

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

_____. Лыщик А.П.
_____.2022

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему:
«РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЗАЦИИ РАБОТЫ
КОСМЕТИЧЕСКОГО САЛОНА»

БГУИР КП 1-40 01 02-08 015 ПЗ

Выполнил студент группы

(подпись студента)
Курсовой проект представлен на
проверку _____.2022

(подпись студента)

Минск 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 АНАЛИЗ РАБОТЫ КОСМЕТИЧЕСКОГО САЛОНА	4
1.1 Общие понятия и значение салона красоты в современном мире	4
1.2 Классификация салонов красоты по ценовой категории	5
1.3 Бизнес-процессы салона красоты.....	6
2 ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЕ РЕШЕНИЯ	8
3 МОДЕЛИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА	9
3.1 Функциональное моделирование на основе стандарта IDEF	9
4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЁ ОПИСАНИЕ	15
5 МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ.....	17
5.1 Диаграмма вариантов использования	17
5.2 Диаграмма последовательностей	18
5.3 Диаграмма классов.....	20
5.4 Диаграмма компонентов	22
5.5 Диаграмма развертывания	22
6 ОПИСАНИЕ АЛГОРИТМОВ РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СИСТЕМЫ	24
6.1 Алгоритм авторизации.....	24
6.2 Алгоритм получения записей	25
7 РУКОВОДСТВО РАЗВЕРТЫВАНИЯ.....	26
8 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ	27
9 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ.....	37
ЗАКЛЮЧЕНИЕ	41
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	42
ПРИЛОЖЕНИЕ А	43
ЛИСТИНГ ПРОГРАММЫ	43
ПРИЛОЖЕНИЕ Б.....	53

ВВЕДЕНИЕ

Автоматизировать управление салонами и косметологическими центрами - значит ежедневно экономить время на выполнении рутинных операций и получении аналитических данных о состоянии дел в салоне или клинике. [1]

В современных условиях эффективное управление представляет собой ценный ресурс организации, наряду с финансовыми, материальными, человеческими и другими ресурсами. Следовательно, повышение эффективности управленческой деятельности становится одним из направлений совершенствования деятельности предприятия в целом. [2]

В любой организации учет служит для того, чтобы иметь контроль над всеми процессами. Поэтому уже сейчас опытные и грамотные предприниматели внедряют автоматизированные программы управления, которые способны дать такую же информацию, но буквально за пару минут, ведь все необходимые статистические данные уже содержатся в них, а заданные параметры анализируются с помощью компьютерных программ. [3]

Высокая оперативность данного вида учёта обеспечивается за счёт краткости и быстроты, поскольку оперативный учёт не предполагает обязательного документирования операций.

Таким образом, целью данного курсового проекта является повышение эффективности работы сотрудников косметического салона посредством автоматизации процесса реализации заказа.

Поставленная цель требует решения следующих задач:

- Проанализировать заданную предметную область;
- Реализовать клиент-серверное взаимодействие;
- Создать базу данных;
- Создать веб-приложение;
- Создать простой и удобный пользовательский интерфейс;

1 АНАЛИЗ РАБОТЫ КОСМЕТИЧЕСКОГО САЛОНА

1.1 Общие понятия и значение салона красоты в современном мире

Салон красоты – заведение, занимающееся косметическим обслуживанием мужчин и женщин. К этой же ветви обслуживания относятся салоны причесок и спа. Салоны красоты в отличие от них занимаются в основном работой с лицами и телами людей.

Отличие салона красоты от парикмахерской заключается в том, что парикмахерская специализируется исключительно на волосах, в то время, как салон красоты является многопрофильным заведением, в котором предлагают всевозможные процедуры по уходу за лицом и телом.

Сегодня салон красоты нередко представляет собой многопрофильное заведение и не ограничивает себя лишь косметическими услугами. Современные салоны красоты предлагают своим клиентам услуги аппаратной медицины, ряд терапевтических программ.

Услуги салонов:

- Парикмахерские услуги (мужская, женская, детская стрижка);
- Маникюр и педикюр (уход, наращивание, дизайн, коррекция);
- Косметологические услуги (уход за кожей и чистка лица, уход за бровями, уход за ресницами, эпиляция, пилинг, перманентный макияж);
- Уход за кожей тела;
- Солярий;
- Продажа косметических средств;

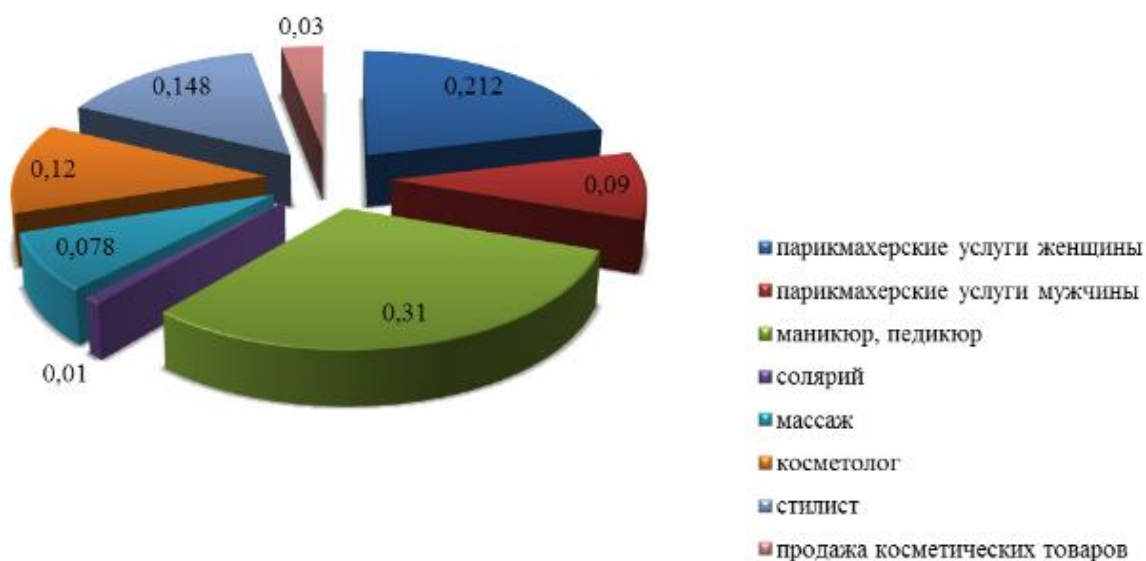


Рисунок 1.1 Распространённые услуги салона красоты

Оказание услуг салона красоты производится с помощью предварительной телефонной записи или по живой очереди без записи. В качестве методов рекламы планируется использование основного канала привлечения клиентов – выбранные социальные сети (Facebook, Вконтакте, Instagram). Данный выбор обусловлен отсутствием постоянных издержек (ведение групп) и большой концентрацией целевой аудитории (таргетинговая реклама).

1.2 Классификация салонов красоты по ценовой категории

Существуют три основных типа салонов красоты: эконом-класса, бизнес-класса, премиум, или люкс-класса.

Салоны эконом-класса специализируются на оказании малозатратных и, как правило, традиционных услуг. Это окрашивание, стрижка. Кроме того, в таких салонах есть и дополнительные услуги – маникюр и педикюр. В салонах работают, как правило, начинающие: студенты или мастера с небольшим стажем работы. Следует заметить, что цены в салонах эконом-класса соответствующие – среди всех категорий салонов данная остается самой дешевой. Клиентами такого заведения чаще всего становятся жители соседних домов или люди, работающие поблизости. Главная характеристика классической парикмахерской – удобное расположение и график работы. Постоянные клиенты в таких местах, как правило, пользуются услугами «своего» мастера, знающего их предпочтения. [4]

В салонах бизнес-класса более широкий перечень услуг с преобладанием профессиональной косметики для волос. Также в салонах данного уровня оказываются дополнительные услуги в сфере ногтевого сервиса: наращивание ногтей и нанесение художественного рисунка.

По сравнению с обычной парикмахерской такой салон предлагает расширенный перечень услуг, куда входят, кроме стрижки и окрашивания, маникюр, педикюр, услуги косметического кабинета и солярия. В подобных заведениях работает более квалифицированный персонал. Там можно получить консультацию мастера, который хорошо ориентируется в модных тенденциях.

Салоны бизнес-класса привлекают качеством услуг и индивидуальным подходом, за что, собственно, клиенты и готовы платить. Как правило, в салонах данного класса есть магазин сопутствующих товаров, где после консультации с мастером можно купить средства по уходу за волосами, а также аксессуары.

В салонах премиум-класса предоставляются эксклюзивные услуги – по уходу за волосами, кожей тела и лица, а также SPA-процедуры, ароматерапия, массаж всех видов (антицеллюлитный, расслабляющий, корректирующий и т.д.). Следует заметить, что все предоставляемые услуги оказывают высококлассные мастера. Особенность салонов премиум-класса – предоставление услуг стилиста. Стилист помогает качественно и быстро создать образ, учитывая современные тенденции моды.

Отличительный признак салонов красоты высшего класса – эксклюзивные услуги. Как правило, все салоны класса «люкс» авторские, т.е. работают под руководством и маркой известного парикмахера-стилиста. Кроме высокого уровня обслуживания, здесь вам предложат создание нового образа, широкую гамму услуг по уходу за волосами, кожей лица и тела. Все услуги строго и абсолютно индивидуальны.

1.3 Бизнес-процессы салона красоты

Для успешного достижения результата – оказания услуги клиенту, каждый бизнес процесс должен быть отлажен и продуман. Даже в самом маленьком салоне, с двумя мастерами и одним администратором ежедневно протекает больше десятка бизнес процессов.

Каждый салон ведет учет клиентов своими удобным способом – кто-то записывает в ежедневник или журнал, кто-то ведет таблицу в Excel, а некоторые уже внедрили CRM-системы. Этот процесс есть в каждом салоне, но, как и учет клиентов – реализуется по-разному. Чем проще и понятнее будет система записи к мастеру, тем больше шанс удовлетворить потребности всех клиентов. Также необходимо вести учет и периодически проводить анализ, чтобы не попасть в неудобную ситуацию – нет запаса дезинфицирующих средств для уборки, закончился обезжириватель, нет безворсовых салфеток или закончились крафт-пакеты. [5]

В салонах ежедневно происходят финансовые операции – оплата клиентами услуг и товаров, поступление материалов, налоговые отчисления, выплата зарплат, текущие затраты на рекламу и ремонт. Если процесс не налажен, как следует, велика вероятность не понять, куда уходят деньги и почему выручка у салона меньше, чем в прошлом месяце.

Нельзя забывать и об контроле качества. Контроль качества, как бизнес процесс – это возможность найти слабые точки и устранить их. Контролировать необходимо все: общение и обращение с клиентами, качество услуги, качество используемой продукции, уровень сервиса.

Неудивительно, что для работы они нуждаются в базе данных. Имеется

много различных СУБД, чтобы работать с ними, однако хорошо было бы создать приложение, удобное в использовании, которое позволяло бы работать с базой данных учета заказов клиента, не используя при этом установленную СУБД. Данная курсовая работа посвящена этому.

2 ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЕ РЕШЕНИЯ

Задачей курсового проекта является разработка системы автоматизации работы косметического салона – создание веб-приложения с организацией клиент-серверного взаимодействия, взаимодействия с базой данных. Для ее достижения необходимо выполнить следующие задачи:

- Создать простой и удобный пользовательский интерфейс;
- Спроектировать и создать иерархию классов;
- Создать базу данных;
- Использовать инкапсуляцию, перегрузку методов, переопределение методов, статические методы, обработку исключительных ситуаций;
- В данном программном продукте необходимо обеспечить добавление, редактирование и удаление записей из базы данных;
- Предусмотреть авторизацию пользователей;

Интерфейс пользователя должен быть понятным, простым для восприятия, доступным людям, не имеющим опыта работы с программным обеспечением.

Со стороны администратора программа должна обеспечивать следующие возможности:

- просматривать, добавлять и удалять информацию о пользователях;
- просматривать, добавлять и удалять информацию об администраторах;
- просматривать, добавлять и удалять информацию о клиентах;
- просматривать, добавлять и удалять информацию о записях;
- осуществлять поиск записей;
- просматривать статистику о клиентах;

Со стороны клиента программа предоставляет следующие возможности:

- предоставлять расписание актуальных услуг;
- предоставлять информацию о сотрудниках;
- создавать личный кабинет;
- предоставлять информацию о своих записях;
- предоставлять возможность сохранения своих записей в файл;

Важным аспектом системы автоматизации работы косметического салона является база данных, которая хранит в себе всю информацию. Поэтому, необходимо обеспечить удобное взаимодействие.

Таким образом были описаны основные задачи, на основе которых будет произведено моделирование разрабатываемой системы.

3 МОДЕЛИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

3.1 Функциональное моделирование на основе стандарта IDEF

IDEF0 — нотация графического моделирования, используемая для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающих эти функции.

Главными компонентами модели являются диаграммы. На них отображаются функции системы в виде прямоугольников, а также связи между ними и внешней средой посредством стрелок. IDEF0 позволяет подключить и активизировать деятельность заказчика по описанию бизнес – процессов с использованием формального и наглядного графического языка.

На рисунке 3.1.1 представлена контекстная диаграмма системы.

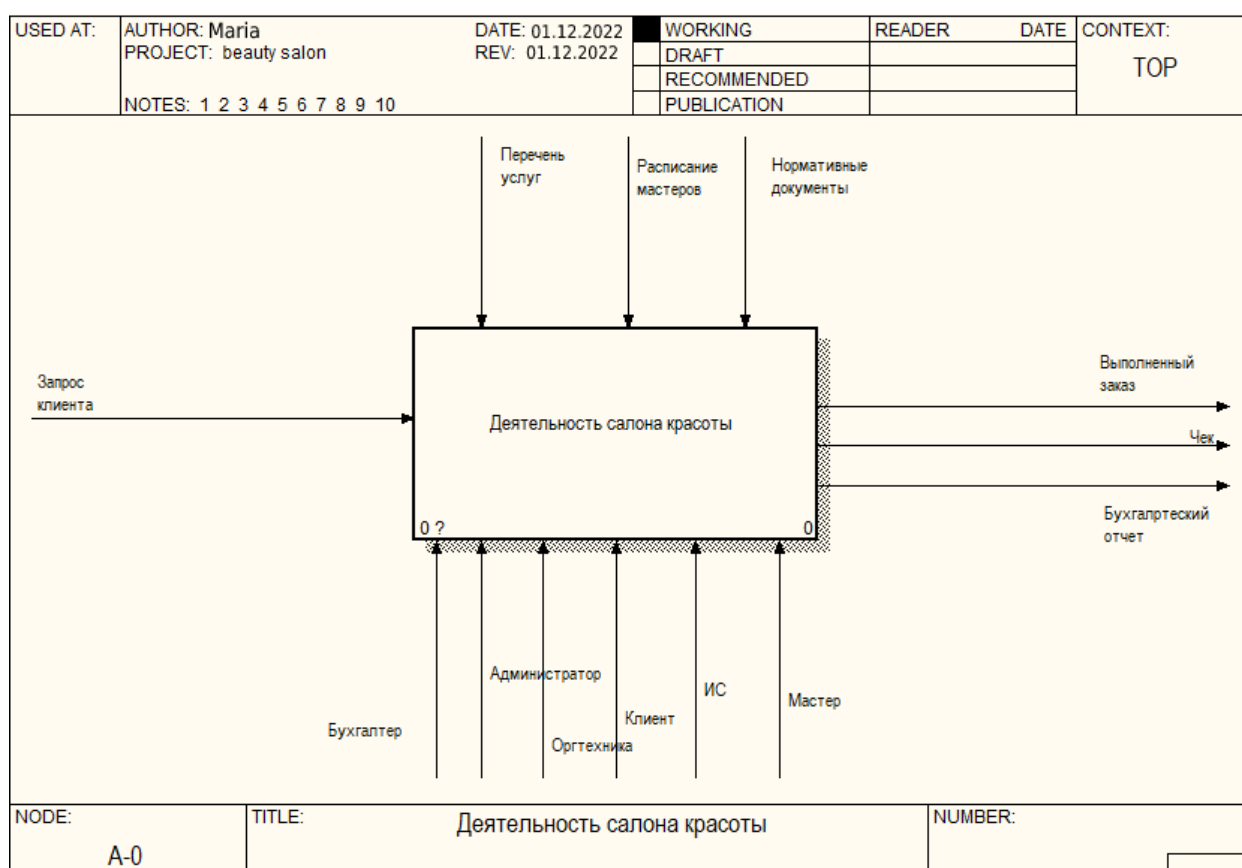


Рисунок 3.1.1 – Контекстная диаграмма

Входной поток включает в себя запрос клиента. После соответствующей обработки в выводном потоке имеем выполненный заказ, чек и бухгалтерский отчет. В роли управляющих воздействий выступает перечень услуг, расписание мастеров и нормативные документы. Механизмами являются

клиент, администратор, бухгалтер, мастер, оргтехника и информационная система.

