе регистрации предоставляется возможность создать запись только обычного пользователя, то есть клиента. Добавление новых сотрудников осуществляется администратором. Для этого пользователю предлагается ввести логин, пароль, которые будут использоваться для дальнейшей авторизации в программе. Также обязательным условием является подтверждение пароля, в целях избегания ввода случайного пароля. Дальше требуется ввести ФИО и номер телефона. Окно регистрации представлено на рисунке 6.2.

The registration window has a light gray background and a dark gray border. At the top center is the title 'Регистрация' in a large, bold, black font. Below the title are four input fields arranged in a 2x2 grid. The top-left field is empty with a blue border. The top-right field is labeled 'ФИО' in a small gray font. The bottom-left field is labeled 'Пароль' in a small gray font. The bottom-right field is labeled 'Телефон' in a small gray font. Below these fields is a single button labeled 'Зарегистрироваться' in a small gray font.

Рисунок 6.2 – Окно регистрации

Пройдя авторизацию, можно попасть либо на страницу выбора действий пользователя, сотрудника или администратора.

На рисунке 6.3 представлено окно с действиями, доступными обычному пользователю. Этот функционал включает в себя:

- просмотр личной информации;
- изменение личной информации;
- действия с записями.

The user actions window has a light gray background and a dark gray border. At the top center is the title 'Выбирайте' in a large, bold, black font. Below the title are three buttons. On the left is a button labeled 'Действия с записями' with a small downward arrow on its right side. To its right is a button labeled 'Просмотр личной информации' with a blue border. Below the 'Просмотр личной информации' button is a button labeled 'Редактирование личной информации'. In the bottom right corner is a button labeled 'Выход'.

Рисунок 6.3 – Окно действий клиента

При выборе просмотра личной информации пользователю выведется окно с его личными данными: логином, ФИО, номером телефона, номером

карты клиента и предоставляемой по карте скидкой. Окно вывода личной информации представлено на рисунке 6.4. Для этой таблицы также доступно изменение данных. Окно реализации функции изменения личных данных представлено на рисунке 6.5.

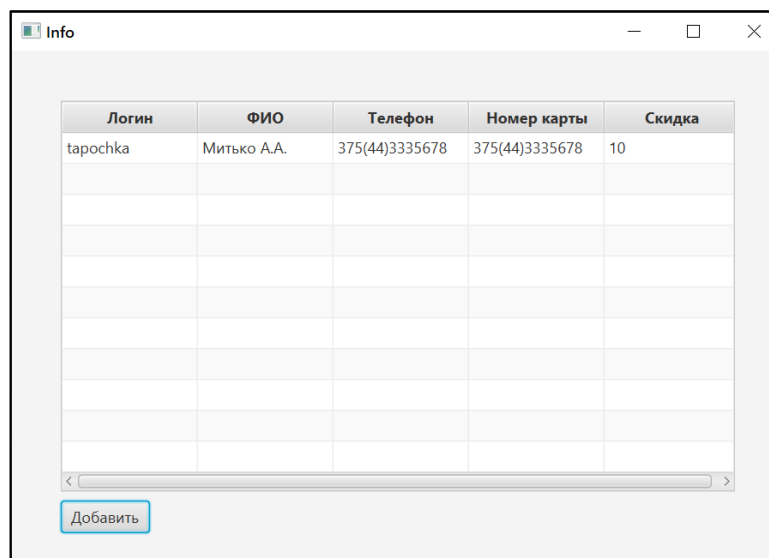


Рисунок 6.4 – Окно просмотра личной информации

The screenshot shows a window titled 'Изменение личной информации' (Change personal information). It contains a form with the label 'Изменить:' (Change) and three input fields: 'ФИО' (Full Name), 'Телефон' (Phone), and 'Пароль' (Password). A 'Подтвердить' (Confirm) button is located at the bottom right.

Рисунок 6.5 – Окно изменения личных данных клиента

Ещё одним функционалом, доступным пользователю, является возможность записи на услугу. При выборе этого пункта меню открывается список предоставляемых услуг, а также поля для ввода информации, необходимой для записи. Его вид представлен на рисунке 6.6.

The 'Add Register' window contains a table with the following data:

Код услуги	Логин сотру...	Наименован...	Доп. инфор...	Стоимость	Длительность
5	mikamin	Стрижка		70.0	60
6	worker	Укладка	Женская укл...	40.0	120
8	mikamin	Стрижка бор...		30.0	60

Below the table are the following controls:

- A blue 'Добавить' button.
- Input fields for 'Код услуги', 'Дата', and 'Время'.
- A grey 'Записаться' button.

Рисунок 6.6 – Окно записи на услугу

Также пользователю предоставляется возможность просмотра имеющихся записей и, при необходимости, отмена записи. Окно отмены имеющихся записей представлено на рисунке 6.7.

The 'Delete Register' window contains a table with the following data:

Номер запи...	Логин клие...	Код услуги	Логин рабо...	Дата	Время
6	tapochka	6	worker	16.12.22	11:30
8	tapochka	8	mikamin	14.12.22	15:00

Below the table are the following controls:

- A blue 'Добавить' button.
- An input field labeled 'Номер записи для удаления'.
- A grey 'Удалить' button.

Рисунок 6.7 – Окно просмотра имеющихся записей

6.2 Руководство для сотрудника

6.2.1 Руководство для администратора. Под администратором при разработке данной программы подразумевались директор и администратор парикмахерской. В связи с этим администратору доступен более широкий функционал:

- просмотр личной информации;
- изменение личной информации;
- действия с пользователями;
- действия с сотрудниками;
- действия с клиентами;
- действия с услугами;
- просмотр списка записей.

Окно выбора действия для администратора представлено на рисунке 6.8.

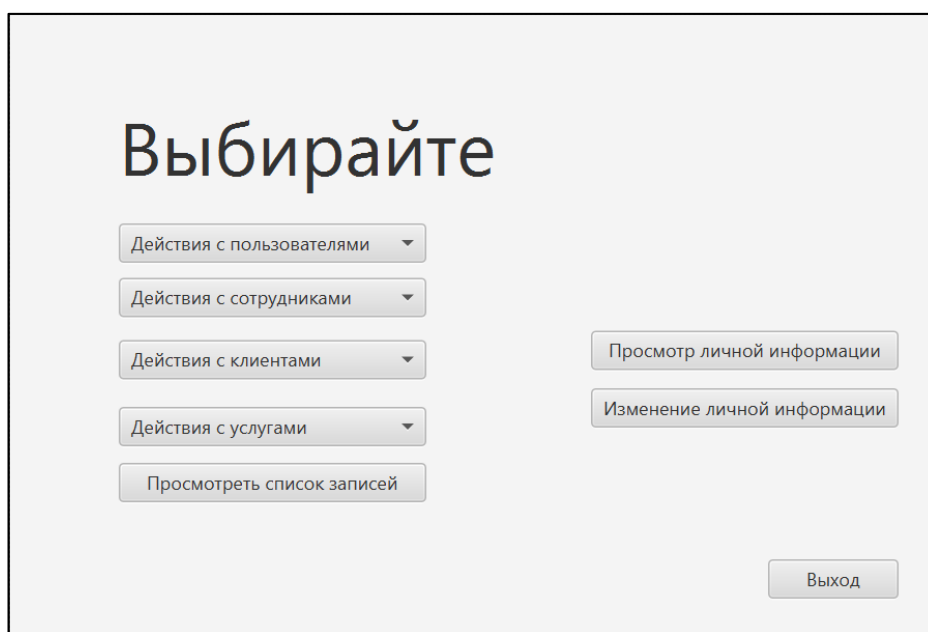


Рисунок 6.8 – Окно выбора действий администратора

При выборе просмотра личной информации администратору выведется окно с его личными данными: логином, ФИО, номером телефона и электронной почтой. Окно редактирования личной информации представлено на рисунке 6.9.

Изменение личной информации

Изменить:

Рисунок 6.9 – Окно редактирования личной информации

Таблица пользователей содержит информацию обо всех зарегистрированных пользователях: клиентах, сотрудниках и администраторов. В ней содержатся их логины, пароли и роли. Для данной таблицы доступны CRUD операции, запись данных в файл и аналитика ролей. Окно, отображающее реализацию этих функций, а также окно аналитики ролей представлены на рисунках 6.10 и 6.11 соответственно.

