Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Проектирование информационных систем»

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

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЗАЦИИ РАБОТЫ КОСМЕТИЧЕСКОГО САЛОНА»**

|  |  |
| --- | --- |
|  | Выполнила студентка группы 074002  Дубаневич Полина Павловна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

СОДЕРЖАНИЕ

[Введение](#_Toc121946415) 5

[1. Анализ и моделирование системы автоматизации работы косметического салона](#_Toc121946416) 7

[1.1 Описание предметной области](#_Toc121946417) 7

[1.2 Разработка функциональной модели предметной области](#_Toc121946418) 10

[1.3 Анализ требований к разрабатываемой системе автоматизации работы косметического салона. Спецификация функциональных требований](#_Toc121946419) .13

[1.4 Разработка информационной модели предметной области](#_Toc121946420) 16

[1.5 UML-модели представления системы автоматизации работы косметического салона и их описание](#_Toc121946421) 17

[2. Проектирование и конструирование системы автоматизации работы косметического салона](#_Toc121946422) 22

[2.1 Постановка задачи](#_Toc121946423) 22

[2.2 Архитектурные решения](#_Toc121946424) 22

[2.3 Описание алгоритмов, реализующих бизнес-логику разрабатываемой системы автоматизации работы косметического салона](#_Toc121946425) 24

[2.4 Проектирование пользовательского интерфейса](#_Toc121946426) 26

[2.5 Обоснование выбора компонентов и технологий для реализации системы автоматизации работы косметического салона](#_Toc121946427) 28

[3 Тестирование и проверка работоспособности системы автоматизации работы косметического салона](#_Toc121946428) 31

[4 Руководство по развертыванию и использованию системы автоматизации работы косметического салона](#_Toc121946429) 35

[Заключение](#_Toc121946430) 42

[Список использованных источников](#_Toc121946431) 43

[Приложение А](#_Toc121946435) 44

[Отчет о проверке на заимствования в системе «Антиплагиат»](#_Toc121946435) 44

[Приложение Б](#_Toc121946435) 45

[Листинг кода алгоритмов, реализующих бизнес-логику](#_Toc121946437) 45

[Приложение В](#_Toc121946438) 50

[Листинг скрипта генерации базы данных 50](#_Toc121946440)

**Перечень условных обозначений, символов и терминов**

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

# **ВВЕДЕНИЕ**

Информационные технологии используются в нашей жизни ежедневно. С их помощью мы можем общаться, находить нужную информацию, учиться, работать, а также за пару минут бронировать столик в ресторане или подбирать подходящий салон красоты и записываться на различные процедуры без звонка администратору. Современные информационные технологии – техника, благодаря которой жизнь многих людей стала проще и легче.

Тема курсового проекта – разработка системы автоматизации работы косметического салона. Данная тема является актуальной как для потребителя, так и для самого бизнеса, ведь часто салоны, не имея качественной системы автоматизации, теряют своих клиентов. В такой конкурентоспособной отрасли, как велнес-индустрия, потеря даже одного клиента стоит очень дорого.

Со стороны клиента можно сказать, что запись на процедуру онлайн намного удобнее, чем лично или по телефону, так как позволяет просмотреть все услуги, оценить их стоимость и прочитать отзывы о мастерах. Автоматизация поможет сэкономить время на рутинных операциях, которые с легкостью может сделать компьютер, а также упростить систему контроля и отчетности за деятельностью салона.

Объектом исследования является процесс автоматизации системы работы косметического салона. Предмет исследования – создание системы для автоматизации деятельности салона красоты.

Таким образом, целью данного курсового проекта является повышение качества обслуживания клиентов путем интеграции системы автоматизации работы косметического салона.

Подобные программы позволяют оптимизировать бизнес-процессы, снизить бумажный документооборот и трату человеческих ресурсов, собирая все необходимые данные по клиентской базе в одном месте, автоматизировать взаимоотношения с клиентами, анализировать предпочтения потребителей и увеличивать доходы от оказанных услуг. Также путем онлайн записи можно не только сэкономить время администратора, но и отследить путь клиента от начала поиска салона до момента записи и определить наиболее эффективные рекламные стратегии.

Задачи данного курсового проекта:

* проанализировать и смоделировать систему работы косметического салона;
* спроектировать программное средство, автоматизирующее такую систему;
* протестировать и проверить работоспособность системы;
* описать руководство по использованию для пользователей.

Программа, реализованная в ходе курсового проекта, поможет клиентам значительно быстрее и легче записывать на желаемую процедуру, а администраторам и мастерам будет проще отслеживать записи и работы косметического салона. Также актуальность данной темы обусловлена большой популярностью косметических процедур среди населения.

# **Анализ и моделирование СИСТЕМЫ АВТОМАТИЗАЦИИ РАБОТЫ КОСМЕТИЧЕСКОГО САЛОНА**

## **Описание предметной области**

Основное преимущество автоматизации – это сокращение избыточности хранимых данных, а следовательно, экономия объема используемой памяти, уменьшение затрат на многократные операции обновления избыточных копий и устранение возможности возникновения противоречий из-за хранения в разных местах сведений об одном и том же объекте, увеличение степени достоверности информации и увеличение скорости обработки информации; излишнее количество внутренних промежуточных документов, различных журналов, папок, заявок и т.д., повторное внесение одной и той же информации в различные промежуточные документы. Также значительно сокращает время автоматический поиск информации, который производится из специальных экранных форм, в которых указываются параметры поиска объекта.

Автоматизированная система управления — комплекс аппаратных и программных средств, а также персонала, предназначенный для управления различными процессами в рамках технологического процесса, производства, предприятия.