На рисунке 3.1.2 отображена декомпозиция контекстной диаграммы, состоящая из четырех блоков: «Прием заказа», «Обработка заказа», «Выполнение заказа», «Оплата».

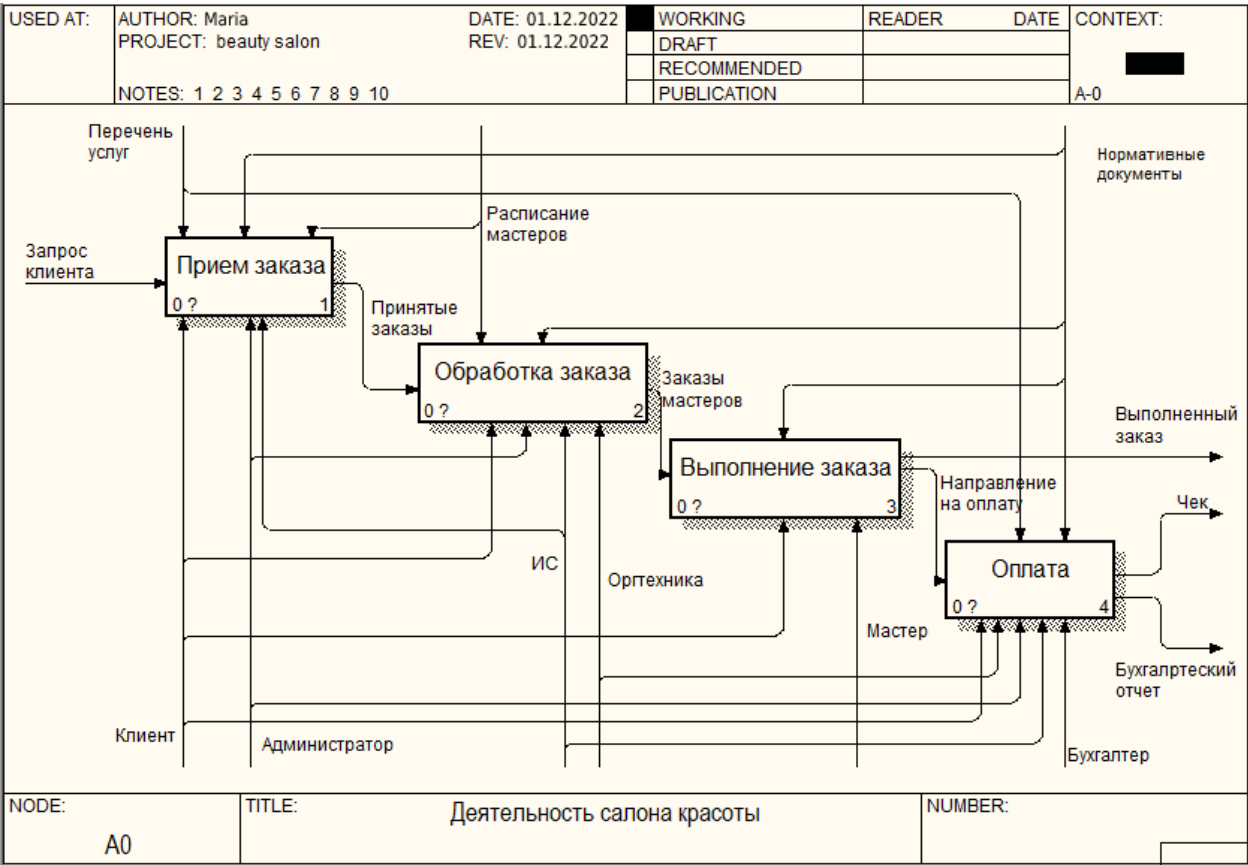


Рисунок 3.1.2 – Диаграмма декомпозиции главного процесса

На рисунке 3.1.3 отображена декомпозиция блока «Прием заказа», состоящая из трех блоков: «Выбор даты на услугу», «Выбор мастера», «Запись контактных данных».

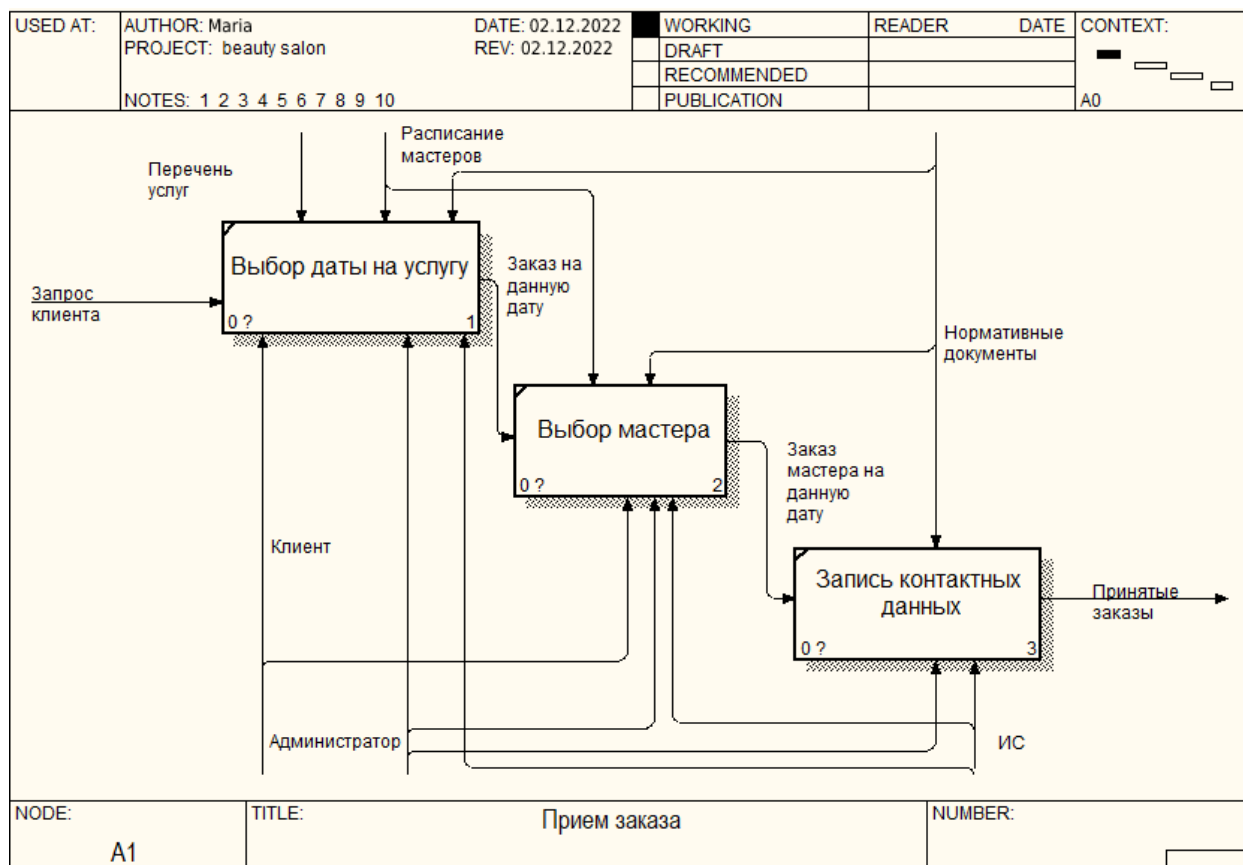


Рисунок 3.1.3 – Диаграмма декомпозиции блока «Прием заказа»

На рисунке 3.1.4 отображена декомпозиция блока «Обработка заказа». Она представлена тремя компонентами: «Обзвон клиентов», «Изменение расписания» и «Распечатать и выдать список заказов мастеру».

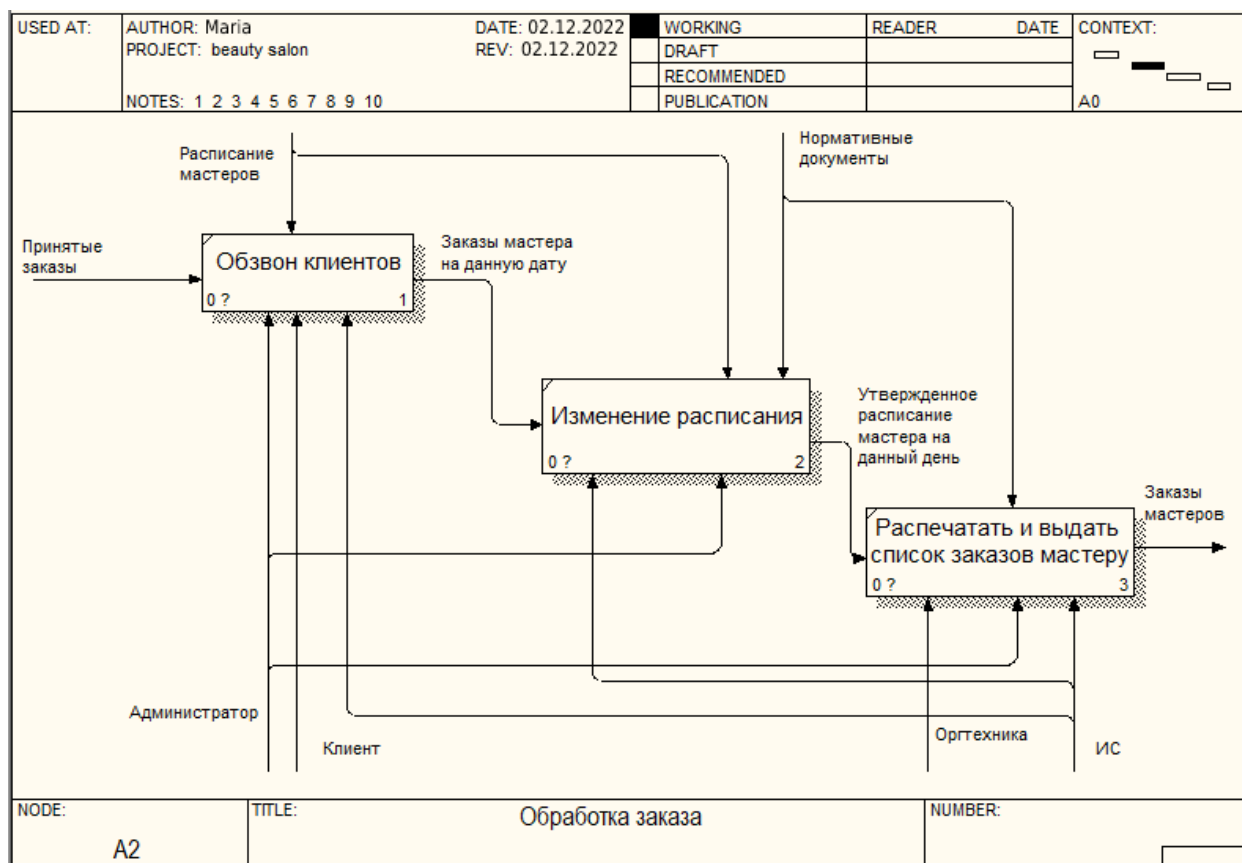


Рисунок 3.1.4 – Диаграмма декомпозиции процесса «Обработка заказа»

Процесс выполнение заказа состоит из трёх действий: обсуждение желаемого результата, выполнение косметических услуг, консультация по уходу (рисунок 3.1.5).

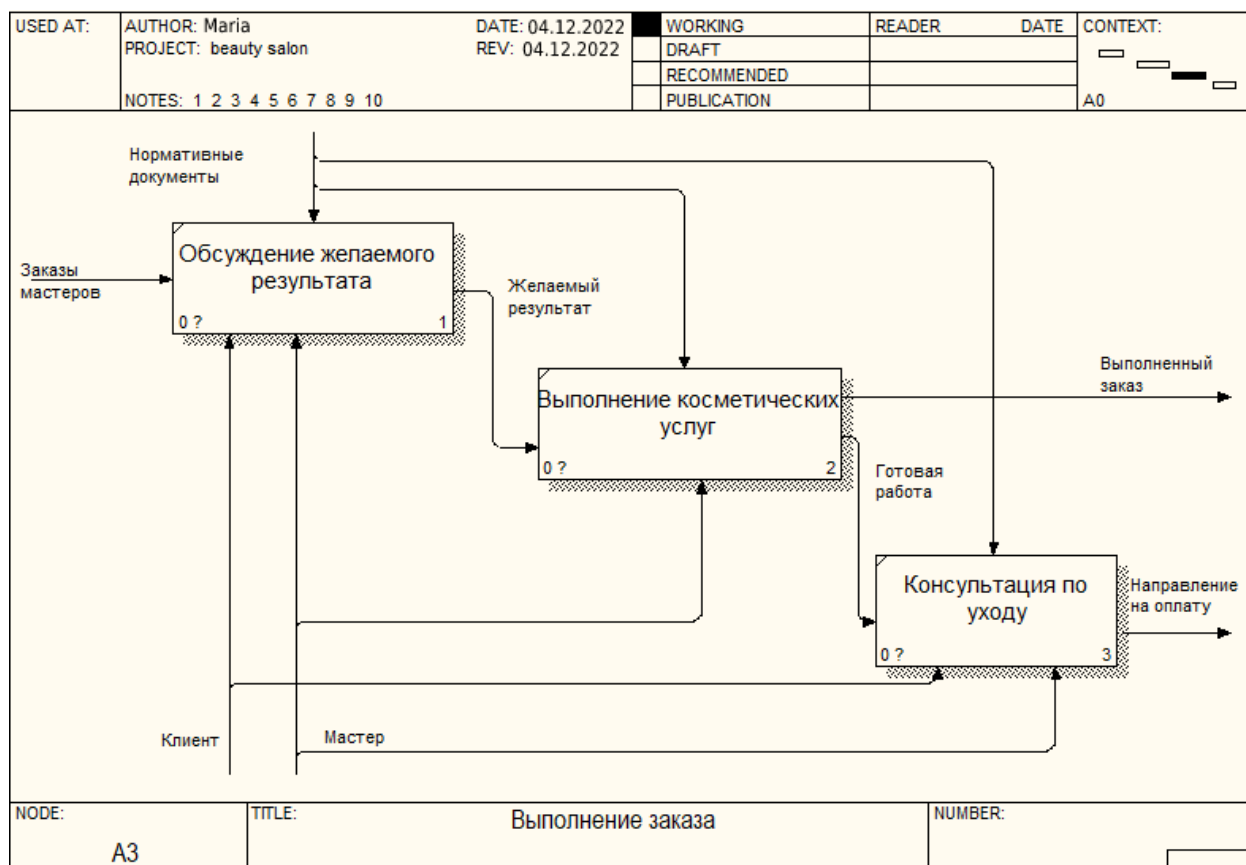


Рисунок 3.1.5 – Диаграмма декомпозиции процесса «Выполнение заказа»

На рисунке 3.1.6 отображена декомпозиция блока «Оплата», состоящая из трех блоков: «Расчет стоимости оказанных услуг», «Прием оплаты», «Формирование отчетной документации».