Выбирайте

Действия с пользователями

- Просмотреть список пользователей
- Составить файл
- Аналитика ролей

Действия с услугами

Просмотреть список записей

Просмотр личной информации

Изменение личной информации

Выход

Рисунок 6.10 – Окно работы с таблицей пользователей

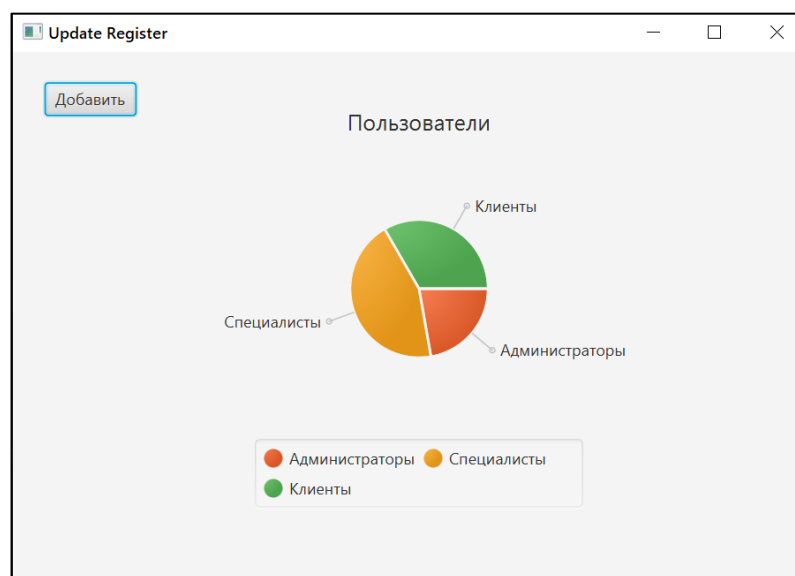


Рисунок 6.11 – Окно аналитики ролей пользователей

Таблица сотрудники содержит всю необходимую информацию о сотрудниках из базы данных. Для таблицы доступны CRUD операции и запись данных в файл. Также здесь администратор может перевести сотрудника в группу администраторов, то есть открыть доступ к полному функционалу. Окно, отображающее реализацию этих функций представлено на рисунке 6.12.

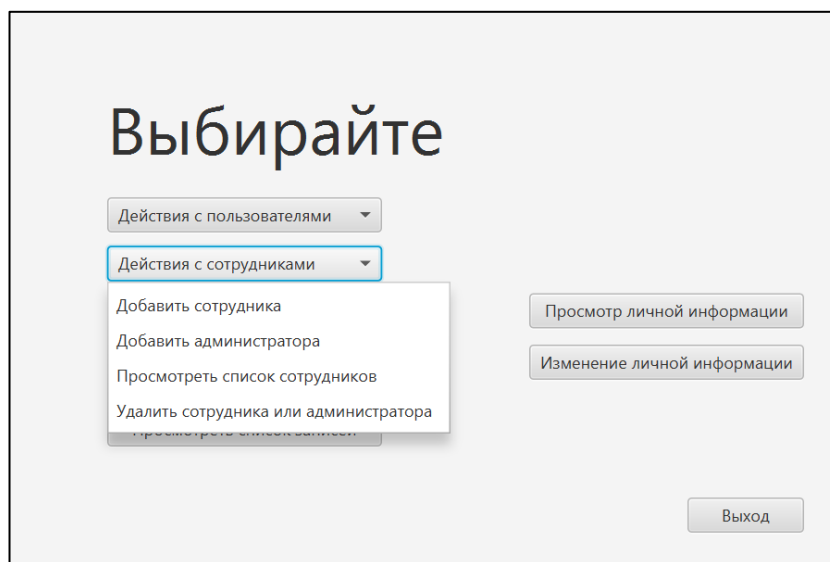


Рисунок 6.12 – Окно работы с таблицей сотрудников

Работа с данными представляет собой действия с 5 таблицами: клиенты, сотрудники, администраторы, предоставляемые услуги и расписание

специалистов. Таблица клиентов содержит такие же поля, как и аналогичная таблица в базе данных. Для таблицы клиентов доступны операции CRUD, запись данных в файл, а также добавление скидок и вывод аналитики по скидкам. Окно, отображающее выбор функций, а также окно аналитики по скидкам представлены на рисунках 6.13 и 6.14 соответственно.

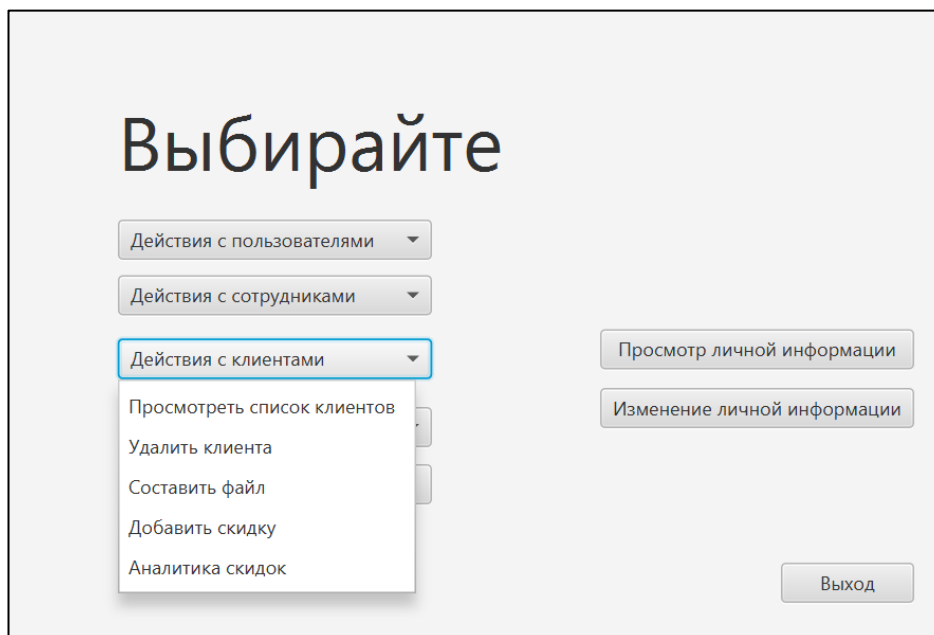


Рисунок 6.13 – Окно отображения функций

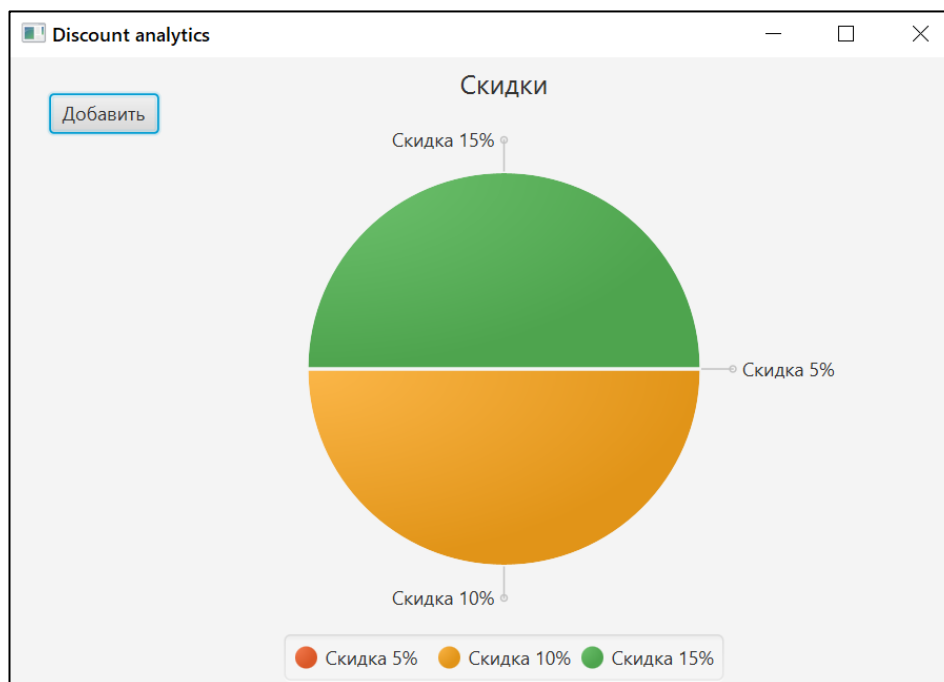


Рисунок 6.14 – Окно аналитики по скидкам

Далее идёт вкладка, отображающая таблицу с предоставляемыми в парикмахерской услугами, по полям совпадающая с таблицей «Услуга» в базе данных. Для таблицы доступны CRUD операции, поиск по наименованию услуги и аналитика по ценам и ценовым сегментам. Окно, отображающее реализацию этих функций представлено на рисунке 6.15. Окна аналитики цен и ценовых сегментов представлены на рисунках 6.16 и 6.17 соответственно.