АСУ применяются в различных отраслях промышленности, энергетике, транспорте и т. п. Термин «автоматизированная», в отличие от термина «автоматическая», подчёркивает сохранение за человеком-оператором некоторых функций, либо наиболее общего, целеполагающего характера, либо не поддающихся автоматизации.

Продвижение косметического салона эффективнее осуществляется за счет выстраивания взаимоотношений с клиентами. Их нужно уведомлять о новых услугах, акциях и скидках. Упростить задачи поможет автоматизированная система работы салона. Это специальная система, в которой хранятся данные о клиентах и фиксируются все переговоры с ними. Комплекс помогает выявить слабые места в управлении салоном и определить пути дальнейшего продвижения бизнеса. [9]

В косметической сфере всегда была высокая конкуренция. И сейчас она намного сильнее. Главный вопрос в повестке дня любого салона – это, как привлечь клиентов и удержать их. Бизнес должен неустанно контролироваться по качеству и актуальности предоставляемых услуг. Именно с помощью автоматизации можно выявить недочеты в работе салонов.

Студии красоты внедрились в нашу жизнь довольно давно. Это значит, что существует множество систем для учета деятельность салона. Рассмотрим несколько из них.

1. Битрикс 24- система оснащена современными и удобными функциями, которые можно применять для полноценного учета клиентов в небольшом или крупном салоне красоты. Программа легко устанавливается на компьютер либо мобильное устройство, ее не нужно специально настраивать. Необходимо зарегистрироваться, и приложение станет доступным. Основные достоинства: любые коммуникации с клиентами сохраняются в базе данных, система сама направляет клиента к цели, по почте и СМС клиентам высылаются уведомления об акциях, контроль за продажами и работой менеджеров.
2. S2 (SalesapCRM) S2 — программа для эффективных продаж с бесплатным тарифом. В ней можно вести учет клиентов и сделок, управлять задачами, сохранять историю работы с каждым посетителем, а также его письма, звонки и информацию о визитах/платежах. При этом S2 обладает набором функций специально для сферы услуг: календарь для записи клиентов к разным специалистам, виджет для самостоятельной онлайн-записи через сайт, калькулятор стоимости услуг, автоматические sms-напоминания о времени визита, email-рассылки клиентам и пр.
3. Клиентикс CRM - Основное преимущество программы — это простота в использовании. У нее понятный интерфейс, что позволяет пользователям быстро освоить функционал. Нужная информация сразу выводится на экран, и даже поверхностный анализ данных поможет понять, какую прибыль получил салон за определенный период. В течение, максимум, трех дней пользователи в состоянии изучить функционал сервиса полностью. С помощью системы можно мгновенно рассчитать зарплату сотрудникам.

Проведя анализ видов записи клиентов было выявлено, что большинство из них используют автоматизированную систему. (Рисунок 1.1)

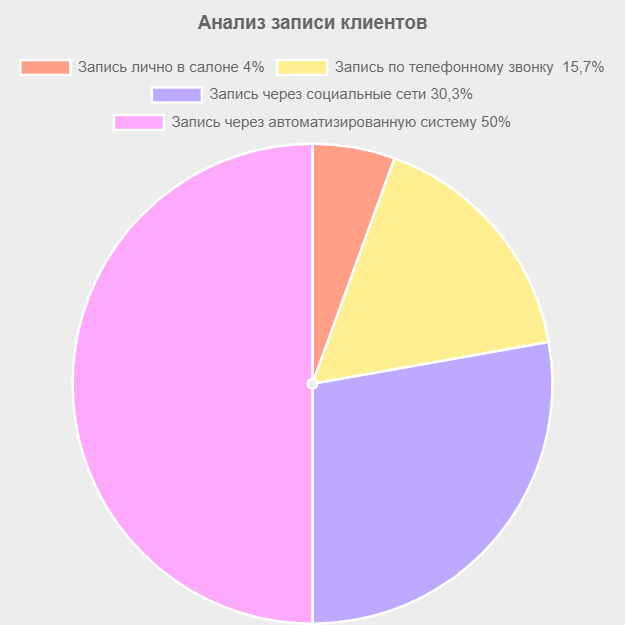


Рисунок 1.1 – Круговая диаграмма анализа записи клиентов

Также был проведён анализ самых востребованных процедур у клиентов, исходя из этого, можно расположить запись на сайте в удобном для потребителей порядке. (Рисунок 1.2)