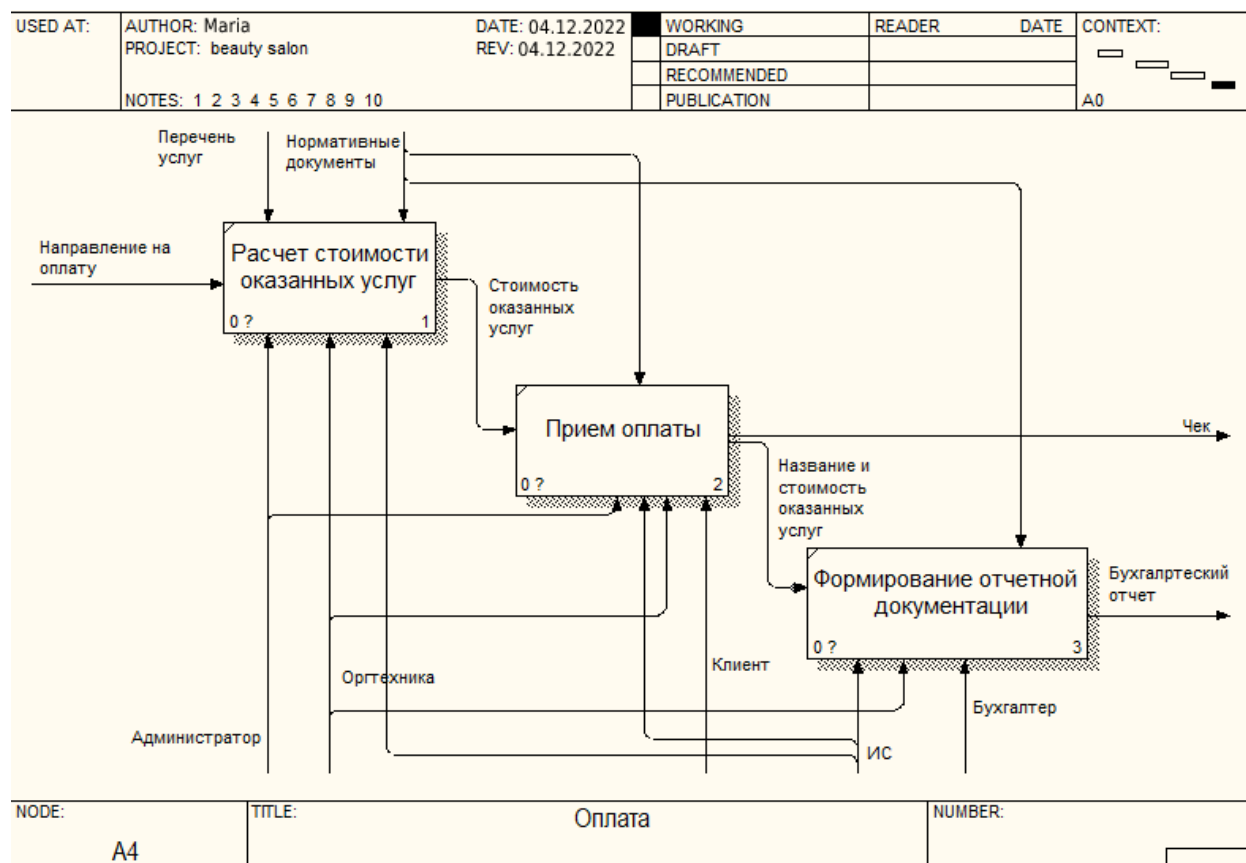


Рисунок 3.1.6 – Диаграмма декомпозиции блока «Оплата»

Таким образом, из представленных декомпозиций можно сделать вывод о том, какой сложно является работа косметического салона. Не возникает сомнений, что такая работа нуждается в автоматизации некоторых процессов для повышения качества обслуживания клиентов и оптимизации работы сотрудников.

4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЁ ОПИСАНИЕ

Информационная модель в данном курсовом проекте была построена с помощью средства MySQL Workbench 8.0 CE.

Сущностями данной модели являются appointments, users, client, admins, typeofservice, services, masters.

Сущность «users» содержит информацию о всех пользователях системы, а именно userID(уникальный номер пользователя), username(логин), password(пароль).

Сущность «admins» содержит информацию об администраторах, а именно adminID(уникальный номер администратора), status(статус администратора).

Сущность «client» содержит информацию о клиентах, а именно clientID(уникальный номер клиента), name(имя клиента), birthDate(дата рождения клиента), sex(пол клиента) .

Сущность «appointments» содержит информацию о записях, а именно clientID(уникальный номер клиента), appointments(название услуги), masterID(уникальный номер мастера), date(дата записи на услугу), time(время записи на услугу), price(стоимость услуги), ID(уникальный номер записи).

Сущность «typeofservice» содержит информацию о типах услуг, а именно typeServiceID(уникальный номер типа услуг), name(название типа услуг).

Сущность «services» содержит информацию об услугах, а именно serviceID(уникальный номер услуги), typeOfserviceID(уникальный номер типа услуг), name(название услуги), price(стоимость услуги).

Сущность «masters» содержит информацию о мастерах, а именно masterID(уникальный номер мастера), serviceTypeID(уникальный номер типа услуг), name(имя мастера), experience(опыт работы мастера).

На рисунке 4.1 представлена информационная модель базы данных.

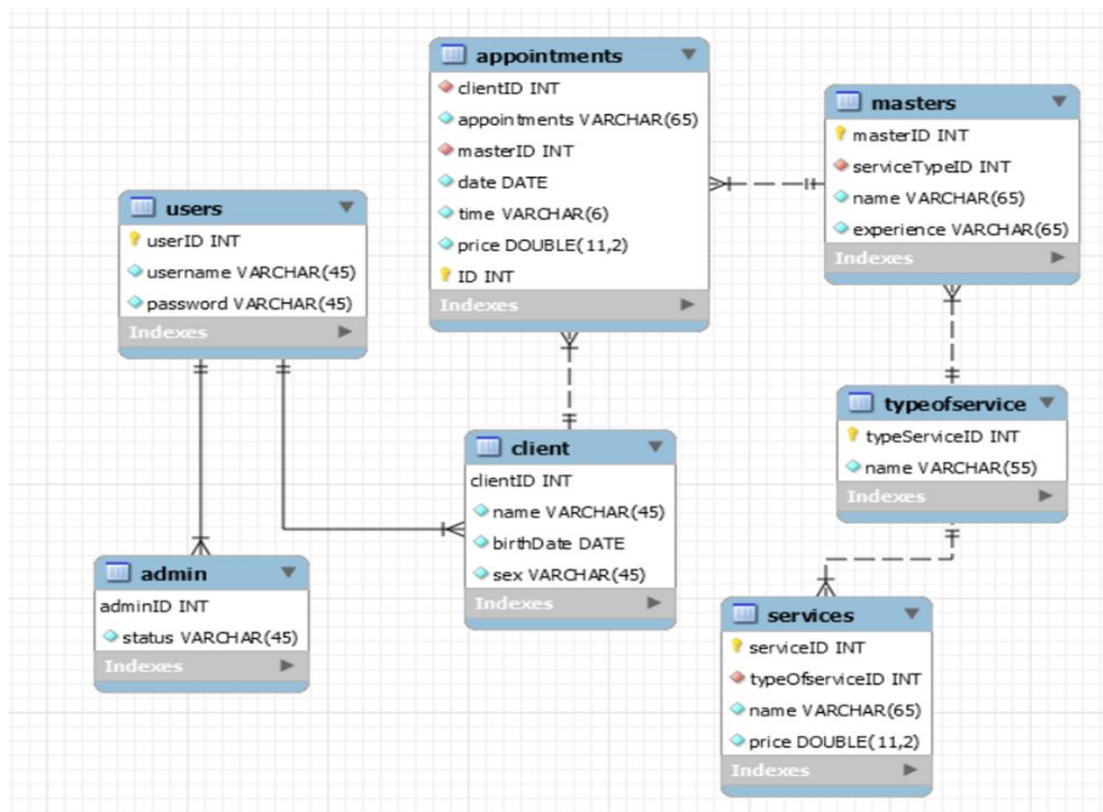


Рисунок 4.1 – Информационная модель базы данных

5 МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ

Для описания представления системы используется язык UML.

Язык UML представляет собой общецелевой язык визуального моделирования, который разработан для спецификации, визуализации, проектирования и документирования компонентов программного обеспечения, бизнес-процессов и других систем.

Язык UML может быть эффективно использован для построения концептуальных, логических и графических моделей сложных систем самого различного целевого назначения. Язык UML предлагает набор инструментальных средств, позволяющих проводить всесторонний анализ сложных ИС как с технической точки зрения, так и с точки зрения потребностей бизнеса. [6]

5.1 Диаграмма вариантов использования

Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация сервисов (функций), которые система предоставляет актеру. [7]

На рисунке 5.1.1 представлена диаграмма вариантов использования для администратора и пользователя.

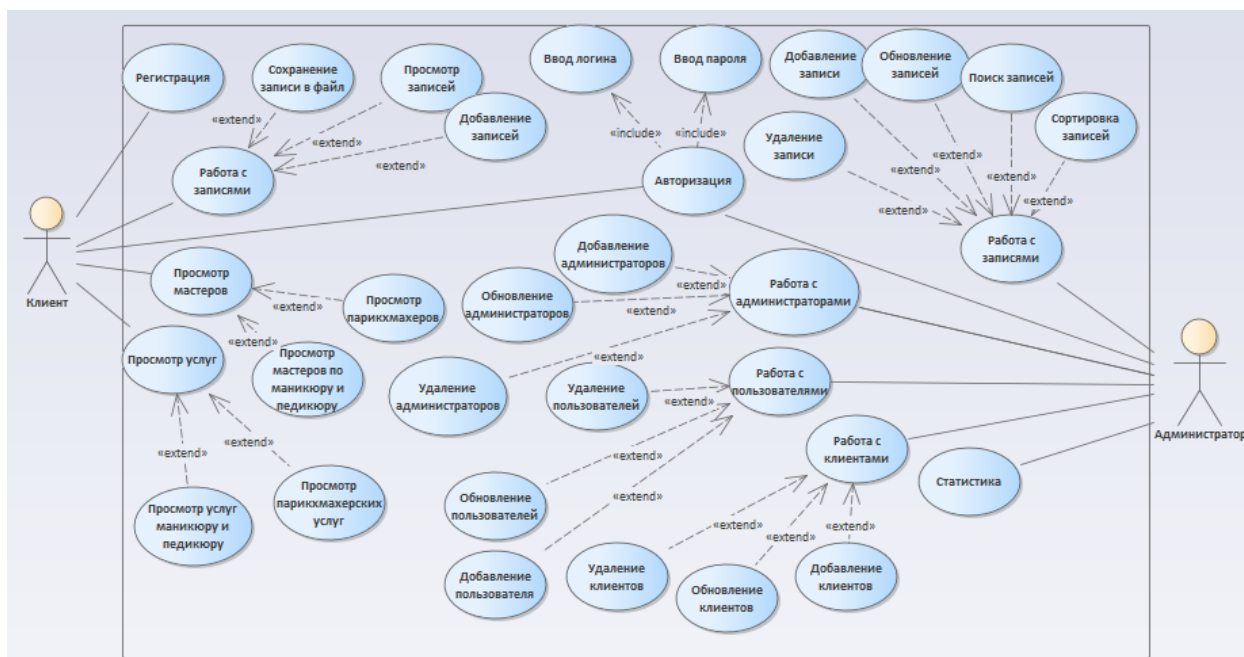


Рисунок 5.1.1 – Диаграмма вариантов использования

В данном курсовом проекте были определены основные действующие лица: администратор и пользователь.

Пользователь имеет доступ только к базовому функционалу программы. Он может просматривать информацию о мастерах и услугах. Что касается записей, то сотрудник может их просматривать, добавлять и сохранять в файл.

Администратор имеет доступ к полному функционалу программы, то есть он может просматривать, добавлять и удалять данные о пользователях, клиентах, администраторах и записях. Для записей есть возможность поиска по таблице записей. Кроме этого администратор имеет доступ к статистическим данным, а именно к диаграммам.

5.2 Диаграмма последовательностей

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

На рисунке 5.2.1 представлена диаграмма последовательности регистрации пользователя.

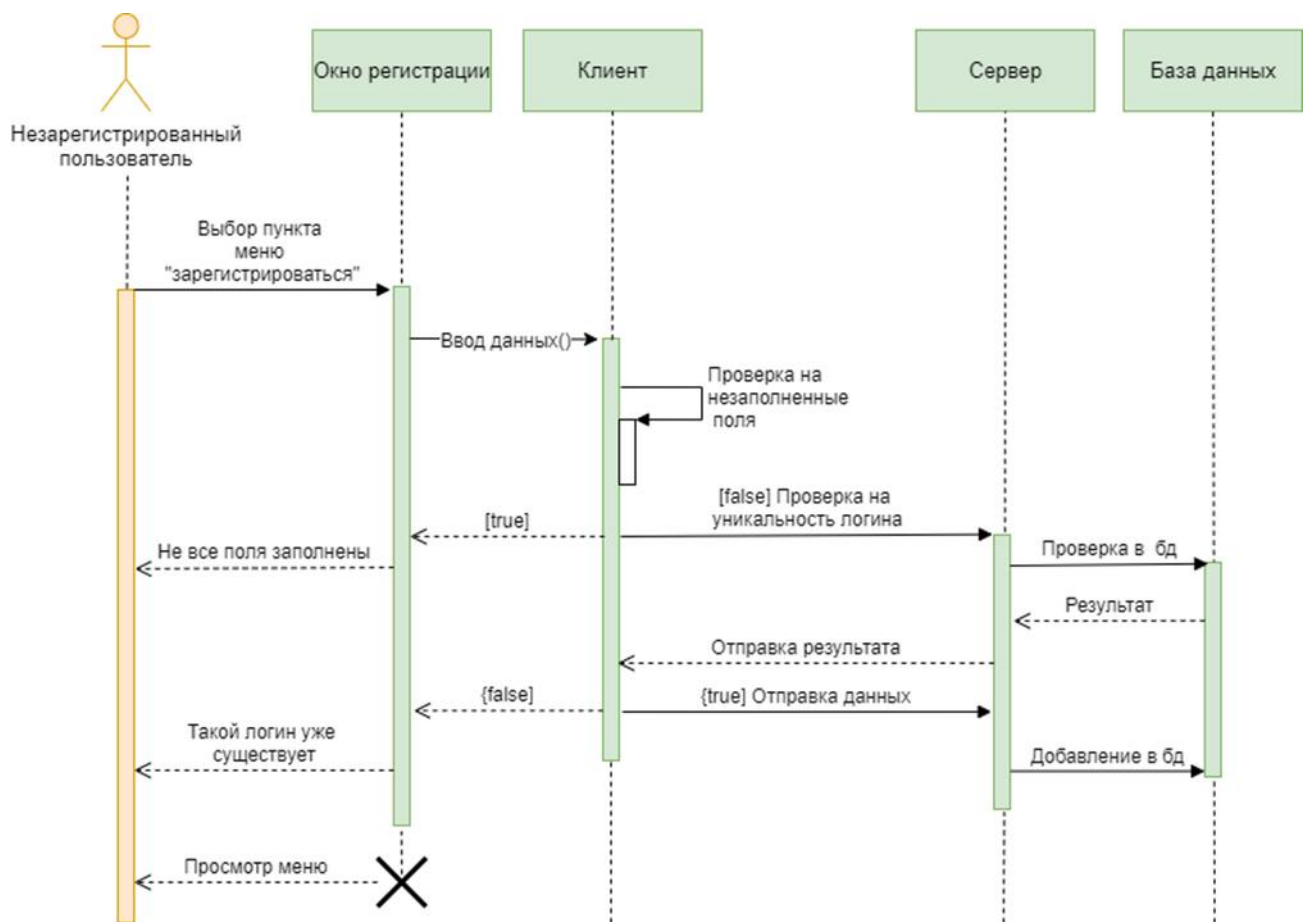


Рисунок 5.2.1 – Диаграмма последовательности регистрации пользователя

Для того, чтобы осуществить регистрацию пользователя необходимо выбрать пункт меню «зарегистрироваться».