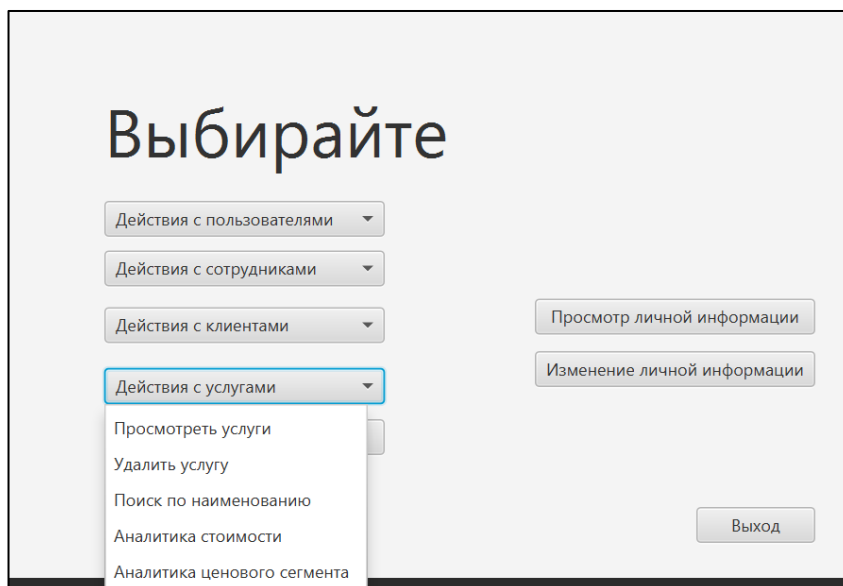


Рисунок 6.15 – Окно работы с таблицей услуг



Рисунок 6.16 – Окно аналитики цен

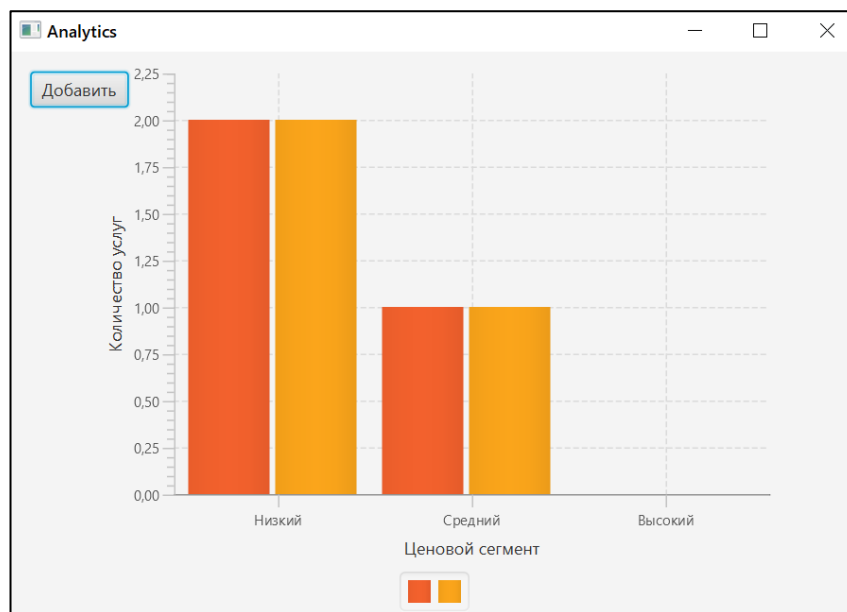


Рисунок 6.17 – Окно аналитики ценовых сегментов

Последней вкладкой данного окна является вкладка, содержащая таблицу с расписаниями всех сотрудников. Окно, отображающее реализацию этих функций представлено на рисунке 6.18.

Номер запи...	Логин клие...	Код услуги	Логин рабо...	Дата	Время
6	tapochka	6	worker	16.12.22	11:30
7	darmel	5	mikamin	15.12.22	15:30
8	tapochka	8	mikamin	14.12.22	15:00

Рисунок 6.18 – Окно работы с таблицей расписаний

6.2.2 Руководство для специалиста. На рисунке 6.19 представлено окно с действиями, доступными специалисту. Этот функционал включает в себя:

- просмотр личной информации;

- изменение личной информации;
- действия с услугами;
- действия с расписанием.

The screenshot shows a window titled "Выбирайте" (Choose). It contains four buttons arranged in a 2x2 grid. The top-left button is labeled "Действия с услугами" (Actions with services) and has a dropdown arrow. The top-right button is labeled "Просмотр личной информации" (View personal information). The bottom-left button is labeled "Действия с расписанием" (Actions with schedule) and has a dropdown arrow. The bottom-right button is labeled "Изменение личной информации" (Change personal information). A "Выход" (Exit) button is located at the bottom right of the window.

Рисунок 6.19 – Окно действий сотрудника

При выборе просмотра личной информации специалиста выведется окно с его личными данными: логином, ФИО, номером телефона, электронной почтой и специализацией. Окно вывода личной информации подобно окну для клиента. Окно для изменения личных данных сотрудника представлено на рисунке 6.20.

The screenshot shows a window titled "Изменение личной информации" (Change personal information). It features a label "Изменить:" (Change:) on the left. To the right of this label are four input fields arranged in a 2x2 grid. The top-left field is labeled "ФИО" (Full name), the top-right field is labeled "Телефон" (Phone), the bottom-left field is labeled "Электронная почта" (Email), and the bottom-right field is labeled "Специализация" (Specialization). A "Подтвердить" (Confirm) button is located at the bottom right of the window.

Рисунок 6.20 – Окно изменения личной информации

Ещё одним функционалом, доступным специалисту, является действия с услугами, представляющий собой CRUD операции. Окно, реализующее этот функционал представлено на рисунке 6.21.

The screenshot shows a web interface titled "Выбирайте" (Choose). It features a dropdown menu labeled "Действия с услугами" (Actions with services) which is open, showing four options: "Добавить услугу" (Add service), "Изменить услугу" (Change service), "Просмотреть услуги" (View services), and "Удалить услугу" (Delete service). To the right of the dropdown are two buttons: "Просмотр личной информации" (View personal information) and "Изменение личной информации" (Change personal information). At the bottom right is a "Выход" (Exit) button.

Рисунок 6.21 – Окно действий с услугами

При выборе пункта «добавление услуги» перед пользователем открывается окно создания услуги, появляются панельки для добавления необходимых данных. Его вид представлен на рисунке 6.22.

The screenshot shows a web interface titled "Вводите" (Enter). It contains four input fields: "Наименование" (Name), "Стоимость" (Cost), "Дополнительная информация" (Additional information), and "Длительность" (Duration). A "Добавить" (Add) button is located at the bottom right.

Рисунок 6.22 – Окно добавления услуги

При выборе пункта «изменение услуги» перед пользователем открывается окно, представленное на рисунке 6.23. Выводятся все услуги, оказываемые сотрудником, а также появляются панели для записи изменяемых данных.

The 'Update Service' window displays a table with the following data:

Код услуги	Логин сотру...	Наименова...	Доп. инфор...	Стоимость	Длительность
5	mikamin	Стрижка		70.0	60
8	mikamin	Стрижка бор...		30.0	60

Below the table, there are input fields for editing:

- Добавить** (button)
- Код услуги** (input field)
- Стоимость** (input field)
- Наименование** (input field)
- Длительность** (input field)
- Дополнительная информация** (input field)
- Редактировать** (button)

Рисунок 6.23 – Окно изменения услуги

При выборе пункта «удаления услуги» перед пользователем открывается окно, представленное на рисунке 6.24. Выводятся все услуги, оказываемые сотрудником, а также появляется панель для записи кода услуги для удаления.

The 'Delete Service' window displays a table with the following data:

Код услуги	Логин сотр...	Наименова...	Доп. инфор...	Стоимость	Длительность
5	mikamin	Стрижка		70.0	60
8	mikamin	Стрижка бо...		30.0	60

Below the table, there are input fields for deletion:

- Добавить** (button)
- Код услуги для удаления** (input field)
- Удалить** (button)

Рисунок 6.24 – Окно удаления услуги

Ещё одним функционалом, доступным специалисту, является действия с расписанием. Для этой таблицы в базе данных специалисту доступно просмотр личного расписание и запись его в файл. Окно, реализующее этот функционал представлено на рисунке 6.25. Окно просмотра личного расписания представлено на рисунке 6.26.