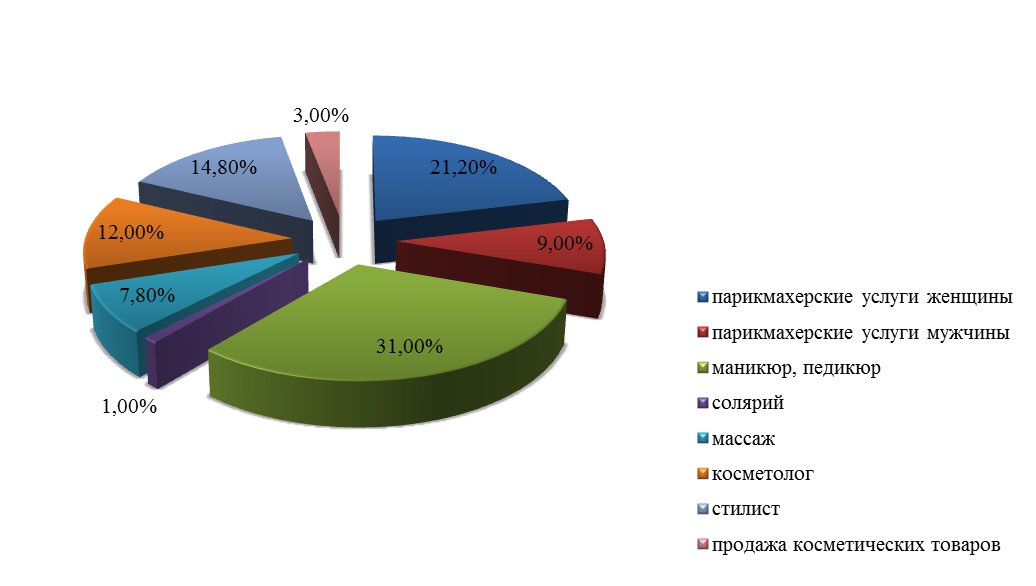


Рисунок 1.2 – Круговая диаграмма самых востребованных процедур в салоне

Таким образом, чтобы самостоятельно открыть салон не нужно тратить долгие годы на изучение бизнеса. В этом поможет автоматизированная система. Она проста в использовании, имеет множество полезных функций для ведения успешного бизнеса в сфере индустрии красоты. Уже с первых недель использования, вы увидите положительные результаты и сможете продвигать свое заведение в лидеры.

## **Разработка функциональной модели предметной области**

IDEF0 — методология функционального моделирования.

Описание системы с помощью IDEF0 называется функциональной моделью. Функциональная модель предназначена для описания существующих бизнес-процессов, в котором используются как естественный, так и графический языки. Для передачи информации о конкретной системе источником графического языка является сама методология IDEF0.[4]

Методология IDEF0 предписывает построение иерархической системы диаграмм - единичных описаний фрагментов системы. Сначала проводится описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция - система разбивается на подсистемы и каждая подсистема описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности. (Рисунок 1.3)

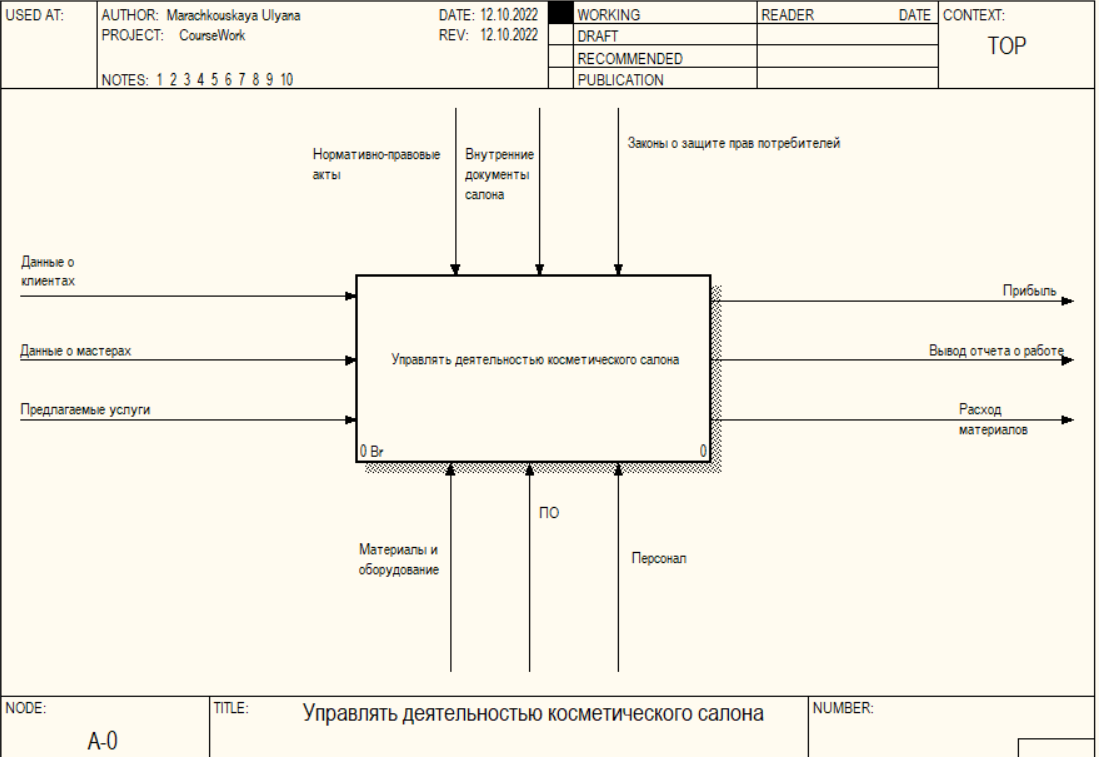


Рисунок 1.3 – Верхний уровень функциональной модели

На контекстной диаграмме верхнего уровня представлена функциональная модель «Управлять деятельностью косметического салона», а также определены потоки входных и выходных данных, механизмы ограничения и управления данными.

На рисунке 1.4 указан путь клиента от регистрации в самой системе вплоть до расчета после оказанной процедуры.