Затем нужно ввести данные для регистрации в окно регистрации. После этого введенные данные проверяются на незаполненные поля на клиенте. Если результат true, то он отправляется в окно регистрации и пользователь получает сообщение о том, что не все поля заполнены. Если же результат false, то клиент посылает запрос на проверку уникальности логина.

Сервер в свою очередь отправляет запрос на проверку в базу данных. База данных отправляет результат на сервер, а сервер в свою очередь - на клиент. Если результат false, то он отправляет его в окно регистрации, а оно в свою очередь отправляет сообщение о существовании такого логина пользователю. Если же результат true, то клиент отправляет серверу данные. Сервер отправляет запрос в базу данных на их добавление. После окно регистрации закрывается и пользователь просматривает меню.

5.3 Диаграмма классов

Так как это приложение создано с помощью клиент-серверной архитектуры, для клиента и для сервера будут определены свои отдельные диаграммы классов. Диаграмма классов изображена на рисунках 5.3.1 – 5.3.4.

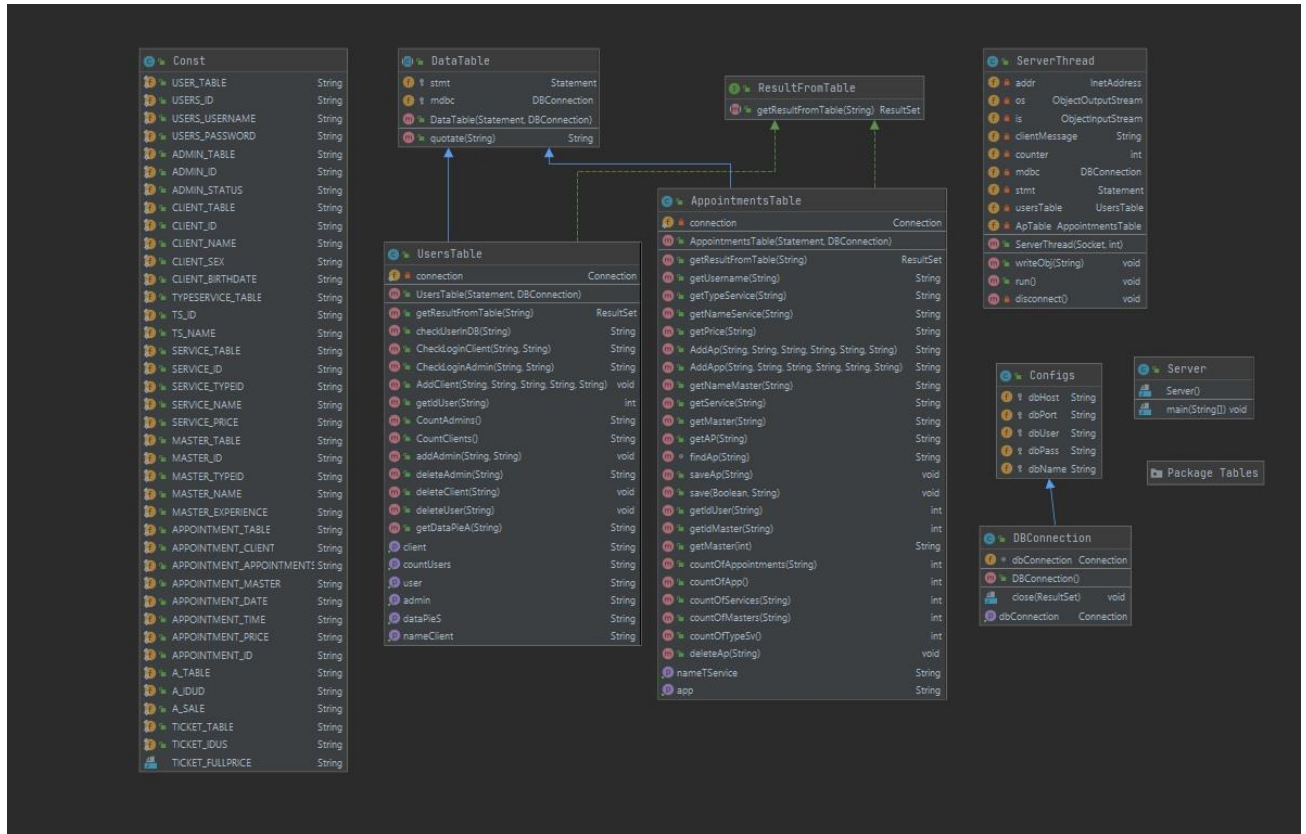


Рисунок 5.3.1 – Диаграмма классов пакета Server

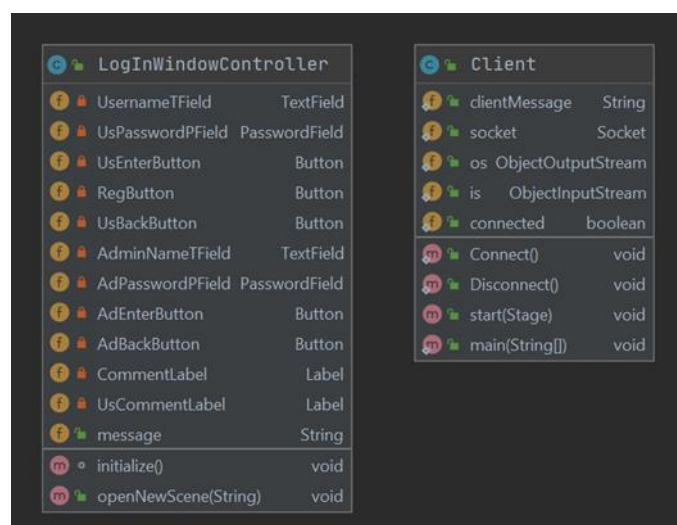


Рисунок 5.3.2 – Диаграмма классов пакета Client

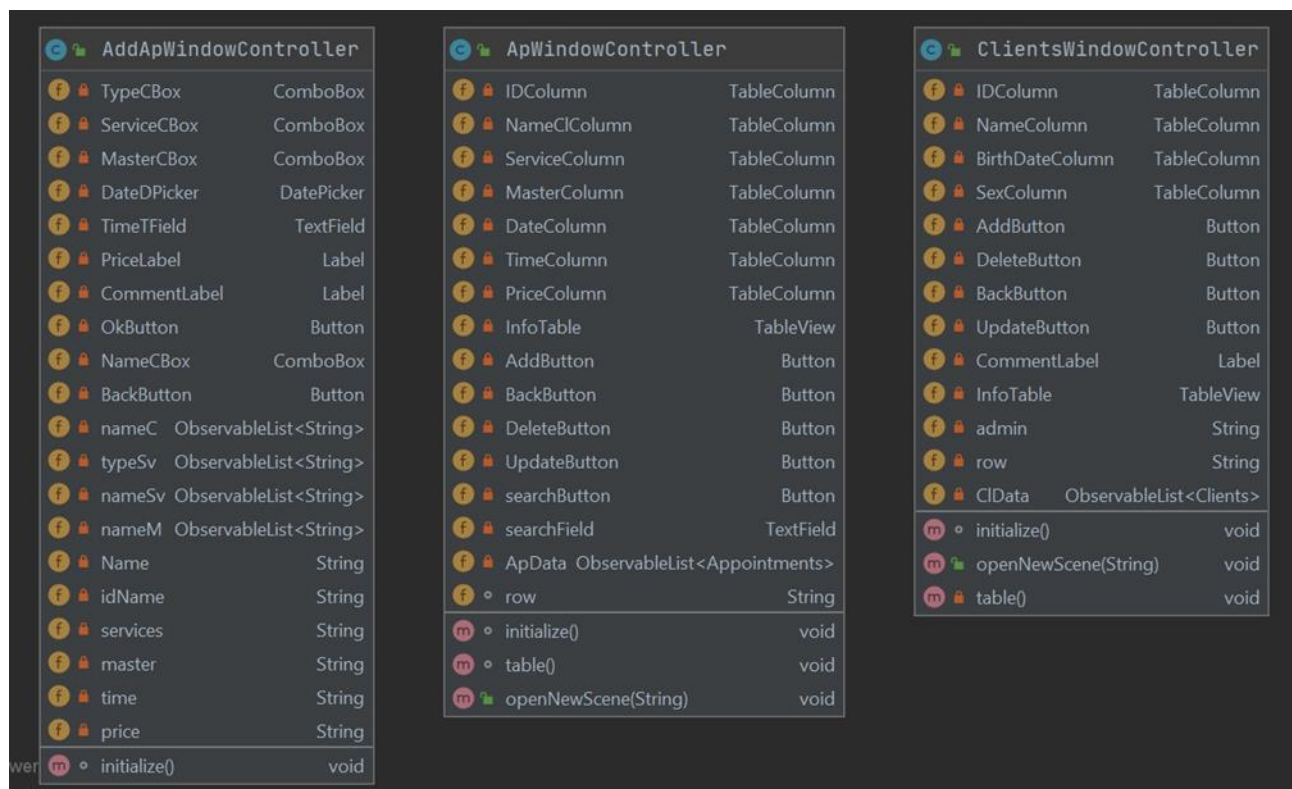


Рисунок 5.3.3 – Диаграмма классов пакета Admin

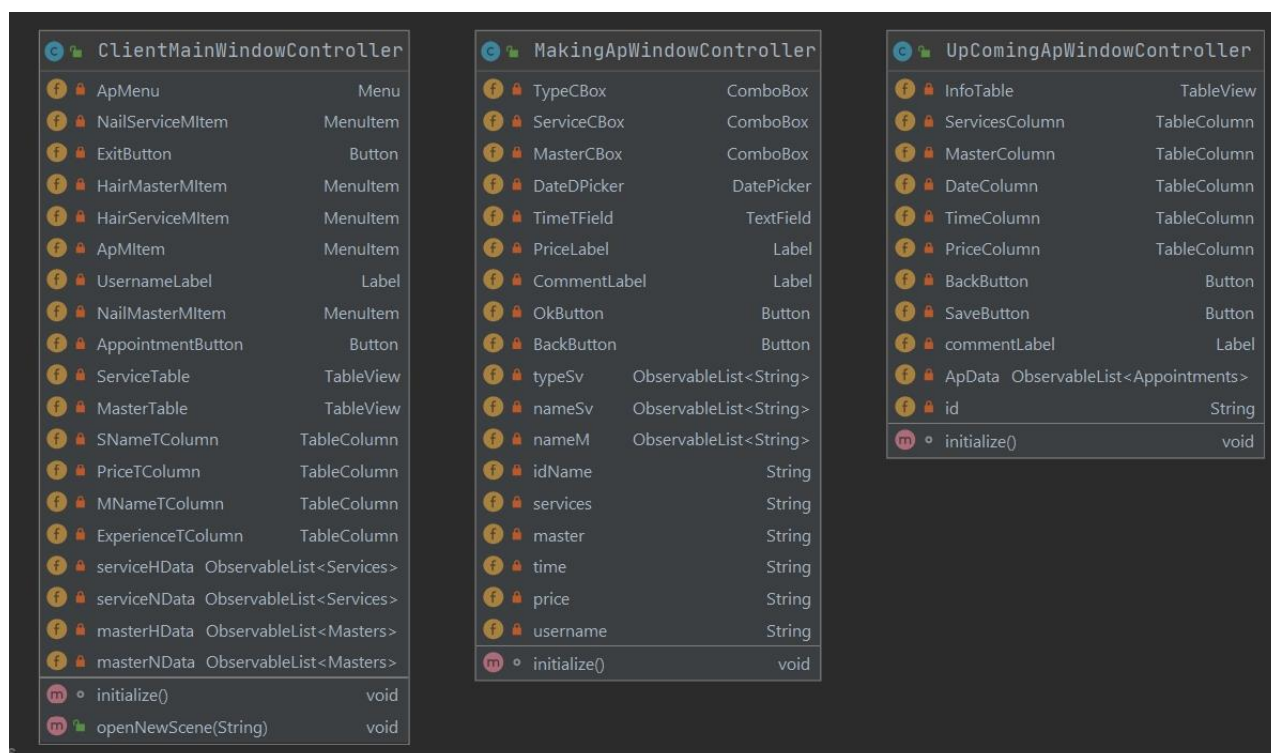


Рисунок 5.3.4 – Диаграмма классов пакета User

5.4 Диаграмма компонентов

Диаграмма компонентов описывает особенности физического представления системы. Она позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный и исполняемый код.

На рисунке 5.4.1 представлена диаграмма компонентов системы.

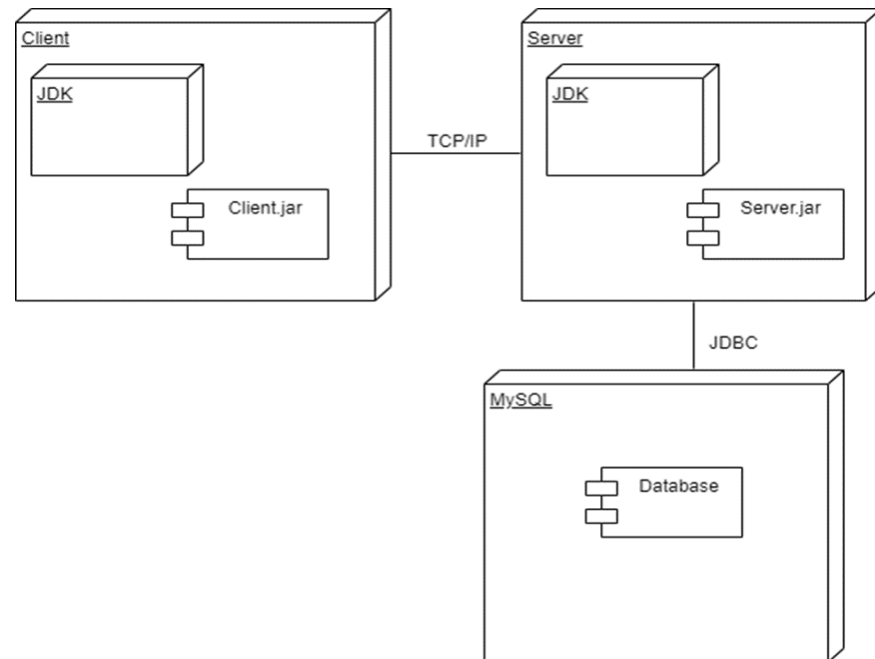


Рисунок 5.4.1 – Диаграмма компонентов системы

5.5 Диаграмма развертывания

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения. При этом представляются только компоненты-экземпляры программы, являющиеся исполняемыми файлами или динамическими библиотеками.