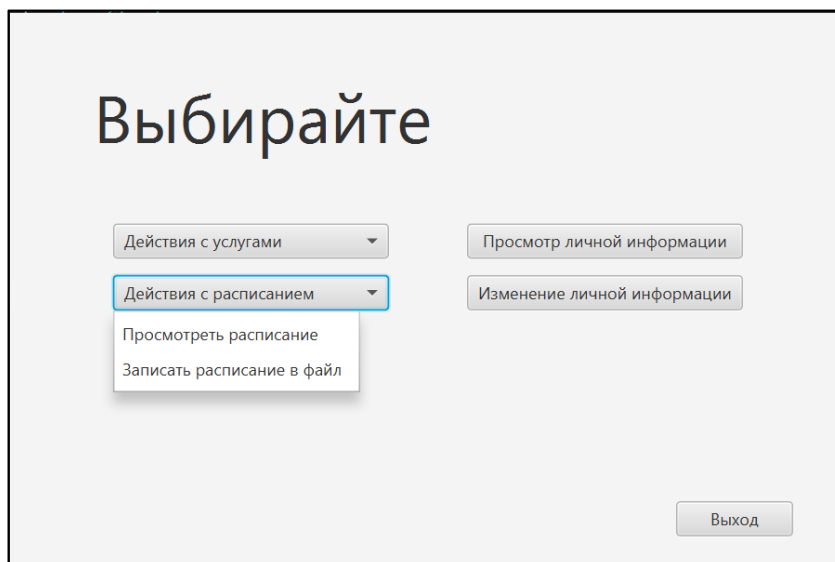


Рисунок 6.25 – Окно действия с расписанием

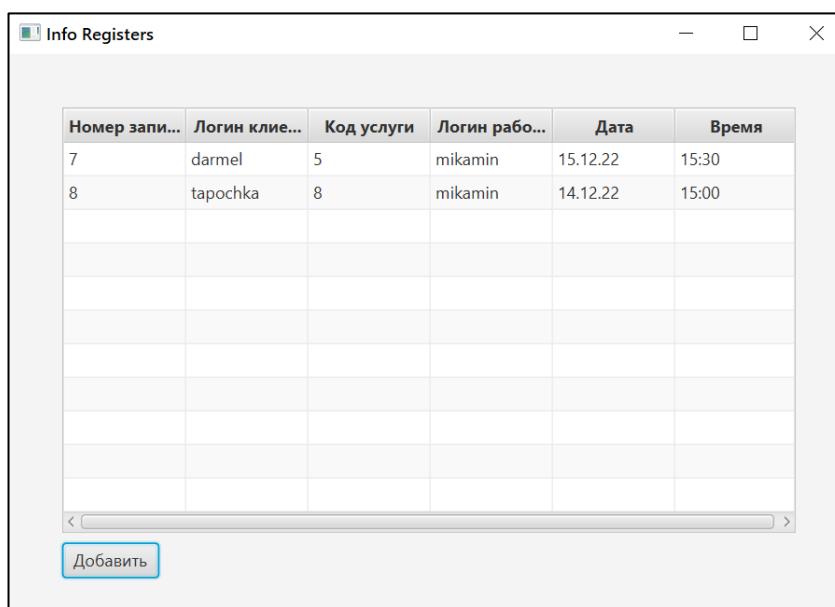


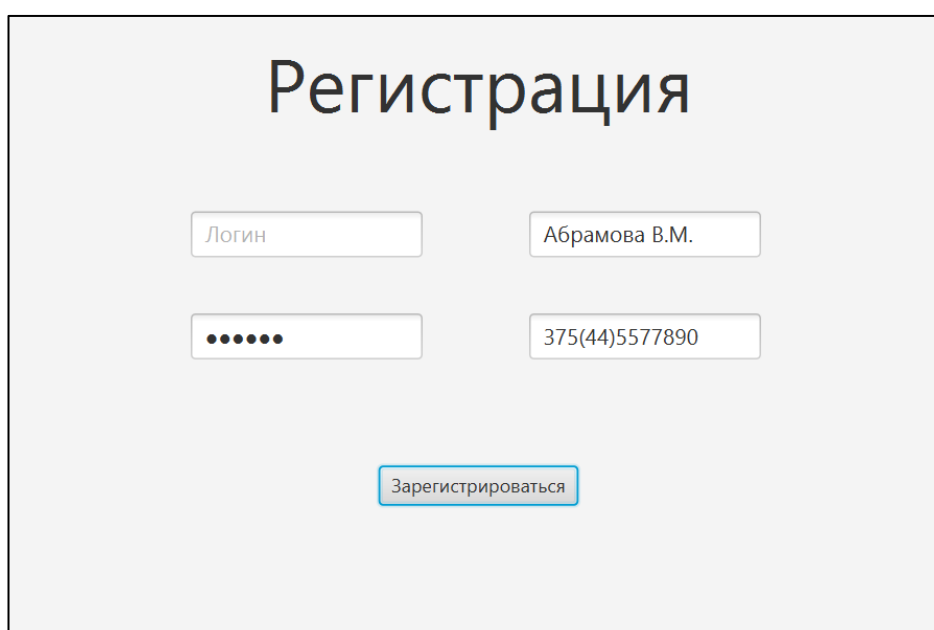
Рисунок 6.26 – Окно просмотра личного расписания

Таким образом, используя вышеописанный функционал, можно автоматизировать систему работы парикмахерской.

7 ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРИЛОЖЕНИЯ

В результате тестирования системы были выявлены и исправлены некоторые недочёты.

При запуске приложения возникает приветственное окно программы с полями для авторизации и кнопкой перехода на окно регистрации. При регистрации присутствует проверка на корректность введения логина, если введённый пользователем логин совпадает с уже имеющимся в базе, окно ввода логина очищается, и регистрация не проходит до тех пор, пока не будет введён не повторяющийся логин. На рисунке 7.1 приведено окно регистрации с очищенным полем ввода логина.



The image shows a registration window with a light gray background and a dark border. At the top center is the title "Регистрация" in a large, bold, dark font. Below the title are four input fields arranged in a 2x2 grid. The top-left field is labeled "Логин" and is empty. The top-right field contains the text "Абрамова В.М.". The bottom-left field contains six black dots, representing a password. The bottom-right field contains the phone number "375(44)5577890". Below these fields is a single button labeled "Зарегистрироваться" with a blue border and a light blue background.

Рисунок 7.1 – Окно регистрации

Другим учтённым недочётом является неверный ввод данных, при добавлении услуг в базу данных. Примером проверки ввода при добавлении услуг является проверка ввода стоимости услуги. Так, при введении отрицательной стоимости, добавление услуги не произойдёт, и поле будет очищаться до тех пор, пока не будет введено не отрицательное значение. Окно добавления услуги с отрицательной ценой представлено на рисунке 7.2.

Рисунок 7.2 – Окно проверки ввода стоимости

Таким образом в данной главе были продемонстрированы проверки ввода данных, используемые в приложении.

ЗАКЛЮЧЕНИЕ

Основной причиной автоматизации является превосходство машин над человеком при обработке информации, объемах обрабатываемой информации за один промежуток времени и скорости этой обработки. Автоматизация позволяет, если не полностью исключить человеческие ошибки, то, по крайней мере, свести их к минимуму, поэтому всё больше областей жизни человека автоматизируется.

Тема, выбранная для данного курсового проекта, является достаточно актуальной в наше время. Весь мир переходит на автоматизацию всех возможных процессов и индустрия красоты не исключение.

В результате разработки данного проекта была создана программа на языке Java, обеспечивающая автоматизированную систему работы парикмахерской, которая позволяет упростить связь сотрудников и клиентов, а также облегчить запись на услуги и составление расписаний. Также в ходе выполнения данной курсовой работы были получены и усвоены знания по работе с JavaFX. Было разработано клиент-серверное приложение, которое имеет возможность обрабатывать несколько клиентов одновременно. Была проведена работа с базой данных, хранящей всю необходимую для работы приложения информацию.