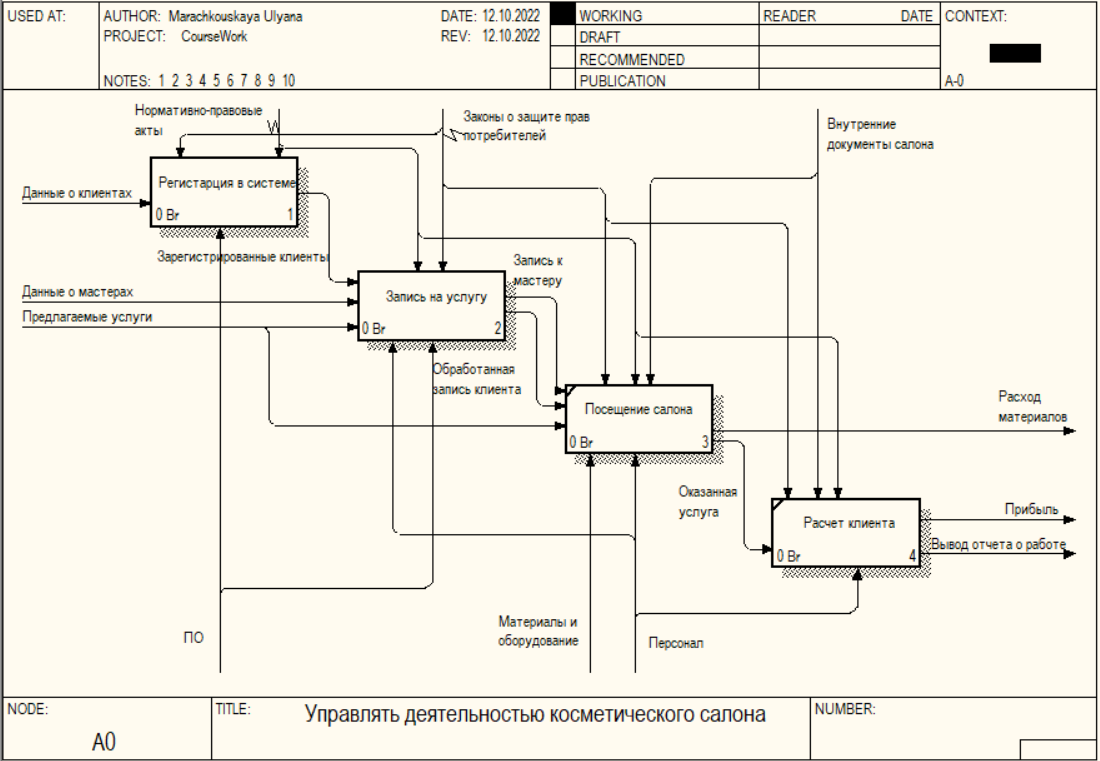


Рисунок 1.4 – Декомпозиция контекстной диаграммы

Декомпозиция контекстной диаграммы состоит из 4 блоков:

1. Регистрация в системе
2. Запись на услугу
3. Посещение салона
4. Расчет клиента

Далее была проведена декомпозиция подфункций. На рисунке 1.5 изображена декомпозиция блока «Регистрация в системе». После регистрации в системе будет доступен ряд функций для удобной онлайн-записи клиентами.

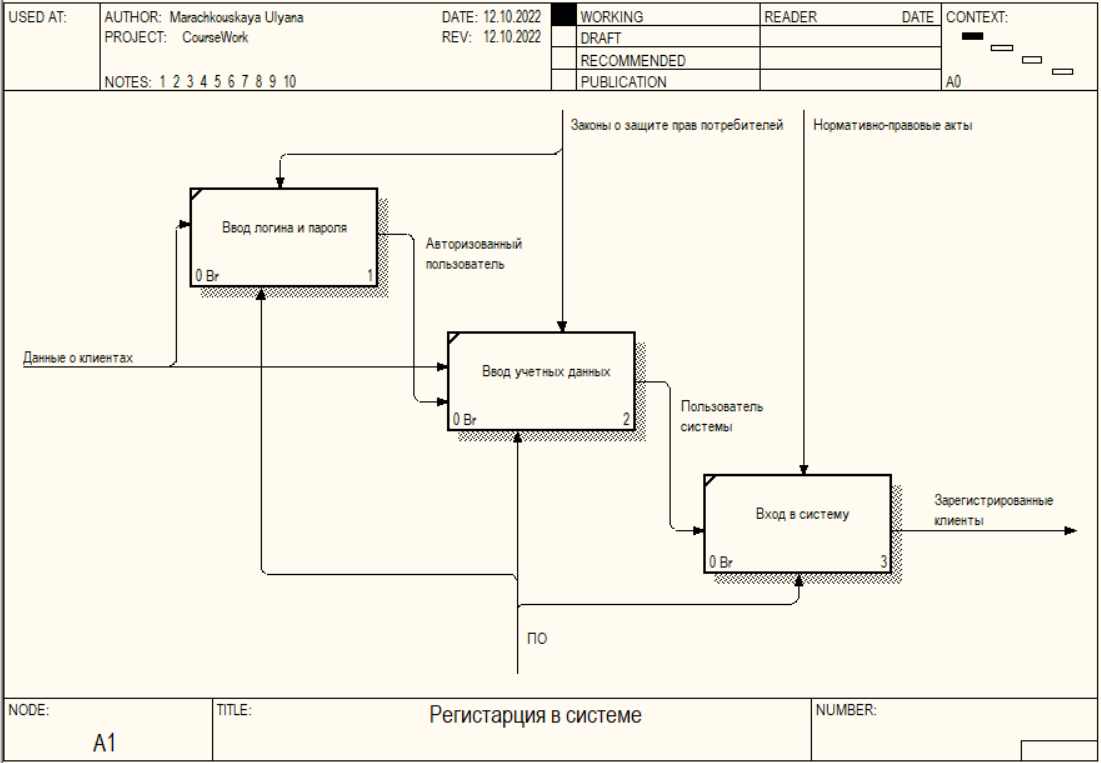


Рисунок 1.5 – Декомпозиция блока «Регистрация в системе»

Данный блок разбит на 3 функции: «Ввод логина и пароля», «Ввод учетных данных» и «Вход в систему».