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними.

На рисунке 5.5.1 представлена диаграмма развертывания.

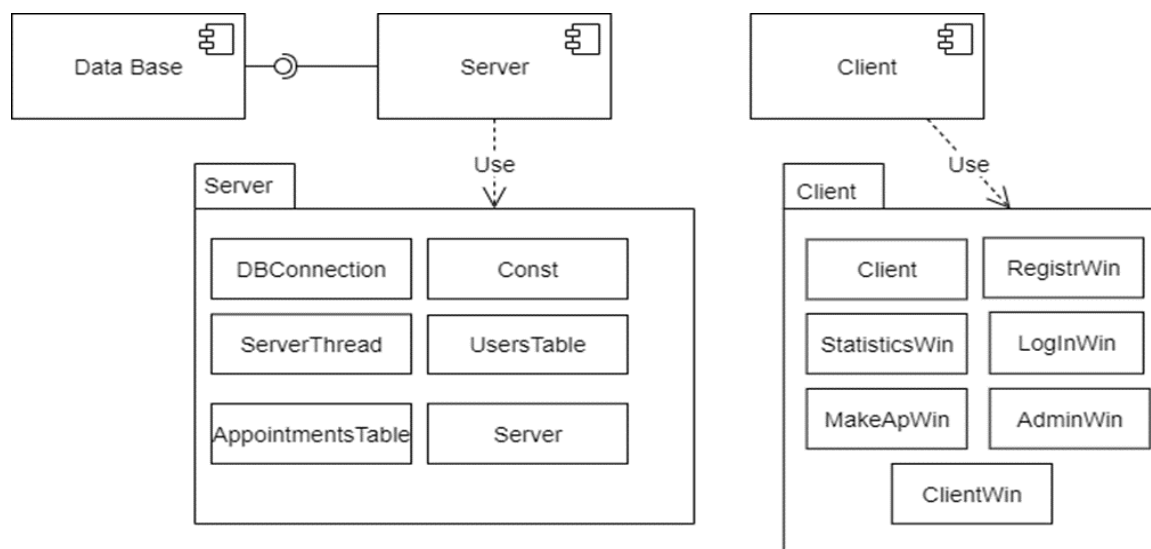


Рисунок 5.5.1 – Диаграмма развертывания

6 ОПИСАНИЕ АЛГОРИТМОВ РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СИСТЕМЫ

Курсовой проект разработан с помощью архитектуры клиент-сервер с графическим интерфейсом. В программе происходит взаимодействие с клиентом через клиентскую часть, а запросы клиента выполняет серверная часть программы.

6.1 Алгоритм авторизации

Данный алгоритм позволяет проверить введенные пользователем данные для авторизации его в системе.

Схема алгоритма авторизации пользователя представлена на рисунке 6.1.

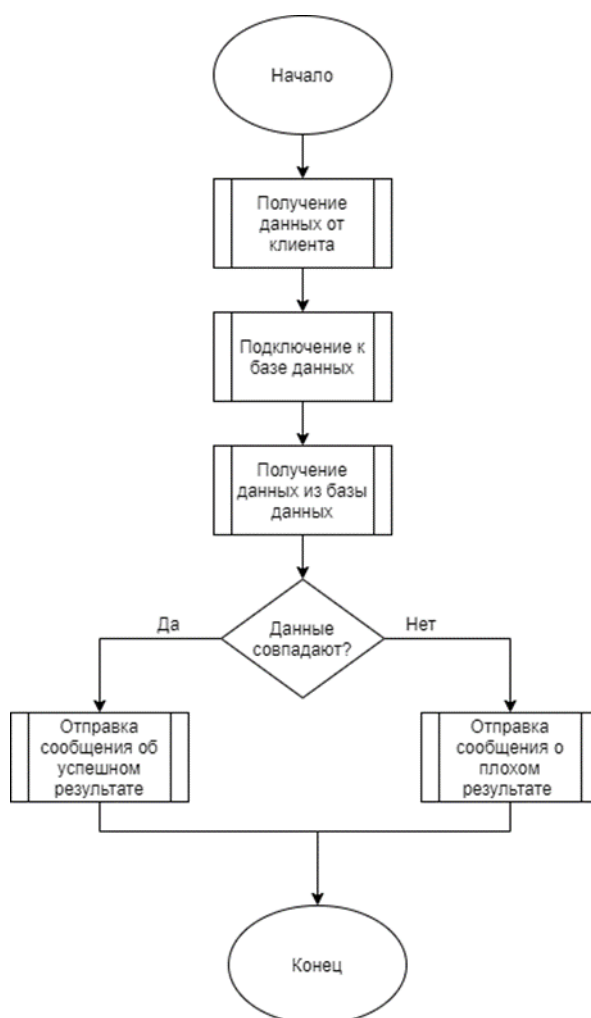


Рисунок 6.6.1 – Алгоритм авторизации

6.2 Алгоритм получения записей

Данный алгоритм позволяет получить все записи, находящиеся в базе данных, для дальнейшей работы с ними.

Схема алгоритма получения записей представлена на рисунке 6.2.2

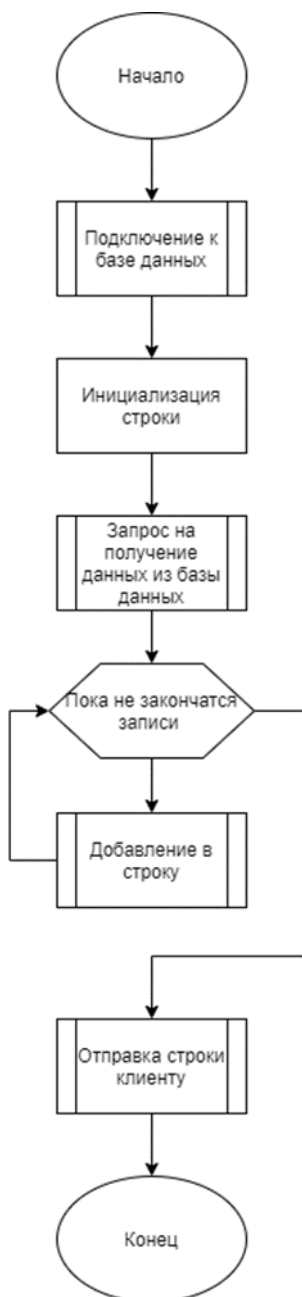


Рисунок 6.6.2 – Алгоритм получения записей

7 РУКОВОДСТВО РАЗВЕРТЫВАНИЯ

Для возможности использования системы требуется наличие на компьютере 32х разрядной или 64х разрядной операционной системы Windows7 и выше, необходимо чтоб на компьютере было установлено java JDK и java JRE для запуска java приложения.

Для корректной работы программы необходимо подключение к базе данных. Подключится к базе данных можно используя систему управления базами данных MySQL Workbench 8.0 CE. В приложении Б приведён листинг скрипта для подключения к базе данных на устройстве. Перед запуском скрипта необходимо выполнить настройку и убедиться, что система управления работает корректно. После этого пользователь может подключиться к базе данных.

Для запуска приложения необходимо последовательно запустить server.jar, а затем client.jar.

В результате описанных выше действий пользователь может корректно запустить данное программное приложение на любом удовлетворяющем требованиям устройстве.

8 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

При запуске программы сразу появляется приветствующее окно, на котором расположена кнопка «Войти». Приветствующее окно представлено на рисунке 8.1.

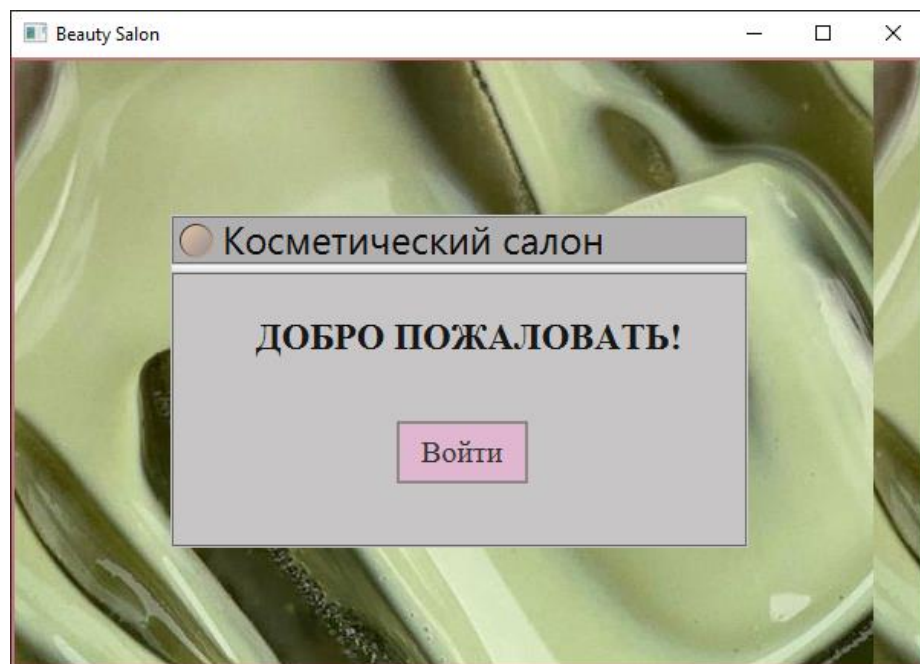


Рисунок 8.1 – Приветствующее окно

При нажатии на кнопку «Войти» появляется окно для авторизации, на котором расположены поля для ввода логина и пароля, а также кнопки «Войти», «Зарегистрироваться» и «Назад» на панели Пользователя. Окно авторизации с панелью пользователя представлено на рисунке 8.2.

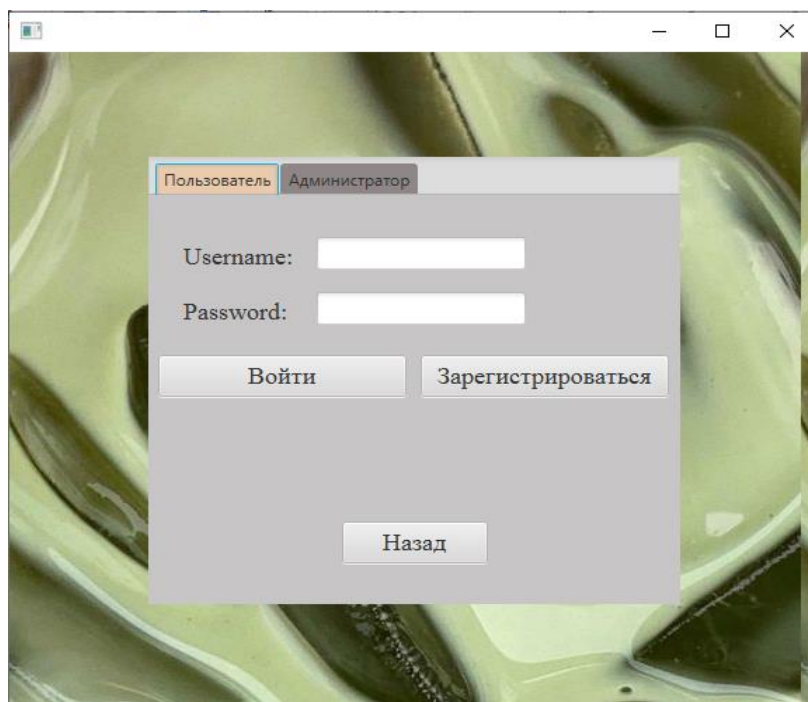


Рисунок 8.2 – Окно авторизации для пользователя

При нажатии на кнопку «Зарегистрироваться» открывается окно регистрации. При нажатии на кнопку «Ок» проверяется заполнены ли все поля, а также уникальность логина. При нажатии на кнопку «Назад» возвращается окно авторизации. Окно регистрации представлено на рисунке 8.3.

The image shows a screenshot of a web application window titled "Регистрация" (Registration). The window has a light gray background with a decorative floral pattern on the sides. It contains several input fields and checkboxes for user registration. The fields are labeled "Имя:" (Name), "Дата рождения:" (Date of birth), "Пол:" (Gender), "Username:", and "Password:". The "Имя:" field contains the text "Алина". The "Дата рождения:" field contains "08.04.1999" and has a small calendar icon to its right. The "Пол:" field has two radio buttons: "Мужской" (Male) and "Женский" (Female), with the "Женский" option selected. The "Username:" field contains the text "alina". The "Password:" field contains five dots, indicating a masked password. At the bottom of the form, there are two buttons: "ОК" (OK) and "Назад" (Back).

Имя:	Алина
Дата рождения:	08.04.1999
Пол:	<input type="radio"/> Мужской <input checked="" type="radio"/> Женский
Username:	alina
Password:

ОК Назад

Рисунок 8.3 – Окно регистрации пользователя

После авторизации пользователя открывается окно с меню пользователя. Там можно просмотреть информацию о своих записях, услугах, мастерах, записаться на услугу. Логин пользователя указан в нижней левой части экрана. Окно меню пользователя представлено на рисунке 8.4.

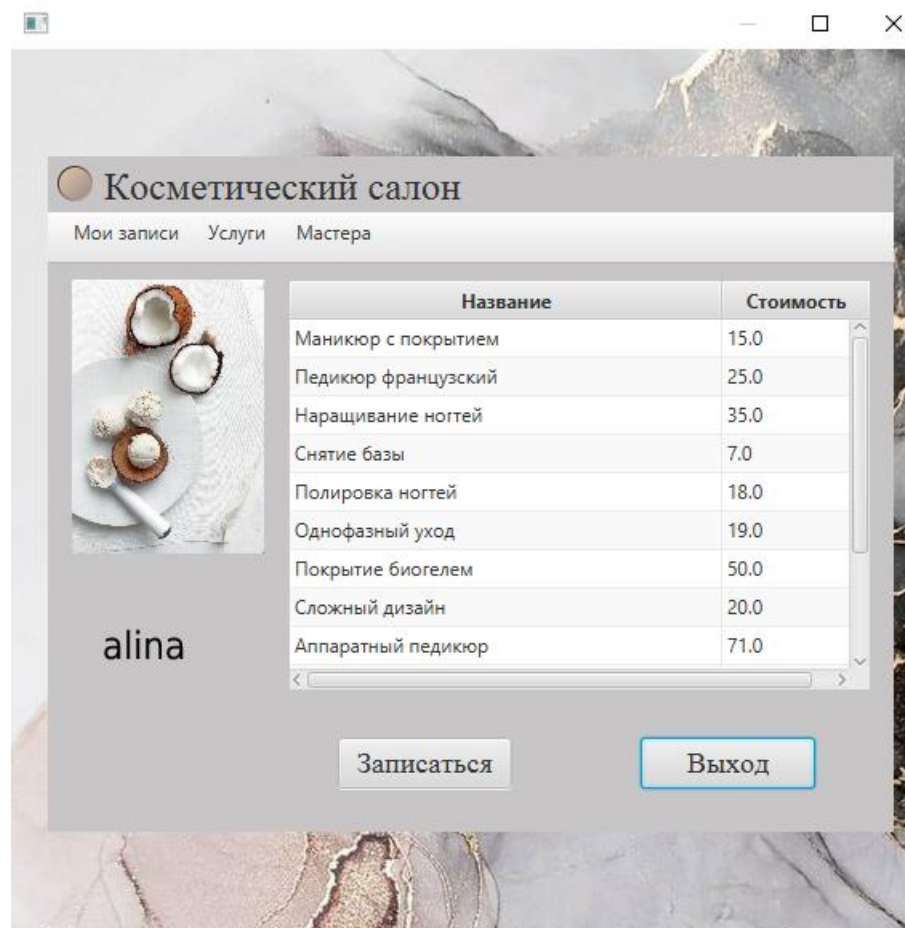


Рисунок 8.4 – Окно пользователя

Если нажать кнопку «Записаться», то откроется окно добавления записи. Там можно выбрать тип услуг, саму услугу и мастера в соответствии с выбранным типом, ввести дату и время. Стоимость автоматически появится в соответствии с выбранной услугой. Если выбранные дата и время уже прошли, то появится соответствующее сообщение. Окно добавления записи представлено на рисунке 8.5.