На сегодняшний день, приобретенные знания в ходе работы над проектом имеют важное значение. Так, например, клиент-серверные приложения окружают нас повсюду, и дальше это направление будет только расширяться. Базы данных также являются неотъемлемой частью многих создаваемых программ и приложений. Поэтому найти применения полученным навыкам в этой сфере не составит никакого труда. Исходя из этого, можно сделать вывод, что данный курсовой проект оставит важные знания, которые могут пригодиться в будущем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Методология IDEF0 – Учебная и научная деятельность Анисимова Владимира Викторовича [Электронный ресурс]. – Функциональное моделирование на основе стандарта IDEF0 – Режим доступа: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema6/tema6_2

[2] Студопедия – Методика построения информационной модели [Электронный ресурс]. – Информационная модель системы и её описание – Режим доступа: https://studopedia.su/14_21650_metodika-postroeniya-informatsionnoy-modeli.html

[3] Студопедия – Этапы проектирования базы данных [Электронный ресурс]. – Информационная модель системы и её описание – Режим доступа: https://studopedia.ru/2_5173_etapi-proektirovaniya-bazi-dannih.html

[4] Студопедия – Связь. Характеристики связей [Электронный ресурс]. – Информационная модель системы и её описание – Режим доступа: https://studopedia.ru/3_164850_svyaz-harakteristiki-svyazey.html