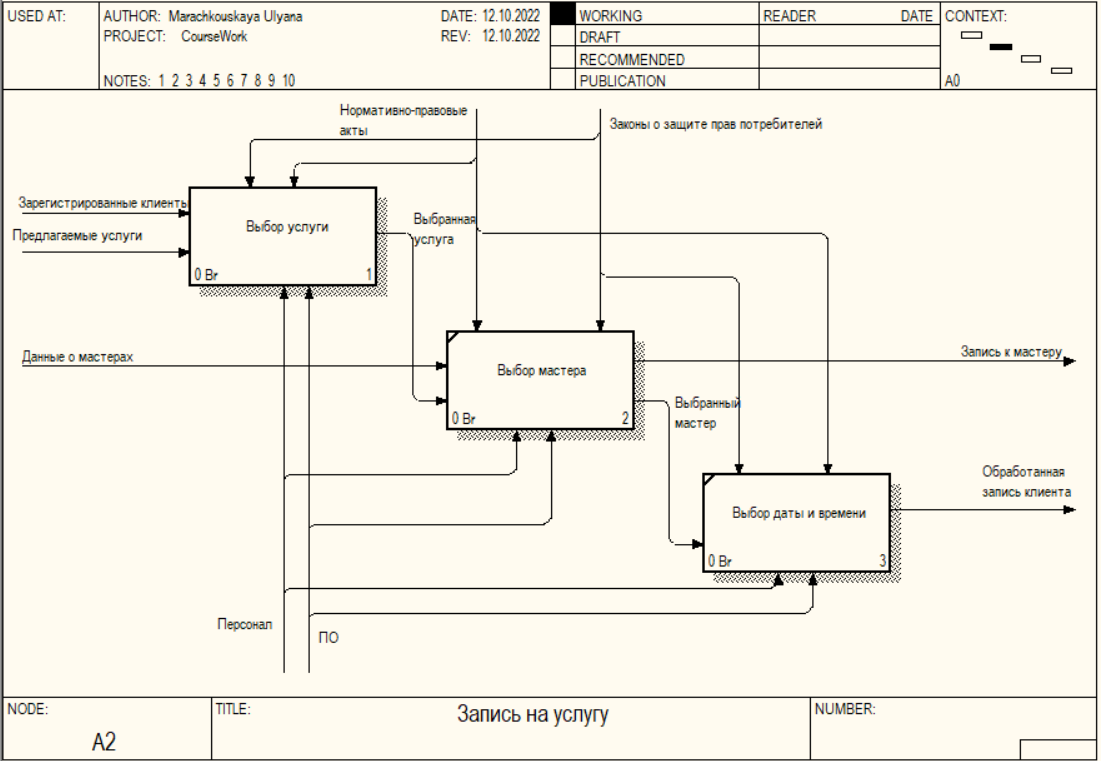


Рисунок 1.6– Декомпозиция блока «Запись на услугу»

Запись на услугу осуществляется в 3 этапа: выбор услуги, выбор мастера и выбор даты и времени.

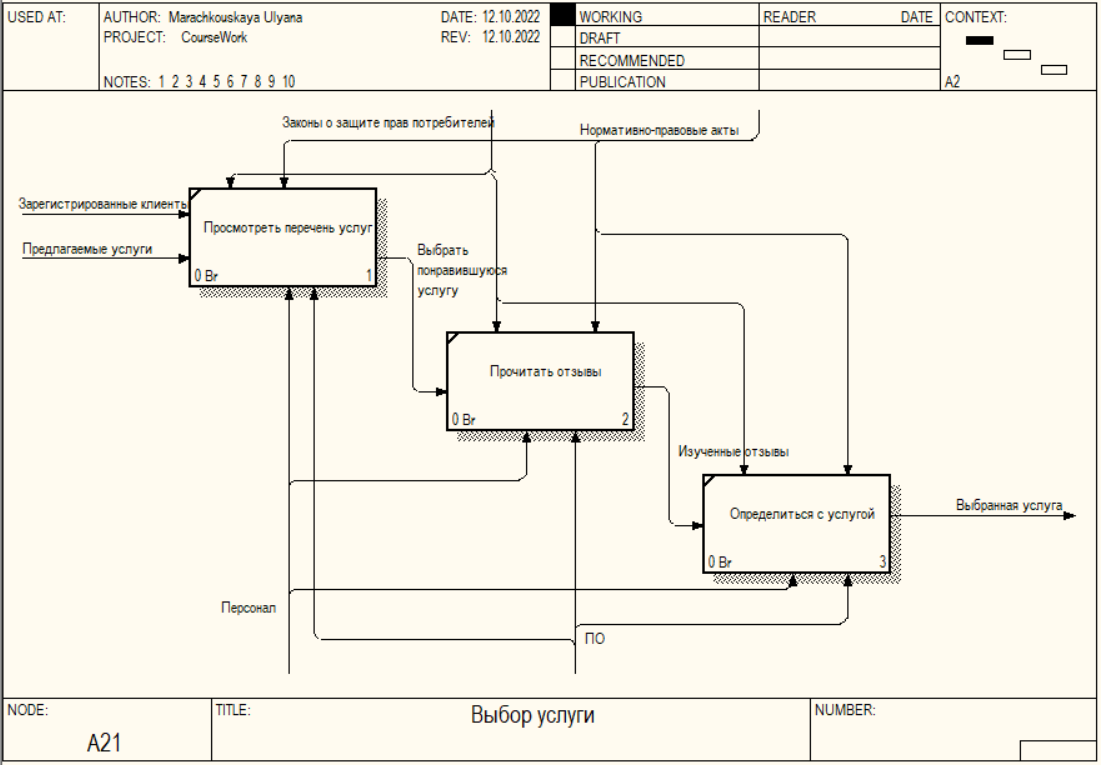


Рисунок 1.7 – Декомпозиция блока «Выбор услуги»

Для того чтобы выбрать услугу нужно:

1. Просмотреть перечень услуг
2. Прочитать отзывы
3. Определиться с услугой

## **Анализ требований к разрабатываемой системе автоматизации работы косметического салона. Спецификация функциональных требований**

Для разработки подобной информационной системы нужно вникнуть в процесс управления косметическим салоном.