Запись

Тип Услуги: Маникюр и педикюр

Услуги: Полировка ногтей

Мастер: Катя

Дата: 08.12.2022

Время: 14:30

Стоимость: 18.0

ОК Назад

Рисунок 8.5 – Окно добавления записи

Если нажать кнопку «Записи» в меню моих записей, то откроется окно с информацией о записях пользователя, где можно сохранить их в файл и появится сообщение об успешном сохранении. Окно моих записей представлено на рисунке 8.6.

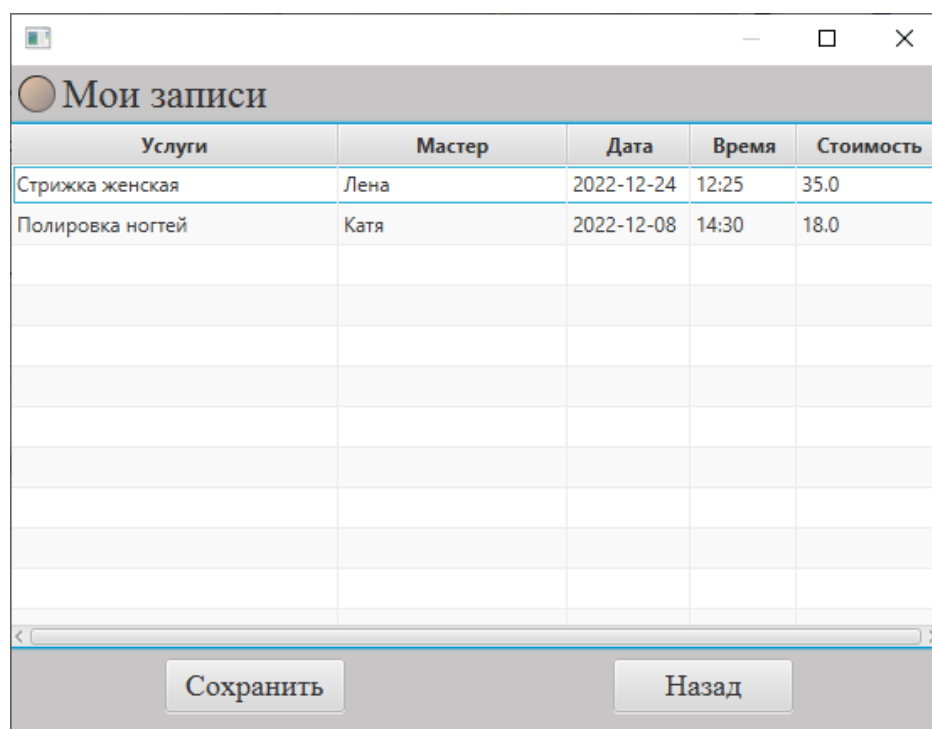


Рисунок 8.6 – Окно моих записей

После авторизации администратора откроется окно с меню администратора. В нем можно просмотреть, добавить и удалить информацию о пользователях, администраторах, клиентах, записях, а также просмотреть статистику и осуществить поиск записей. Окно меню администратора представлено на рисунке 8.7.

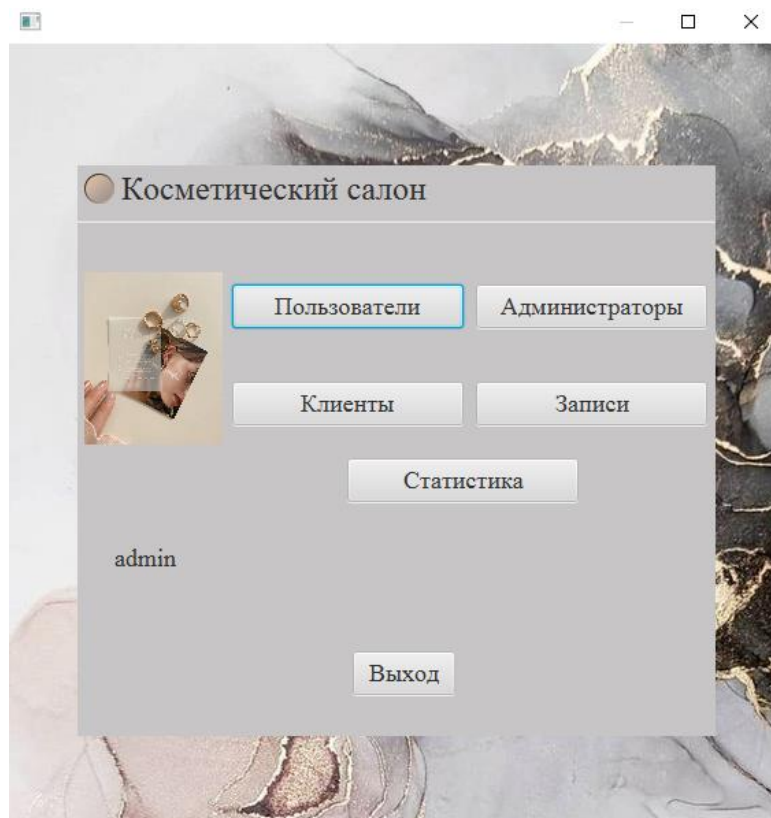


Рисунок 8.7 – Окно меню администратора

Окно пользователя представлено на рисунке 8.8. Окно администратора представлено на рисунке 8.9. Окно клиента представлено на рисунке 8.10. Есть возможность добавления, удаления, а также обновления записей.

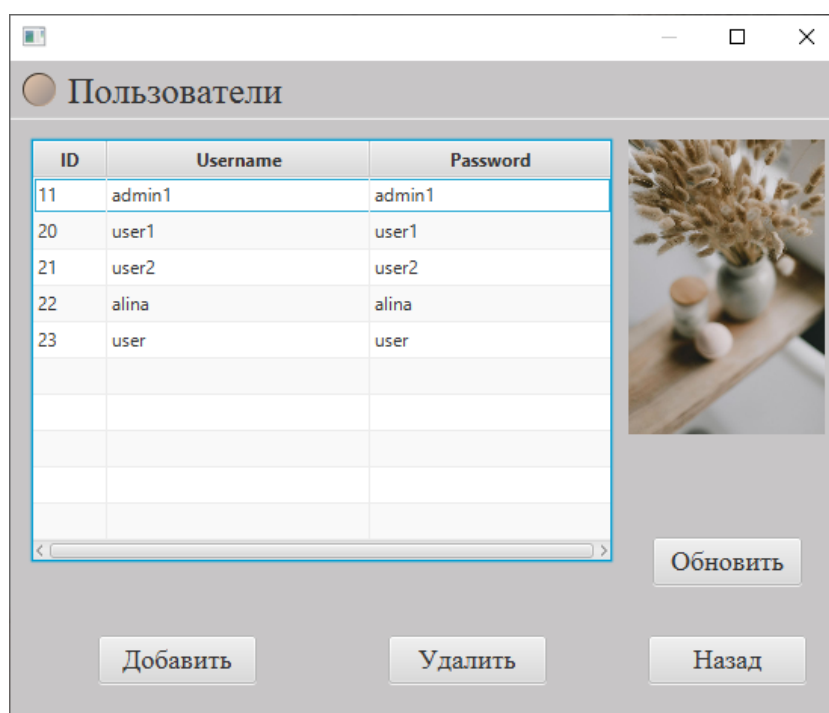


Рисунок 8.8 – Окно пользователя

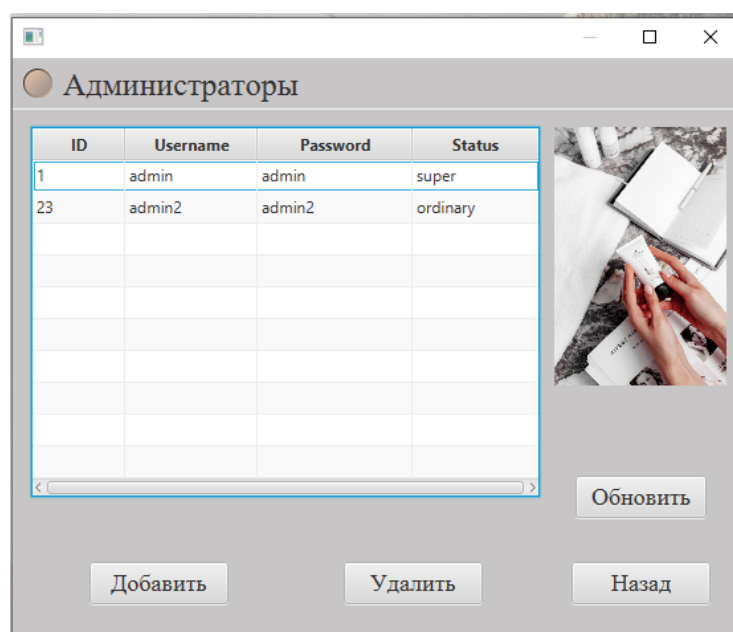


Рисунок 8.9 – Окно администраторов

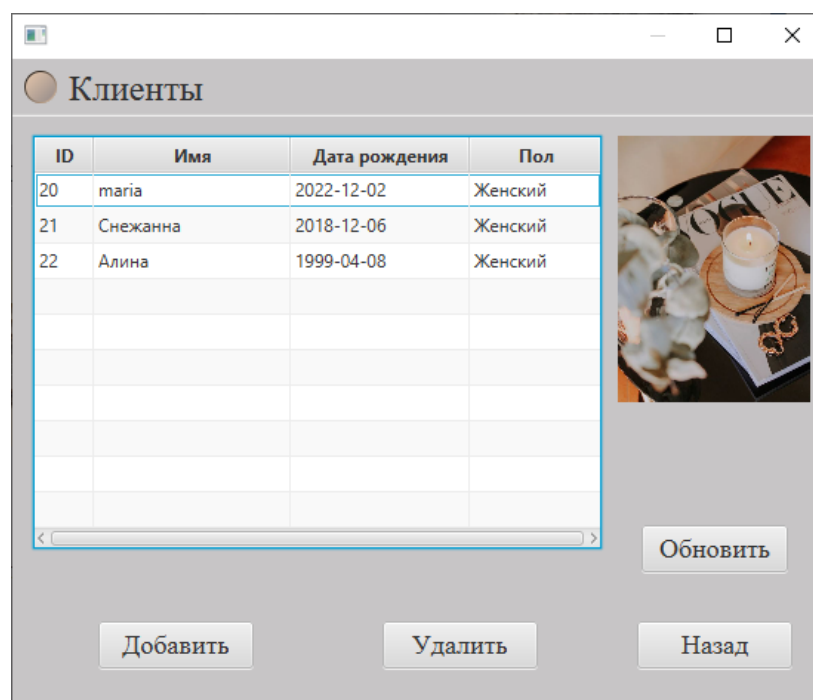


Рисунок 8.10 – Окно клиентов

В окне записей можно совершить поиск. Администратор вводит слово и отображается все записи, в которых оно найдено. А также есть возможность сортировки данных, необходимо нажать на колонку, которую хотите отсортировать. Окно записей представлено на рисунке 8.11- 8.13.

Записи

ID	Имя Клиента	Услуги	Мастер	Дата	Время	Стоимость
1	maria	Маникюр с покрытием	Катя	2022-12-15	15:00	15.0
2	Алина	Стрижка женская	Лена	2022-12-24	12:25	35.0
3	Алина	Полировка ногтей	Катя	2022-12-08	14:30	18.0

Добавить поиск Удалить Назад

Обновить

Рисунок 8.11 – Окно записей

Записи

ID	Имя Клиента	Услуги	Мастер	Дата	Время	Стоимость ▾
2	Алина	Стрижка женская	Лена	2022-12-24	12:25	35.0
3	Алина	Полировка ногтей	Катя	2022-12-08	14:30	18.0
1	maria	Маникюр с покрытием	Катя	2022-12-15	15:00	15.0

Добавить поиск Удалить Назад

Обновить

Рисунок 8.12 – Окно сортировки записей

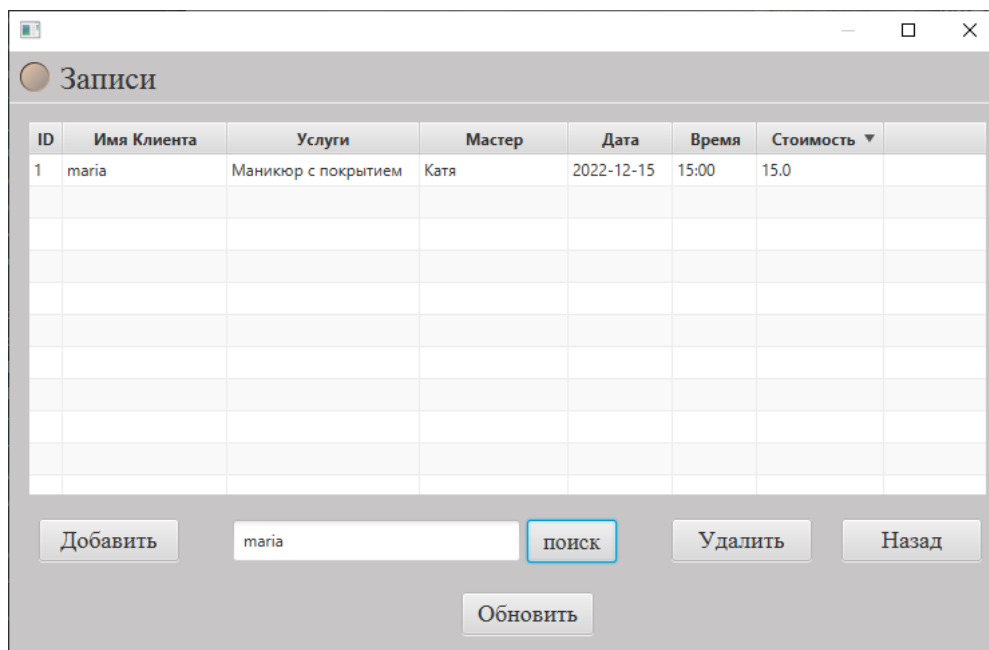


Рисунок 8.13 – Окно поиска записей

Окно со статистикой представлено на рисунке 8.14 и там отображены 2 диаграммы. На одной показано соотношение мужчин и женщин, а на другой соотношение мужчин или женщин в соответствии с выбранным параметром по возрасту.