Диаграмма вариантов использования

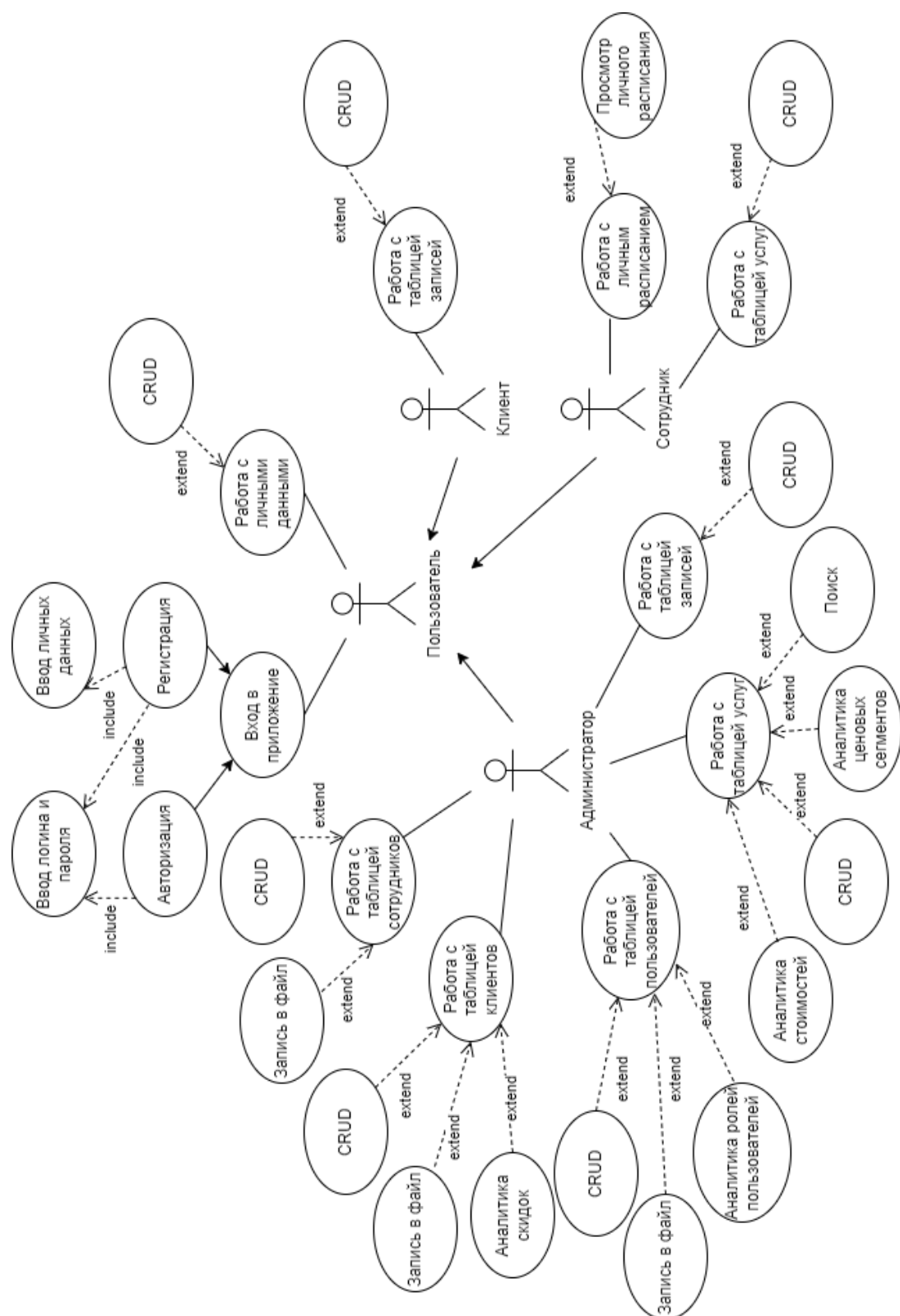


Рисунок Б.1 – Диаграмма вариантов использования

Приложение Б
(обязательное)
Диаграмма состояний

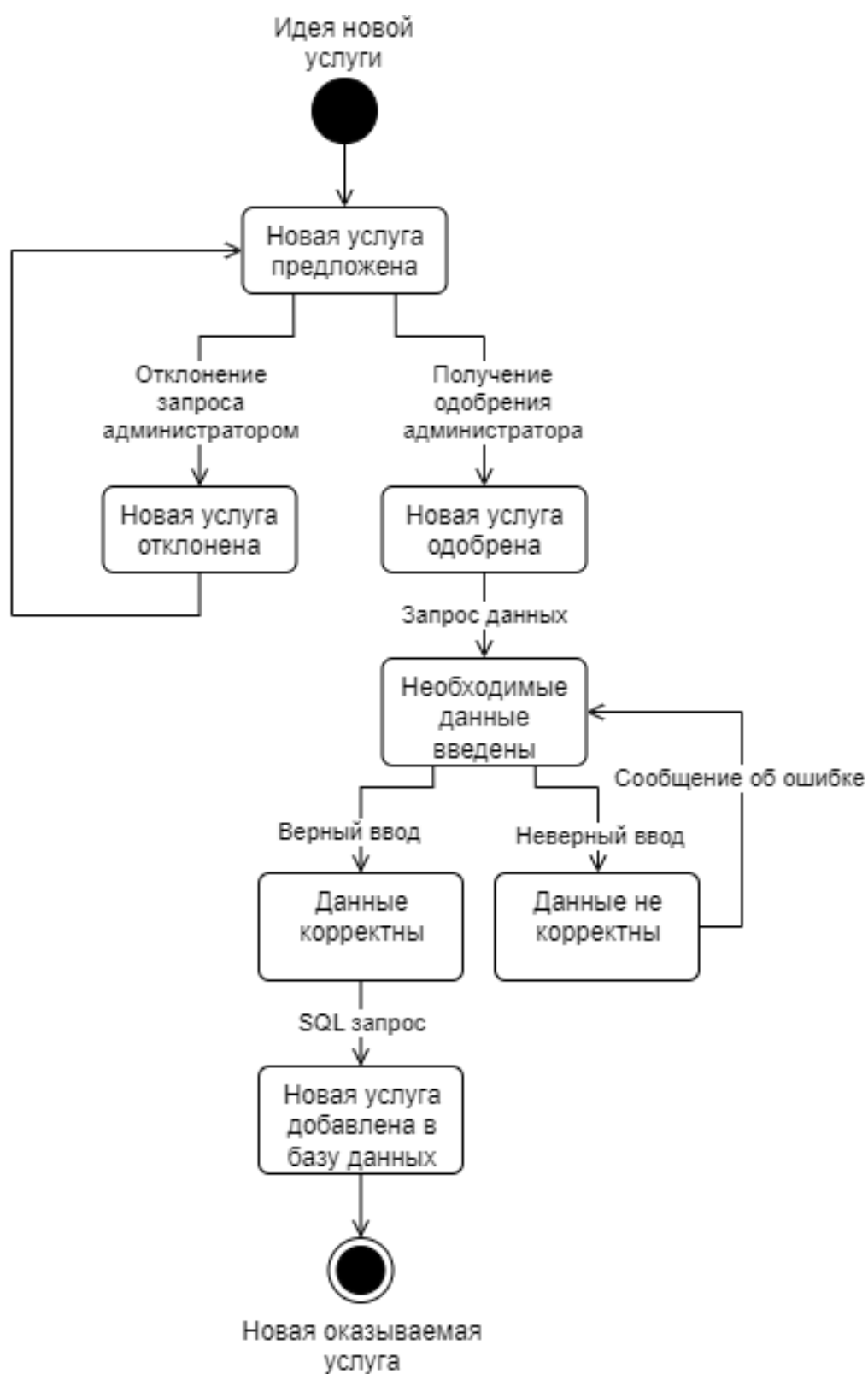


Рисунок Б.1 – Диаграмма состояний

Приложение В **(обязательное)** **Диаграмма последовательностей**

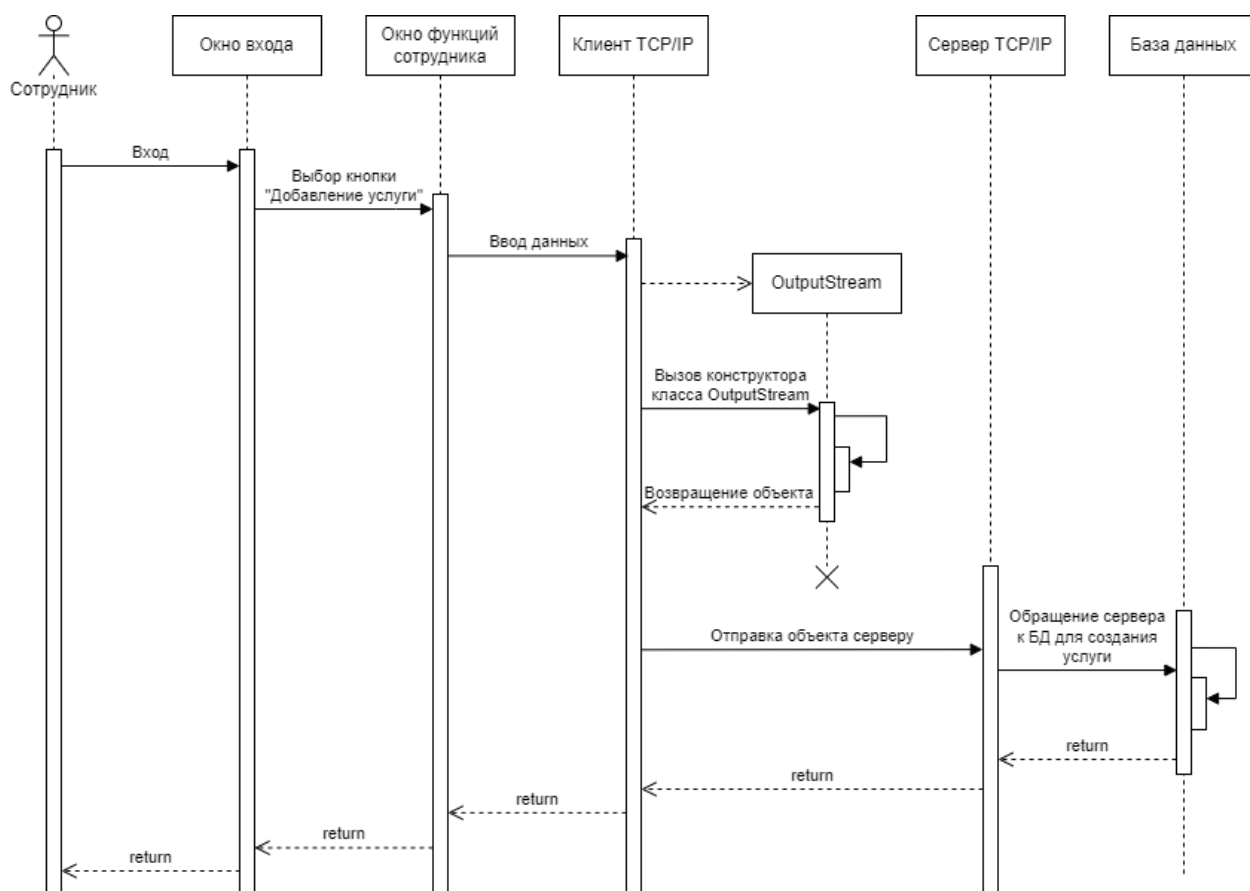


Рисунок В.1 – Диаграмма последовательностей

Приложение Г (обязательное) Диаграммы классов

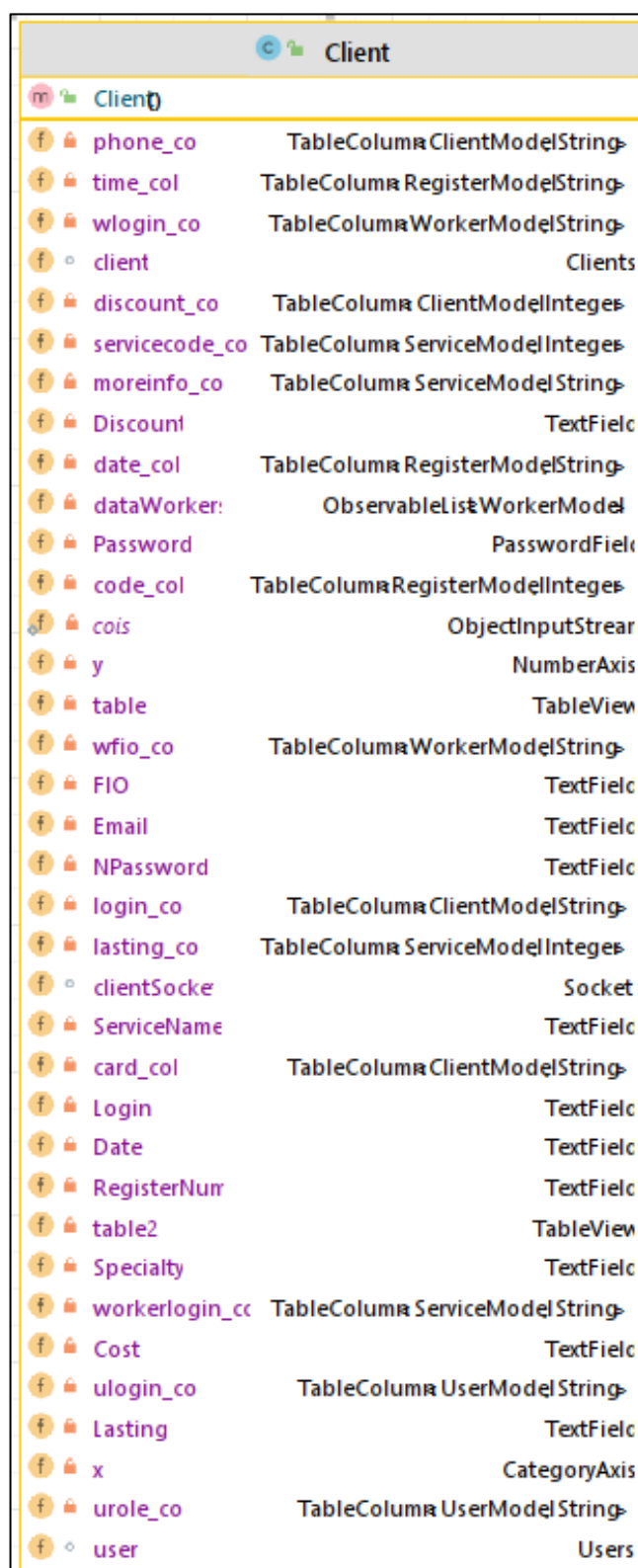


Рисунок Г.1 – Класс Client

Продолжение приложения Г















































































		<code>coos</code>	<code>ObjectOutputStream</code>
		<code>pieChart</code>	<code>PieChart</code>
		<code>barChart</code>	<code>BarChart</code> ?, ?>
		<code>MoreInfo</code>	<code>TextArea</code>
		<code>dataClient</code>	<code>ObservableList ClientModel</code>
		<code>dataRegiste</code>	<code>ObservableList RegisterModel</code>
		<code>dataService</code>	<code>ObservableList ServiceModel</code>
		<code>dataUsers</code>	<code>ObservableList UserModel</code>
		<code>clientLogin_cc</code>	<code>TableColumn RegisterModelString</code>
		<code>workerLogin_cc</code>	<code>TableColumn RegisterModelString</code>
		<code>email_co</code>	<code>TableColumn WorkerModelString</code>
		<code>ServiceCode</code>	<code>TextField</code>
		<code>regNum_col</code>	<code>TableColumn RegisterModelInteger</code>
		<code>specialty_co</code>	<code>TableColumn WorkerModelString</code>
		<code>cost_col</code>	<code>TableColumn ServiceModelDouble</code>
		<code>wphone_co</code>	<code>TableColumn WorkerModelString</code>
		<code>fio_col</code>	<code>TableColumn ClientModelString</code>
		<code>Phone</code>	<code>TextField</code>
		<code>servicename_co</code>	<code>TableColumn ServiceModelString</code>
		<code>Time</code>	<code>TextField</code>
		<code>updateClientDiscount</code>	<code>void</code>
		<code>buttonEnter(ActionEvent)</code>	<code>void</code>
		<code>SetupWorkerTable</code>	<code>void</code>
		<code>deleteClientDialog</code>	<code>void</code>
		<code>addServiceDialog</code>	<code>void</code>
		<code>deleteViewClient</code>	<code>void</code>
		<code>workerViewPersonalInfoDialog</code>	<code>void</code>
		<code>addAdmin</code>	<code>void</code>
		<code>servicesViewInfo</code>	<code>void</code>
		<code>deleteViewRegister</code>	<code>void</code>
		<code>analyticsServiceDialog</code>	<code>void</code>
		<code>CloseWindow</code>	<code>void</code>
		<code>deleteViewService</code>	<code>void</code>
		<code>SetupUserTable</code>	<code>void</code>
		<code>clientUpdatePersonalInfoDialog</code>	<code>void</code>
		<code>clientUpdatePersonalInfo(ActionEvent)</code>	<code>void</code>
		<code>SetupClientTable</code>	<code>void</code>
		<code>intoFileWorker</code>	<code>void</code>
		<code>analyticsCostDialog</code>	<code>void</code>

Рисунок Г.2 – Класс Client

Продолжение приложение Г

m	addService()	void
m	servicesViewDialog()	void
m	analyticsCost()	void
m	analyticsUsersDialog()	void
m	deleteWorker()	void
m	servicesSearchDialog()	void
m	registersViewClient()	void
m	updateViewRegister()	void
m	intoFileUser()	void
m	addRegisterView()	void
m	adminViewPersonalInfoDialog()	void
m	deleteRegisterDialog()	void
m	userViewInfo()	void
m	clientViewPersonalInfoDialog()	void
m	deleteClient()	void
m	SignUp()	void
m	updateRegister()	void
m	addRegisterDialog()	void
m	servicesSearchInfo()	void
m	deleteWorkerDialog()	void
m	SetupServiceTable()	void
m	workerUpdatePersonalInfoDialog()	void
m	clientsViewDialog()	void
m	registersViewDialog()	void
m	analyticsService()	void
m	SetupRegisterTable()	void
m	viewServicesWorker()	void
m	usersViewDialog()	void
m	deleteServiceDialog()	void
m	analyticsDiscountDialog()	void
m	workerUpdatePersonalInfo(ActionEvent)	void
m	updateServiceDialog()	void
m	clientViewPersonalInfo()	void
m	analyticsDiscount()	void
m	adminUpdatePersonalInfoDialog()	void
m	addAdminDialog()	void
m	addRegister()	void
m	workerViewDialog()	void
m	deleteViewWorkers()	void

Рисунок Г.3 – Диаграмма класса Client

Продолжение приложения Г





























		deleteRegister		void
		addWorker		void
		adminsViewDialog		void