В любом косметическом салоне есть достаточное количество посетителей, мастеров и управленческого персонала, соответственно, есть потребность в хранении записей, отчетов и процедур. Все данные, полученные программой, должны храниться в одном месте, для этого будет использован *MYSQL* *Server*. «*MySQL*» - одна из самых популярных баз данных для веб-приложений. Фактически, является стандартом *de facto* для веб-серверов, которые работают под управлением операционной системы *Linux*. *MySQL* - это бесплатный пакет программ, однако новые версии выходят постоянно, расширяя функционал и улучшая безопасность. Существуют специальные платные версии, предназначенные для коммерческого использования. В бесплатной версии наибольший упор делается на скорость и надежность, а не на полноту функционала, который может стать и достоинством и недостатком - в зависимости от области внедрения.

Разработку и поддержку *MySQL* осуществляет корпорация *Oracle*, получившая права на торговую марку вместе с поглощённой *Sun Microsystems*, которая ранее приобрела шведскую компанию *MySQL* *AB*. Продукт распространяется как под *GNU General Public License*, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

Эта СУБД позволяет выбирать различные движки для системы хранения, которые позволяют менять функционал инструмента и выполнять обработку данных, хранящихся в различных типах таблиц. 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. Более того, СУБД *MySQL* поставляется со специальным типом таблиц *EXAMPLE*, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и *GPL*-лицензированию, в СУБД *MySQL* постоянно появляются новые типы таблиц. Она также имеет простой в использовании интерфейс, и пакетные команды, которые позволяют удобно обрабатывать огромные объемы данных. Система невероятно надежна и не стремится подчинить себе все доступные аппаратные ресурсы.

Достоинства:

* распространяется бесплатно;
* прекрасно документирована;
* предлагает много функций, даже в бесплатной версии;
* пакет *MySQL* включен в стандартные репозитории наиболее распространённых дистрибутивов операционной системы *Linux*, что позволяет устанавливать её элементарно;
* поддерживает набор пользовательских интерфейсов;
* может работать с другими базами данных, включая *DB2* и *Oracle*.

Недостатки:

* придётся потратить много времени и усилий, чтобы заставить *MySQL* выполнять несложные задачи, хотя другие системы делают это автоматически, например: создавать инкрементные резервные копии;
* отсутствует встроенная поддержка *XML* или *OLAP*;
* для бесплатной версии доступна только платная поддержка;
* идеально подходит для: организаций, которым требуется надежный инструмент управления базами данных, но бесплатный.

Хранение данных таким образом поможет упорядочить информацию и получить к ней доступ в любое время, а также поможет сохранить важные документа и записи при непредвиденном сбое системы. Подключение к базе данных будет осуществляться при авторизации и в зависимости от вида пользователя будет видна доступная ему информация. Например, если пользователь авторизовался под логином администратора, то ему будет доступна информация о записях, клиентах, мастерах и услугах, в то время, как при авторизации под логином клиента эта информация будет не видна.

Для работы администратора будет предоставлена онлайн-запись клиентов, которые хотят записаться по телефону или лично в салоне, а также составление расписания, перечня услуг, просмотр важной документации и отчетов для улучшения работы салона.

Диаграмма вариантов использования отражает весь функционал субъектов, участвующий в системе. В качестве главных субъектов выделены: администратор и клиенты. (Рисунок 1.8)

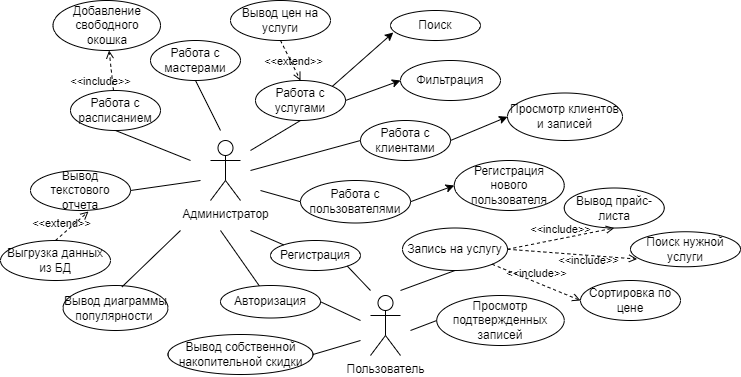


Рисунок 1.8 – Диаграмма вариантов использования системы

На данной диаграмме изображены действия, которые могут осуществлять пользователи в зависимости от вида авторизации в системе. Авторизоваться можно под видом администратора или клиента. Клиент салона может осуществить авторизацию и регистрацию, просмотреть нужную информацию и свои уже подтвержденные записи, выбрать услугу и записаться на нее онлайн. У администратора значительно больше возможностей:

- Просмотр, добавление, редактирование и удаление данных;

- Разработка расписания;

- Запись клиентов онлайн;

- Просмотр отчетов;

- Работа с перечнем услуг;

- Вывод самых популярных услуг.