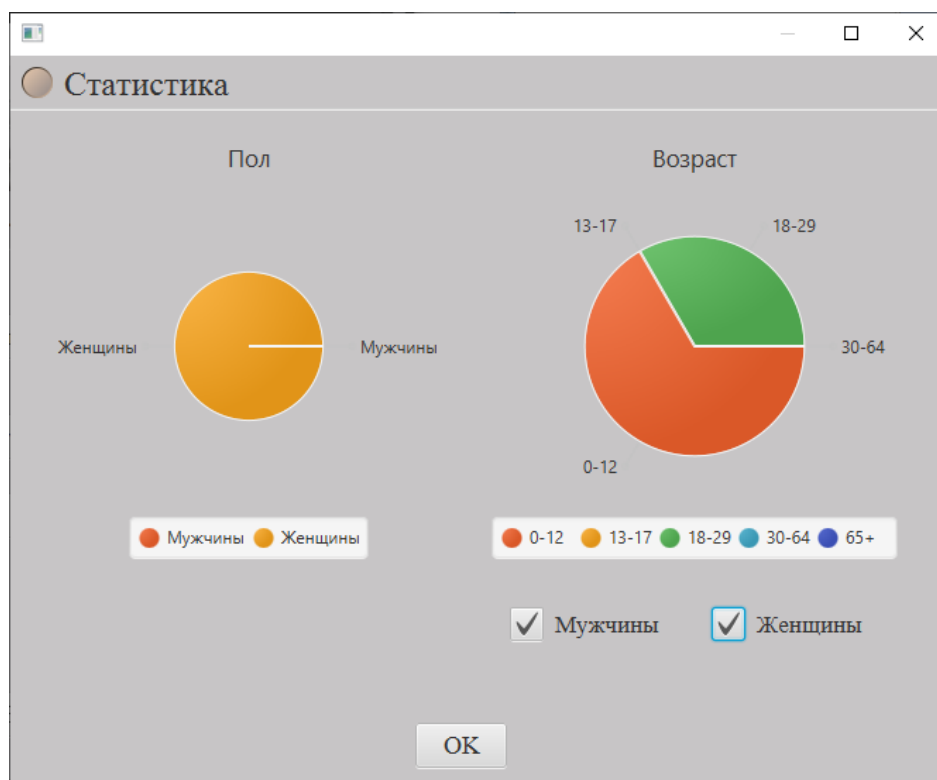


Рисунок 8.14 – Окно статистики

9 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ

В процессе разработки программного продукта была реализована логика пользователя с предусмотренными вероятностными ошибками ввода, с которыми может столкнуться пользователь. Поэтому в программе обеспечены необходимые проверки.

При нажатии на кнопку «Войти», если хотя бы одно поле окажется не заполненным, то появится сообщение о том, что все поля не заполнены. Окно с оповещением о незаполненных полях на панели пользователя представлено на рисунке 9.1.

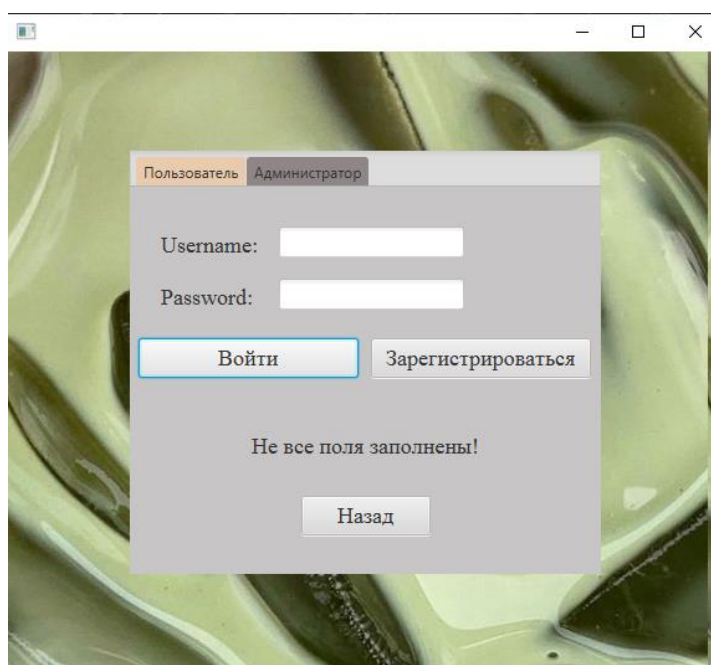


Рисунок 9.1 – Окно входа под пользователем с оповещением об ошибке

Если пользователь введет неверный пароль или логин, то появится оповещение, представлено на рисунке 9.2.

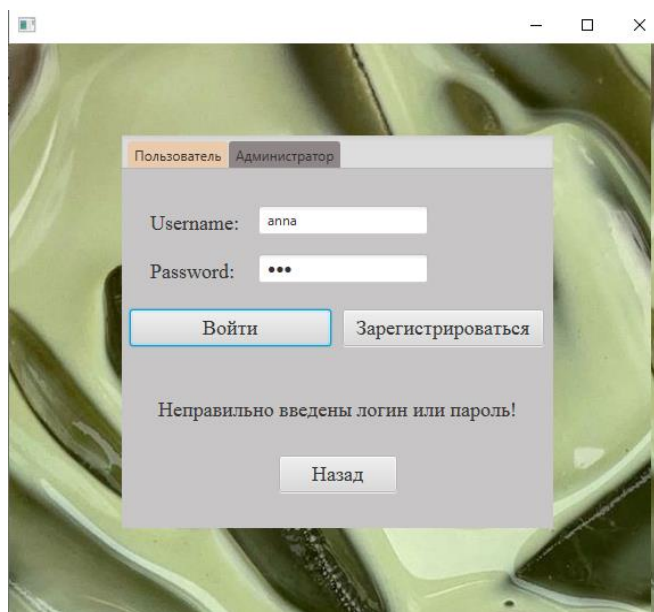


Рисунок 9.2 – Окно с оповещением о некорректности ввода на панели пользователя

При нажатии на кнопку «Зарегистрироваться» открывается окно регистрации. При нажатии на кнопку «Ок» проверяется заполнены ли все поля, а также уникальность логина. Предоставлено на рисунке 9.3 и 9.4.

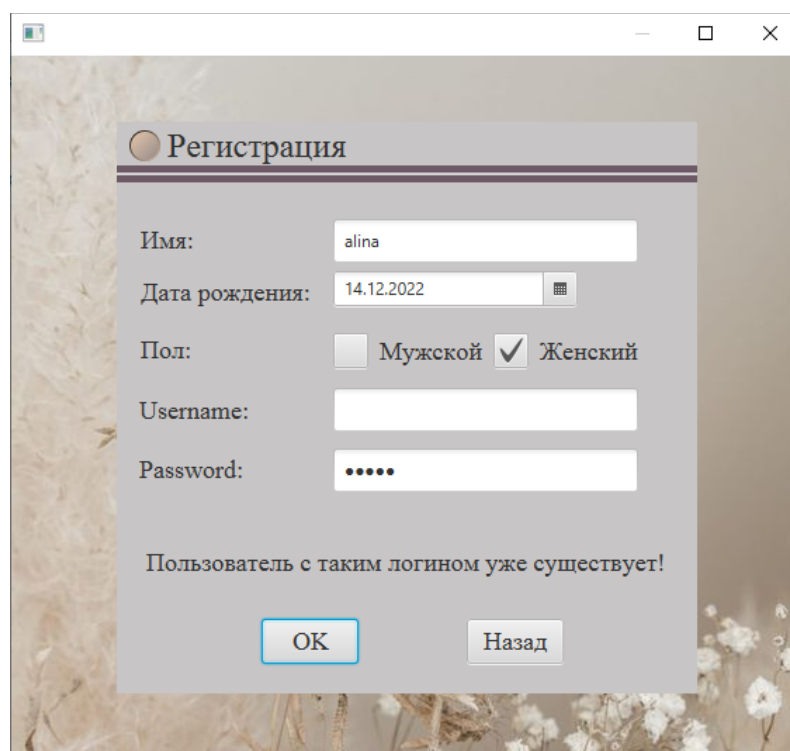
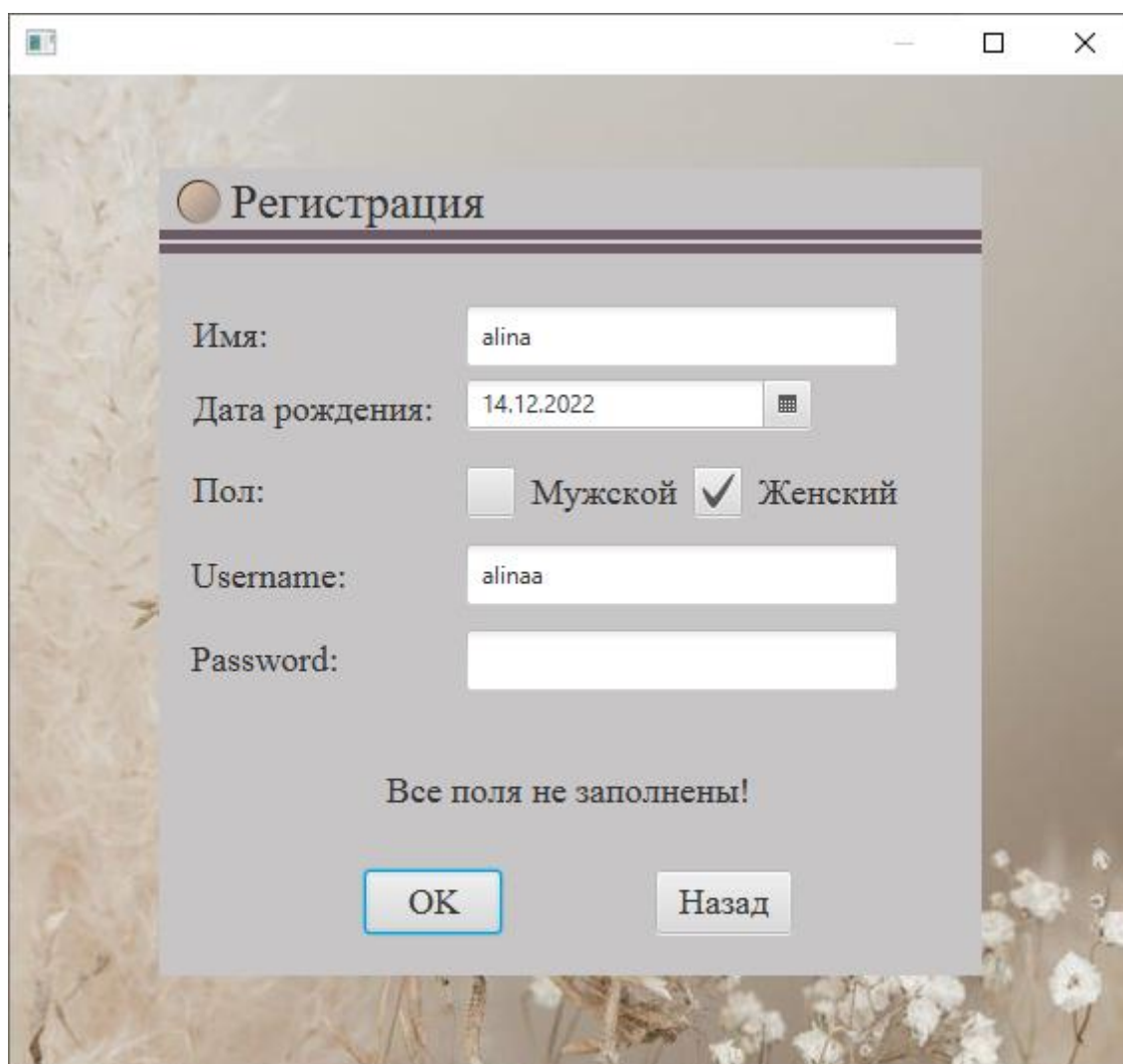


Рисунок 9.3 – Окно с оповещением о некорректности логина пользователя



The image shows a registration window titled "Регистрация" (Registration). It contains several input fields: "Имя:" (Name) with the value "alina", "Дата рождения:" (Date of birth) with the value "14.12.2022", "Пол:" (Gender) with radio buttons for "Мужской" (Male) and "Женский" (Female), where "Женский" is selected. Below these are "Username:" with the value "alinaa" and "Password:" which is empty. At the bottom, a message states "Все поля не заполнены!" (All fields are not filled!). There are two buttons: "ОК" (OK) and "Назад" (Back).

Регистрация

Имя: alina

Дата рождения: 14.12.2022

Пол: ☐ Мужской ☒ Женский

Username: alinaa

Password:

Все поля не заполнены!

ОК Назад

Рисунок 9.4 – Окно регистрации с оповещением об ошибке

Если нажать кнопку «Записаться», то откроется окно добавления записи. Если выбранные дата и время уже прошли, то появится соответствующее сообщение. Также необходимо заполнить все поля. Предоставлены оповещения на рисунках 9.5, 9.6.

Запись

Тип Услуги: Парикмахерская

Услуги: Мытье волос

Мастер: Елена

Дата: 01.12.2021

Время: 11:00

Стоимость: 2.5

Этот день уже прошел!

OK Назад

Рисунок 9.5 – Окно добавления записи с оповещением об ошибке

Запись

Тип Услуги: Парикмахерская

Услуги: Мытье волос

Мастер: Елена

Дата: 19.12.2021

Время:

Стоимость: 2.5

Не все поля заполнены!

OK Назад

Рисунок 9.6 – Окно добавления записи с оповещением об ошибке

Аналогичные проверки реализованы и у администратора, а также администратор не может удалить сам себя, это продемонстрировано на

рисунке 9.7.

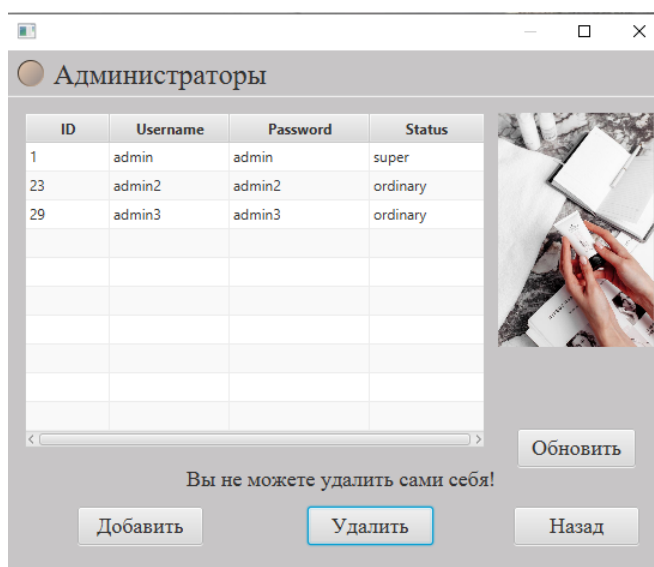


Рисунок 9.7 – Окно удаления администратора с оповещением об ошибке

ЗАКЛЮЧЕНИЕ

В рамках курсового проекта по дисциплине было разработано приложение с клиент-серверной архитектурой для автоматизации работы косметического салона.

Для наглядности предметной области была построена информационная модель IDEF0 и диаграммы UML. Также были пройдены различные этапы проектирования автоматизированной системы работы косметического салона.

Для её достижения был проведен анализ предметной области, функциональное и визуальное моделирование, разработана информационная модель (база данных) системы, разработаны различные функции и графический пользовательский интерфейс программы, программа была протестирована.

В процессе разработки программного продукта была реализована логика пользователя с предусмотренными вероятностными ошибками ввода, с которыми может столкнуться пользователь. Поэтому в программе обеспечены необходимые проверки.