		deleteService		void
		registersViewInfo		void
		analyticsUser		void
		buttonRegistration	ActionEvent	void
		workersViewInfo		void
		addWorkerDialog		void
		intoFileClient		void
		updateService		void
		registersViewClientDialog		void
		updateRegisterDialog		void
		updateClientDiscountDialog		void

Рисунок Г.4 – Класс Client

**Приложение Д
(обязательное)
Скрипт создания базы данных**

```
CREATE TABLE `user` (  
  `login` VARCHAR(30) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',  
  `password` VARCHAR(30) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',  
  `role` ENUM('администратор','клиент','специалист') NOT NULL DEFAULT  
  'клиент' COLLATE 'utf8mb4_0900_ai_ci',  
  PRIMARY KEY (`login`) USING BTREE )  
COLLATE='utf8mb4_0900_ai_ci'  
ENGINE=InnoDB;  
  
CREATE TABLE `worker` (  
  `workerLogin` VARCHAR(30) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',  
  `workerFIO` VARCHAR(100) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',  
  `phone` VARCHAR(14) NOT NULL DEFAULT '375 (XX)XXXXXXX' COLLATE  
  'utf8mb4_0900_ai_ci',  
  `e-mail` VARCHAR(50) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',  
  `specialty` VARCHAR(30) NOT NULL DEFAULT 'Администратор' COLLATE  
  'utf8mb4_0900_ai_ci',  
  PRIMARY KEY (`workerLogin`) USING BTREE,  
  CONSTRAINT `FK_worker_user` FOREIGN KEY (`workerLogin`) REFERENCES  
  `user` (`login`) ON UPDATE CASCADE ON DELETE CASCADE )  
COLLATE='utf8mb4_0900_ai_ci'  
ENGINE=InnoDB ;  
  
CREATE TABLE `client` (  
  `clientLogin` VARCHAR(30) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',  
  `clientFIO` VARCHAR(100) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',  
  `phone` VARCHAR(14) NOT NULL DEFAULT '375 (XX)XXXXXXX' COLLATE  
  'utf8mb4_0900_ai_ci',  
  `clientCardNumber` VARCHAR(14) NOT NULL DEFAULT '375 (XX)XXXXXXX'  
  COLLATE 'utf8mb4_0900_ai_ci',  
  `discount` TINYINT(3) NULL DEFAULT NULL,
```


Продолжение приложения Д

```
PRIMARY KEY (`clientLogin`) USING BTREE,  
  
CONSTRAINT `FK_client_user` FOREIGN KEY (`clientLogin`) REFERENCES  
`user` (`login`) ON UPDATE CASCADE ON DELETE CASCADE )  
  
COLLATE='utf8mb4_0900_ai_ci'  
  
ENGINE=InnoDB ;  
  
CREATE TABLE `service` (  
  
  `serviceCode` INT(10) NOT NULL AUTO_INCREMENT,  
  
  `workerLogin` VARCHAR(30) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',  
  
  `serviceName` VARCHAR(20) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',  
  
  `moreInfo` VARCHAR(300) NULL DEFAULT NULL COLLATE  
'utf8mb4_0900_ai_ci',  
  
  `cost` DOUBLE NOT NULL,  
  
  `lasting` INT(10) NOT NULL,  
  
  PRIMARY KEY (`serviceCode`) USING BTREE,  
  
  INDEX `FK_service_worker` (`workerLogin`) USING BTREE,  
  
  CONSTRAINT `FK_service_worker` FOREIGN KEY (`workerLogin`) REFERENCES  
  `worker` (`workerLogin`) ON UPDATE CASCADE ON DELETE CASCADE )  
  
  COLLATE='utf8mb4_0900_ai_ci'  
  
  ENGINE=InnoDB  
  
  AUTO_INCREMENT=9;  
  
  CREATE TABLE `register` (  
  
    `registerNumber` INT(10) NOT NULL AUTO_INCREMENT,  
  
    `clientLogin` VARCHAR(30) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',  
  
    `serviceCode` INT(10) NOT NULL,  
  
    `workerLogin` VARCHAR(30) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',  
  
    `date` VARCHAR(10) NOT NULL DEFAULT 'xx.xx.xxxx' COLLATE  
'utf8mb4_0900_ai_ci',  
  
    `time` VARCHAR(5) NOT NULL DEFAULT 'xx:xx' COLLATE  
'utf8mb4_0900_ai_ci',
```

Продолжение приложения Д

```
PRIMARY KEY (`registerNumber`) USING BTREE,  
  
INDEX `FK_register_client` (`clientLogin`) USING BTREE,  
  
INDEX `FK_register_service` (`serviceCode`) USING BTREE,  
  
INDEX `FK_register_service_2` (`workerLogin`) USING BTREE,  
  
CONSTRAINT `FK_register_client` FOREIGN KEY (`clientLogin`) REFERENCES  
`client` (`clientLogin`) ON UPDATE CASCADE ON DELETE CASCADE,  
  
CONSTRAINT `FK_register_service` FOREIGN KEY (`serviceCode`)  
REFERENCES `service` (`serviceCode`) ON UPDATE CASCADE ON DELETE  
CASCADE,  
  
CONSTRAINT `FK_register_service_2` FOREIGN KEY (`workerLogin`)  
REFERENCES `service` (`workerLogin`) ON UPDATE CASCADE ON DELETE  
CASCADE )  
  
COLLATE='utf8mb4_0900_ai_ci'  
  
ENGINE=InnoDB  
  
AUTO_INCREMENT=9;
```