## **1.4 Разработка информационной модели предметной области**

При проектировании системы было использовано 6 сущностей:

* *User;*
* *Client;*
* *Master;*
* *Service;*
* *Shedule;*
* *Admin.*

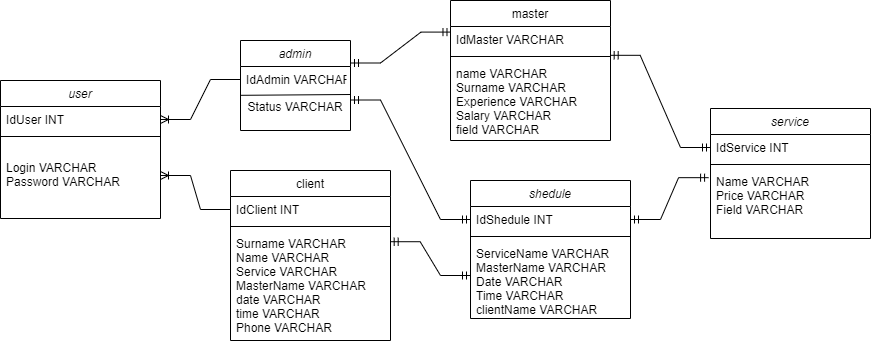


Рисунок 1.9 – Информационная модель системы

Ниже представлено подробное описание всех сущностей, входящих в модель.

Сущность *User* содержит в себе следующие атрибуты:

* *User\_id* – хранит идентификационный номер пользователя;
* *Password, login* – отвечает за хранение пароля и логина пользователя;

Сущность *Client* наследуется от сущности *User* и содержит следующие атрибуты:

* *Client\_id* – хранит идентификационный номер;
* *name*, *surname* – хранит фио клиента;
* *Service*– хранит процедуру, которую выбрал клиент;
* *MasterName*– хранит имя мастера;
* *date* – хранит дату проведения процедуры;
* *time* - хранит время проведения процедуры;
* *phone* – хранит телефон клиента.

Сущность *admin* наследуется от сущности *User* и содержит следующие атрибуты:

* *IdAdmin* – хранит идентификационный номер администратора.

В свою очередь сущность *Service* наследуются от двух других: *Shedule* и *Master*. У сущности *Service* присутствуют следующие атрибуты:

* *IdService* –уникальный номер посетителя;
* *Name* ­ – хранит название процедуры;
* *Price* – хранит цену процедуры;
* *Field* – хранит область процедуры.

У сущности *Master* присутствуют следующие атрибуты:

* *IdMaster* – хранит идентификационный номер;
* *experience* – хранит опыт работы мастера;
* *name*– хранит имя мастера;
* *surname* – хранит фамилию мастера;
* *Salary* – хранит заработную плату мастера;
* *field* - хранит область работы.

Сущность *Shedule* содержит атрибуты:

* *IdShedule* – хранит уникальный номер выполняемой услуги;
* *ServiceName*– хранит название процедуры;
* *MasterName* – хранит имя мастера;
* *Date* – хранит дату проведения процедуры;
* *Time* – хранит время проведения процедуры;
* *ClientName –* хранит фио клиента.

## **UML-модели представления системы автоматизации работы косметического салона и их описание**

UML — это язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур. Диаграмма – это графическое представление набора элементов, чаще всего изображенного в виде связного графа вершин (сущностей) и путей (связей). В данной главе описаны и показаны 3 вида диаграмм: диаграмма состояние, диаграмма последовательности и диаграмма развертывания.[1]

Главное предназначение диаграммы состояний – описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла. Чаще всего диаграммы состояний используются для описания поведения отдельных экземпляров классов (объектов), но они также могут быть применены для спецификации функциональности других компонентов моделей, таких как варианты использования, актеры, подсистемы, операции и методы.

На рисунке 1.10 представлена диаграмма состояний.

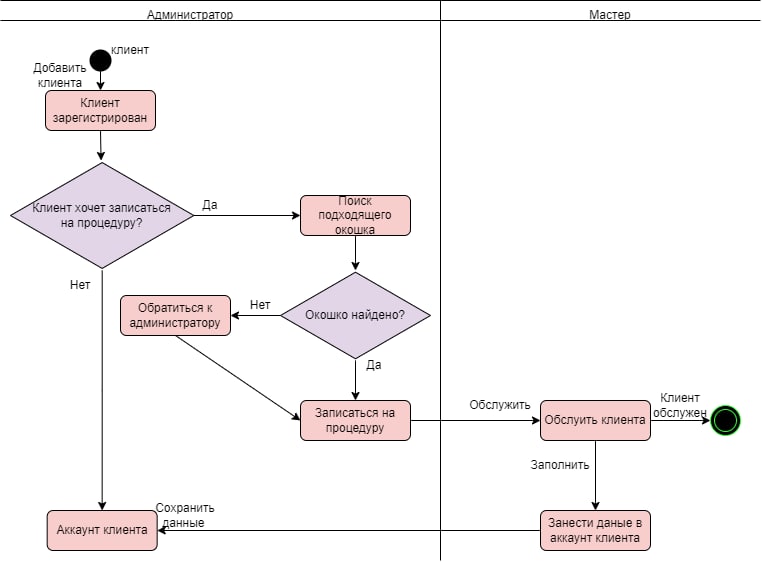


Рисунок 1.10 – Диаграмма состояний обслуживания клиента

На данной диаграмме описаны состояния проведения процедуры в салоне. Для начала клиент регистрируется и начинает записываться на процедуру. Если же подходящее окошко не было найдено на помощь приходит администратор. Далее клиента записывают на процедуру к мастеру. Задача мастера – обслужить клиента и занести данные в базу о том, что клиент был обслужен.