Таким образом цель курсового проекта достигнута, разработанная система удобна в использовании и соответствует поставленным задачам.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Арнольд, К., Гослинг, Дж., Холмс, Д. Язык программирования Java. – 3-е изд. – М. : Вильямс, 2001. — 624 с.
- [2] Эккель, Б. Философия Java. – 4-е изд. –СПб. : Питер, 2011. –640 с.
- [3] Блох, Д. Java. Эффективное программирование. –М. : Лори, 2002. – 224 с.
- [4] Макконнелл, С. Совершенный код. –СПб. : Питер, 2005. –896 с.
- [5] Хорстманн, К. С., Корнелл, Г. Библиотека профессионала. Java 2 : Том 1. Основы. –8-е изд. –М. : Вильямс, 2013. –816 с.
- [6] Шилд Г. «Java. Полное руководство»
- [7] Хорстманн, К. С., Корнелл, Г. Библиотека профессионала. Java 2. : Том 2. Тонкости программирования. –8-е изд. –М. : Вильямс, 2012. –992 с.
- [8] Тейт, Б. Горький вкус Java – СПб. : Питер, 2003. – 334 с.
- [9] Буч, Г., Рамбо, Дж., Джекобсон, А. Язык UML. Руководство пользователя. – М. : ДМК, 2000. – 432 с.
- [10] Шилд Г. «Java. Руководство для начинающих»

ПРИЛОЖЕНИЕ А
(обязательное)
ЛИСТИНГ ПРОГРАММЫ

```
class Server
package Server;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.SQLException;

public class Server {
    public Server (){};

    public static void main(String[] args) {
        try {
            int counter = 0;
            ServerSocket serverSocket = new
ServerSocket(8081);
            System.out.println("Server started...");

            while (true) {
                Socket socket = serverSocket.accept();

                System.out.println(socket.getInetAddress().getHostName() + "
connected");

                ++counter;
                System.out.println("Client №" + counter + "is
connected");
            }
        }
    }
}
```

```

        ServerThread thread = null;
        try {
            thread = new ServerThread(socket,
counter);

            } catch (SQLException e) {
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
            thread.start();
        }
    } catch (IOException var5) {
        System.err.println(var5);
    }
}

```

class DBConnection

```

package Server;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.ResultSet;

public class DBConnection extends Configs{
    Connection dbConnection;
    public DBConnection(){};

    public Connection getDbConnection() throws
ClassNotFoundException, SQLException
    {
        String
connectionString="jdbc:mysql://localhost:3306/hairdresser"+
        "?verifyServerCertificate=false"+
        "&useSSL=false"+
        "&requireSSL=false"+
        "&useLegacyDatetimeCode=false"+
        "&"+
        "&serverTimezone=UTC";
        Class.forName("com.mysql.cj.jdbc.Driver");
        dbConnection =
DriverManager.getConnection(connectionString,
        dbUser, dbPass);
        return dbConnection;
    }
}

```

```

    }

    public void close(ResultSet rs) {
        if (rs != null) {
            try
            {
                rs.close();
            }
            catch (Exception e3)
            {
                e3.printStackTrace();
            }
        }
    }
}

class ServerThread
package Server;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.InetAddress;
import java.net.Socket;
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;

public class ServerThread extends Thread {
    private InetAddress addr;
    private ObjectOutputStream os;
    private ObjectInputStream is;
    private String clientMessage;
    private int counter;
    private DBConnection mdbc;
    private Statement stmt;
    private UsersTable usersTable;
    private AppointmentsTable ApTable;

    public ServerThread(Socket s, int counter) throws
IOException, SQLException, ClassNotFoundException {
        this.counter = counter;
        this.os = new
ObjectOutputStream(s.getOutputStream());
        this.is = new ObjectInputStream(s.getInputStream());
        this.addr = s.getInetAddress();
        this.mdbc = new DBConnection();

```

```

        Connection conn = this.mdbc.getDbConnection();

        try {
            this.stmt =
conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                        ResultSet.CONCUR_UPDATABLE);
            this.usersTable = new UsersTable(this.stmt,
this.mdbc);
            this.ApTable = new AppointmentsTable(this.stmt,
this.mdbc);
        } catch (SQLException e5) {
            System.out.println(e5);
        }
    }

    public void writeObj(String str) {
        this.clientMessage = str;

        try {
            this.os.writeObject(this.clientMessage);
        } catch (IOException e3) {
            System.err.println("I/O thread error" + e3);
        }
    }

    public void run()
    {
        boolean i = false;
        String messageToClient = "";
        String str;
        int temp=0;
        String ThreadStop = "";

        try {
            System.out.println("Сервер ожидает действий от
клиента");

            while (!ThreadStop.equals("Exit")) {
                str = (String) this.is.readObject();
                System.out.println(str);
                String[] messageParts = str.split(",");
                switch (messageParts[0])
                {
                    case "checkLoginClient":
                        String UserLogin = messageParts[1];

```

```

        String UserPassword =
messageParts[2];
        messageToClient =
this.usersTable.CheckLoginClient(UserLogin, UserPassword);
        this.writeObj(messageToClient);
        break;
    case "checkLoginAdmin":
        String UserLogin1 = messageParts[1];
        String UserPassword1 =
messageParts[2];
        messageToClient =
this.usersTable.CheckLoginAdmin(UserLogin1, UserPassword1);
        this.writeObj(messageToClient);
        break;
    case "addClient":

this.usersTable.AddClient(messageParts[1], messageParts[2],
messageParts[3], messageParts[4], messageParts[5]);
        break;
    case "addAdmin":

this.usersTable.addAdmin(messageParts[1], messageParts[2]);
        break;
    case "checkUserInDB":
        messageToClient =
this.usersTable.checkUserInDB(messageParts[1]);
        this.writeObj(messageToClient);
        break;
    case "sendUsername":
        messageToClient =
this.ApTable.getUsername(messageParts[1]);
        this.writeObj(messageToClient);
        break;
    case "sendUser":
        messageToClient =
this.usersTable.getUser();
        this.writeObj(messageToClient);
        break;
    case "sendAdmin":
        messageToClient =
this.usersTable.getAdmin();
        this.writeObj(messageToClient);
        break;
    case "sendClient":
        messageToClient =
this.usersTable.getClient();

```

```

        this.writeObj (messageToClient);
        break;
    case "sendAp":

messageToClient=this.ApTable.getAP (messageParts [1]);
        this.writeObj (messageToClient);
        break;
    case "sendService":

messageToClient=this.ApTable.getService (messageParts [1]);
        this.writeObj (messageToClient);
        break;
    case "sendMaster":

messageToClient=this.ApTable.getMaster (messageParts [1]);
        this.writeObj (messageToClient);
        break;
    case "sendTypeService":

messageToClient=this.ApTable.getTypeService (messageParts [1]);
        this.writeObj (messageToClient);
        break;
    case "sendNameTServ":

messageToClient=this.ApTable.getNameTService ();
        this.writeObj (messageToClient);
        break;
    case "sendNameService":

messageToClient=this.ApTable.getNameService (messageParts [1]);
        this.writeObj (messageToClient);
        break;
    case "sendPrice":

messageToClient=this.ApTable.getPrice (messageParts [1]);
        this.writeObj (messageToClient);
        break;
    case "sendDeleteUser":

this.usersTable.deleteUser (messageParts [1]);
        break;
    case "sendMakeAp":

messageToClient=this.ApTable.AddAp (messageParts [1],messageParts [
2],messageParts [3],messageParts [4],messageParts [5],messageParts [
6]);

```



```

        this.writeObject(messageToClient);
        break;
    case "sendMakeApp":

messageToClient=this.ApTable.AddApp(messageParts[1],messageParts
[3],messageParts[4],messageParts[5],messageParts[6],messageParts
[7]);

        this.writeObject(messageToClient);
        break;
    case "sendNameMaster":

messageToClient=this.ApTable.getNameMaster(messageParts[1]);
        this.writeObject(messageToClient);
        break;
    case "sendCountAp":

messageToClient=Integer.toString(this.ApTable.countOfAppointment
s(messageParts[1]));

        this.writeObject(messageToClient);
        break;
    case "sendCountTypeSv":

messageToClient=Integer.toString(this.ApTable.countOfTypeSv());
        this.writeObject(messageToClient);
        break;
    case "sendCountServ":

messageToClient=Integer.toString(this.ApTable.countOfServices(me
ssageParts[1]));

        this.writeObject(messageToClient);
        break;
    case "sendCountApp":

messageToClient=Integer.toString(this.ApTable.countOfApp());
        this.writeObject(messageToClient);
        break;
    case "sendUserID":

messageToClient=Integer.toString(this.usersTable.getIdUser(messa
geParts[1]));

        this.writeObject(messageToClient);
        break;
    case "sendCountMaster":

messageToClient=Integer.toString(this.ApTable.countOfMasters(mes
sageParts[1]));

```

```

        this.writeObj(messageToClient);
        break;
    case "sendCountUsers":

messageToClient=this.usersTable.getCountUsers();
        this.writeObj(messageToClient);
        break;
    case "sendCountAdmins":

messageToClient=this.usersTable.CountAdmins();
        this.writeObj(messageToClient);
        break;
    case "sendNameClient":

messageToClient=this.usersTable.getNameClient();
        this.writeObj(messageToClient);
        break;
    case "sendCountClients":

messageToClient=this.usersTable.CountClients();
        this.writeObj(messageToClient);
        break;
    case "sendDeleteAdmin":

messageToClient=this.usersTable.deleteAdmin(messageParts[1]);
        this.writeObj(messageToClient);
        break;
    case "sendApp":

messageToClient=this.ApTable.getApp();
        this.writeObj(messageToClient);
        break;
    case "sendSaveAp":
        this.ApTable.saveAp(messageParts[1]);
        break;
    case "sendDeleteClients":

this.usersTable.deleteClient(messageParts[1]);
        break;
    case "sendDeleteAp":

this.ApTable.deleteAp(messageParts[1]);
        break;
    case "sendDataPieS":

messageToClient=this.usersTable.getDataPieS();

```

```

        this.writeObj(messageToClient);
        break;
    case "sendDataPieA":

messageToClient=this.usersTable.getDataPieA(messageParts[1]);
        this.writeObj(messageToClient);
        break;
    case "sendFindAp":

messageToClient=this.ApTable.findAp(messageParts[1]);
        this.writeObj(messageToClient);
        break;

        }}}catch (Exception var17) {
            System.err.println("Соединение закрыто");
        } finally {
            this.disconnect();
        }
    }

    private void disconnect() {
        try {
            if (this.os != null) {
                this.os.close();
            }
            if (this.is != null) {
                this.is.close();
            }
            System.out.println(this.addr.getHostName() + "
Закрытие соединения " + this.counter + "го клиента");
        } catch (IOException var5) {
            var5.printStackTrace();
        } finally {
            this.interrupt();
        }
    }
}

public class AppointmentsTable extends DataTable implements
ResultFromTable{
    public AppointmentsTable (Statement stmt, DBConnection mdbc)
    {
        super(stmt, mdbc);
        try {
            connection=mdbc.getDbConnection();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException throwables) {

```

```

        throwables.printStackTrace();
    }

}

private static Connection connection;

public ResultSet getResultFromTable(String table) {
    ResultSet rs = null;

    try
    {
        rs = this.stmt.executeQuery("SELECT * FROM " +
table);
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
    return rs;
}

public String AddApp(String name,String services, String
master, String date, String time, String price)
{
    String mes="false";
    String clientID[]=name.split(",");
    int masterID=getIdMaster(master);
    try {

        String insert = "INSERT INTO " +
Const.APPOINTMENT_TABLE + "(" +
        Const.APPOINTMENT_CLIENT + "," +
Const.APPOINTMENT_APPOINTMENTS + "," +
        + Const.APPOINTMENT_MASTER+ "," +
+Const.APPOINTMENT_DATE+ "," + Const.APPOINTMENT_TIME+
        "," + Const.APPOINTMENT_PRICE+)" " +
        "VALUES (" +clientID[0]+"," +
this.quotate(services) + "," +masterID+"," + this.quotate(date)
+"," + this.quotate(time)+"," +price+ ")";
        PreparedStatement
pstmt=connection.prepareStatement(insert);
        this.stmt.executeUpdate(insert);
        mes="true";
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

```

        }
        return mes;
    }
    public void deleteAp (String username)
    {
        try {
            stmt.executeUpdate("DELETE FROM " +
Const.APPOINTMENT_TABLE + " WHERE (" + Const.APPOINTMENT_ID + "
LIKE '" + username+ "');");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

ПРИЛОЖЕНИЕ Б

(Обязательное)

SQL-скрипт для генерации базы данных

```

CREATE DATABASE IF NOT EXISTS `hairstresser` /*!40100 DEFAULT CHARACTER SET
utf8mb4 COLLATE utf8mb4_0900_ai_ci */;
USE `hairstresser`;
CREATE TABLE IF NOT EXISTS `hairstresser`.`users` (
  `userID` INT(11) NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`userID`))
ENGINE = InnoDB
AUTO_INCREMENT = 20
DEFAULT CHARACTER SET = utf8mb4;

CREATE TABLE IF NOT EXISTS `hairstresser`.`client` (
  `clientID` INT(11) NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `birthDate` Date NOT NULL,
  `sex` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`clientID`),
  CONSTRAINT `userIDClFK`
    FOREIGN KEY (`clientID`)
    REFERENCES `hairstresser`.`users` (`userID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `hairstresser`.`admin` (
  `adminID` INT(11) NOT NULL AUTO_INCREMENT,

```

```

        `status` VARCHAR(45) NOT NULL,
        PRIMARY KEY (`adminID`),
        CONSTRAINT `userIDFK`
            FOREIGN KEY (`adminID`)
            REFERENCES `hairstresser`.`users` (`userID`)
            ON DELETE CASCADE
            ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 18
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `hairstresser`.`typeOfService` (
    `typeServiceID` INT(11) NOT NULL,
    `name` VARCHAR(55) NOT NULL,
    PRIMARY KEY (`typeServiceID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `hairstresser`.`services` (
    `serviceID` INT(11) NOT NULL AUTO_INCREMENT,
    `typeOfserviceID` INT(11) NOT NULL,
    `name` VARCHAR(65) NOT NULL,
    `price` DOUBLE(11,2) NOT NULL,
    PRIMARY KEY (`serviceID`),
    CONSTRAINT `typeServiceIDFK`
        FOREIGN KEY (`typeOfserviceID`)
        REFERENCES `hairstresser`.`typeOfService` (`typeServiceID`)
        ON DELETE CASCADE
        ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `hairstresser`.`masters` (
    `masterID` INT(11) NOT NULL AUTO_INCREMENT,
    `serviceTypeID` INT(11) NOT NULL,
    `name` VARCHAR(65) NOT NULL,
    `experience` VARCHAR(65) NOT NULL,
    PRIMARY KEY (`masterID`),
    FOREIGN KEY (`serviceTypeID`)
        REFERENCES `hairstresser`.`typeOfService` (`typeServiceID`)
        ON DELETE CASCADE
        ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `hairstresser`.`appointments` (
    `clientID` INT(11) NOT NULL AUTO_INCREMENT,
    `appointments` VARCHAR(65) NOT NULL,

```

```

`masterID` INT(11) NOT NULL,
`date` DATE NOT NULL,
`time` VARCHAR(6) NOT NULL,
CONSTRAINT `clientIDFK`
FOREIGN KEY (`clientID`)
REFERENCES `hairstresser`.`client` (`clientID`)
ON DELETE CASCADE

ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```