Приложение Е
(обязательное)
Листинг программного кода

Сервер:

```
package Server;

import dao.ConnectionFactory;
import java.io.*;
import java.net.*;

public class Server {
    public static void main(String[] arg) {
        ConnectionFactory.Connect();

        try (ServerSocket serverSocket = new ServerSocket(2525)) {
            System.out.println("Сервер начал работу");
            while (true) {
                Socket finalClientAccepted = serverSocket.accept();
                ClientThread clientThread = new
ClientThread(finalClientAccepted);
                new Thread(clientThread).run();
            }
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}

package Server;

import authorization.LogIn;
import dao.*;
import model.*;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.List;
import java.util.Objects;

public class ClientThread implements Runnable {

    private Socket client;
    private ObjectInputStream cois;
    private ObjectOutputStream coos;
```

Продолжение приложения E

```
public ClientThread() {
}

public ClientThread(Socket client) {
    this.client = client;
    this.cois = null;
    this.coos = null;
}

@Override
public void run() {
    try {
        cois = new ObjectInputStream(client.getInputStream());
        coos = new ObjectOutputStream(client.getOutputStream());
    } catch (IOException e) {
        e.printStackTrace();
        System.out.println("Connection refused");
        return;
    }
    String clientCommand;
    String login = "", role = "";
    try {
        String loginType = (String) cois.readObject();
        if (loginType.equals("login")) {
            LogIn logIn = new LogIn();
            login = (String) cois.readObject();
            String password = (String) cois.readObject();
            role = logIn.authorization(login, password);
            coos.writeObject(role);
        } else if (loginType.equals("register")) {
            login = (String) cois.readObject();
            String password = (String) cois.readObject();
            role = "клиент";
            String FIO = (String) cois.readObject();
            String phone = (String) cois.readObject();
            user = new Users(login, password, role);
            clients = new Clients(login, FIO, phone, phone);
            userDao.addUser(user, clients);
            coos.writeObject("Здравствуйте, " + login);
        }
        switch (role) {
            case "администратор":{
                ...
            }
            case "специалист":{
                ...
            }
        }
    }
}
```

Продолжение приложения Е

```
        }
        case "клиент":{
            ...
        }
    }
    System.out.println("Exit");
    client.close();
} catch (Exception e) {
    e.printStackTrace();
}
}
```

Клиент:

```
package Client;

import javafx.application.Application;
import javafx.stage.Stage;

public class ClientApplication extends Application {
    @Override
    public void start(Stage stage) throws Exception {
        BasicWindowAction basicWindowActions=new BasicWindowAction();
        basicWindowActions.OpenWindow("HelloWindow.fxml", 600,400);
    }
    public static void main(String[] args) {
        launch();
    }
}
```

FXML файл AdminMenu:

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.MenuButton?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>

<AnchorPane          prefHeight="400.0"          prefWidth="600.0"
xmlns="http://javafx.com/javafx/19" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="Client.Client">
    <children>
        <Button layoutX="374.0" layoutY="203.0" mnemonicParsing="false"
onAction="#adminViewPersonalInfoDialog"          prefHeight="25.0"
prefWidth="198.0" text="Просмотр личной информации" />
```

Продолжение приложения Е

```
<Button layoutX="374.0" layoutY="240.0" mnemonicParsing="false"
onAction="#adminUpdatePersonalInfoDialog" text="Изменение личной
информации" />
<Label layoutX="70.0" layoutY="51.0" text="Выбирайте">
    <font>
        <Font size="48.0" />
    </font>
</Label>
<Button layoutX="488.0" layoutY="350.0" mnemonicParsing="false"
onAction="#CloseWindow" prefHeight="25.0" prefWidth="84.0" text="Выход"
/>

<MenuButton layoutX="70.0" layoutY="169.0"
mnemonicParsing="false" prefHeight="25.0" prefWidth="198.0"
text="Действия с сотрудниками">
    <items>
        <MenuItem mnemonicParsing="false"
onAction="#addWorkerDialog" text="Добавить сотрудника" />
        <MenuItem mnemonicParsing="false"
onAction="#addAdminDialog" text="Добавить администратора" />
        <MenuItem mnemonicParsing="false"
onAction="#adminsViewDialog" text="Просмотреть список сотрудников" />
        <MenuItem mnemonicParsing="false"
onAction="#deleteWorkerDialog" text="Удалить сотрудника или
администратора" />
    </items>
</MenuButton>

<MenuButton layoutX="70.0" layoutY="209.0"
mnemonicParsing="false" prefHeight="25.0" prefWidth="198.0"
text="Действия с клиентами">
    <items>
        <MenuItem mnemonicParsing="false"
onAction="#clientsViewDialog" text="Просмотреть список клиентов" />
        <MenuItem mnemonicParsing="false"
onAction="#deleteClientDialog" text="Удалить клиента" />
        <MenuItem mnemonicParsing="false"
onAction="#intoFileClients" text="Составить файл" />
        <MenuItem mnemonicParsing="false"
onAction="#updateClientDiscountDialog" text="Добавить скидку" />
        <MenuItem mnemonicParsing="false"
onAction="#analyticsDiscountDialog" text="Аналитика скидок" />
    </items>
</MenuButton>

<MenuButton layoutX="70.0" layoutY="134.0"
mnemonicParsing="false" prefHeight="25.0" prefWidth="198.0"
text="Действия с пользователями">
    <items>
```

Продолжение приложения Е

```
<MenuItem mnemonicParsing="false" onAction="#usersViewDialog"
text="Просмотреть список пользователей" />
    <MenuItem mnemonicParsing="false" onAction="#intoFileUsers"
text="Составить файл" />
    <MenuItem mnemonicParsing="false"
onAction="#analyticsUsersDialog" text="Аналитика ролей" />
</items>
</MenuButton>
<MenuButton layoutX="70.0" layoutY="252.0"
mnemonicParsing="false" prefHeight="25.0" prefWidth="198.0"
text="Действия с услугами">
    <items>
        <MenuItem mnemonicParsing="false"
onAction="#servicesViewDialog" text="Просмотреть услуги" />
        <MenuItem mnemonicParsing="false"
onAction="#deleteServiceDialog" text="Удалить услугу" />
        <MenuItem mnemonicParsing="false"
onAction="#servicesSearchDialog" text="Поиск по наименованию" />
        <MenuItem mnemonicParsing="false"
onAction="#analyticsServiceDialog" text="Аналитика стоимости" />
        <MenuItem mnemonicParsing="false"
onAction="#analyticsCostDialog" text="Аналитика ценового сегмента" />
    </items>
</MenuButton>
<Button layoutX="70.0" layoutY="288.0" mnemonicParsing="false"
onAction="#registersViewDialog" prefHeight="25.0" prefWidth="198.0"
text="Просмотреть список записей" />
</children>
</AnchorPane>
```

Вызывающая AdminMenu функция:

```
public void buttonEnter(ActionEvent event) {
    String str = "", str2 = "hello world";
    try {
        clientSocket = new Socket("127.0.0.1", 2525);
        coos = new ObjectOutputStream(clientSocket.getOutputStream());
        cois = new ObjectInputStream(clientSocket.getInputStream());
        coos.writeObject("login");
        coos.writeObject(Login.getText());
        coos.writeObject>Password.getText());
        str = (String) cois.readObject();
        if (Objects.equals(str, "администратор")) {
            str2 = "Admin";
        }
    }
}
```

Продолжение приложения Е

```
    } else if (Objects.equals(str, "клиент")) {
        str2 = "Client";
    } else if (Objects.equals(str, "специалист")) {
        str2 = "Worker";
    }
} catch (Exception e) {
    e.printStackTrace();
}
try {
    System.out.println("Connected! " + str2);
    FXMLLoader fxmlLoader = new
FXMLLoader(ClientApplication.class.getResource(str2 + "Menu.fxml"));
    Stage stage = (Stage)
event.getSource().getScene().getWindow();
    Scene scene = new Scene(fxmlLoader.load(), stage.getWidth(),
stage.getHeight());
    stage.setTitle("Menu");
    stage.setScene(scene);
    stage.show();

} catch (IOException e) {
    e.printStackTrace();
}
}
```