Диаграммы взаимодействия описывают поведение взаимодействующих групп объектов. Как правило, диаграмма взаимодействия охватывает поведение объектов в рамках только одного варианта использования. На такой диаграмме отображается ряд объектов и те сообщения, которыми они обмениваются между собой. Диаграмма последовательности (англ. *sequence* *diagram*) – диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл (создание-деятельность-уничтожение) и взаимодействие (отправка запросов и получение ответов). Подобно диаграммам коммуникации, диаграммы последовательности (*Sequence* *Diagrams*) отображают поток событий в конкретном сценарии варианта использования, но в них больше внимания уделяется именно времени.

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

На данной диаграмме объекты располагаются слева направо таким образом, чтобы крайним с лева был тот объект, который инициирует взаимодействие.

Неотъемлемой частью объекта на диаграмме последовательности является линия жизни объекта. Линия жизни показывает время, в течение которого объект существует в системе. Периоды активности объекта в момент взаимодействия показываются с помощью фокуса управления. Временная шкала на диаграмме направлена сверху вниз.

Таким образом, диаграмма последовательности иллюстрирует последовательность действий, реализующих вариант использования «Запись на услугу». Диаграмма последовательности для выбранного варианта использования представлена на рисунке 1.11

На приведённой диаграмме представлена последовательность всего жизненно цикла взаимодействия пользователя с интерфейсом, серверной частью системы. Первым делам пользователь открывает все возможные записи и услуги. После чего пользователь может выбрать услугу, сервер проверит ее наличии и отправит пользователю соответствующее сообщение.

Вариант использования «Запись на услугу» представлен на рисунке 1.11

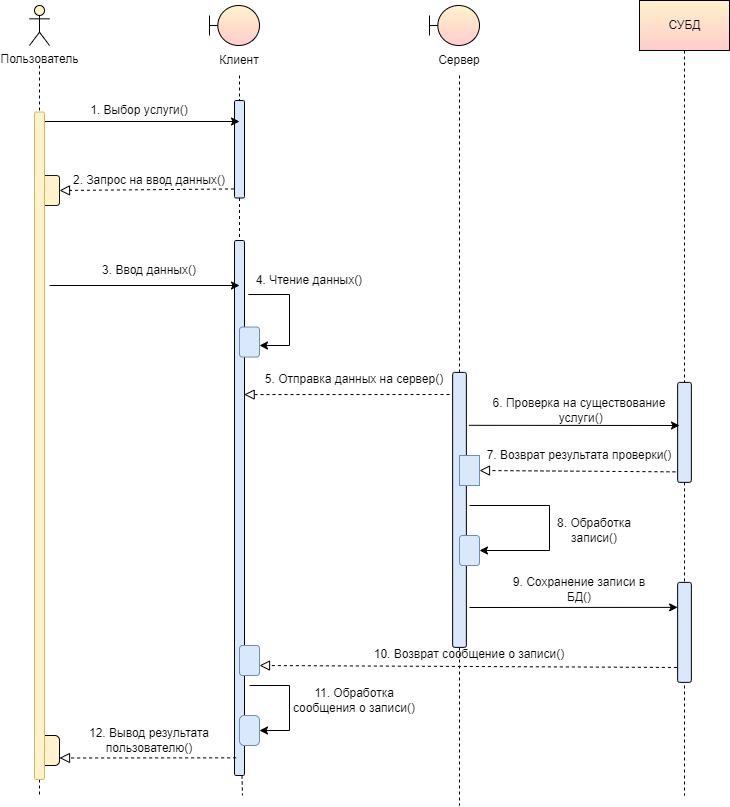


Рисунок 1.11 – Диаграмма последовательности

Пользователь выбирает услугу, нажимая на кнопку «Запись», товар добавляется в корзину данного пользователя. После этого он может перейти в корзину и оплатить данную услугу. Затем данные отправляются на сервер обрабатывающий сервис. После обработки данные записываются в соответствующую таблицу в базе данных. Ответом служит подтверждающее сообщение о записи отображаемое на экране пользователя, а также, данное сообщение дублируется на почту.

При физическом проектировании распределенных программных систем необходимо определить наиболее оптимальный вариант размещения программных компонентов на реальном оборудовании в локальной или глобальной сетях. Для этого используют диаграмму развертывания, показанную на рисунке 1.12.

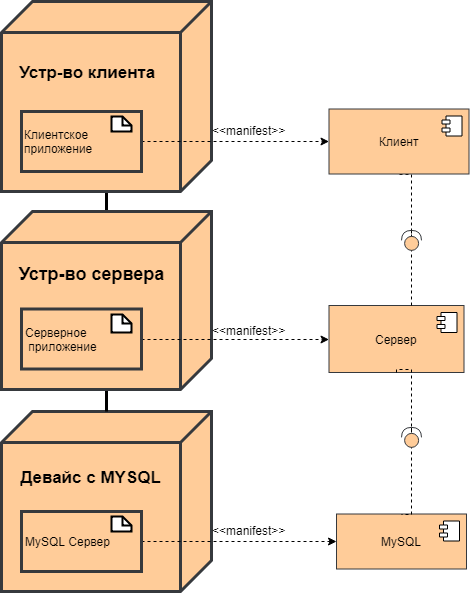


Рисунок 1.12 – Диаграмма развертывания

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

# **ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ СИСТЕМЫ АВТОМАТИЗАЦИИ РАБОТЫ КОСМЕТИЧЕСКОГО САЛОНА**

## **2.1 Постановка задачи**

Целью данного курсового проекта является повышение качества обслуживания клиентов путем интеграции системы автоматизации работы косметического салона. Выделены следующие сущности, принимающие участие в процессе управления:

* администратор;
* пользователь.

Ввиду необходимой одновременной работы нескольких пользователей была выбрана система многопоточный клиент-сервер с использованием TCP протокола. Выдвинуты следующие требования к разрабатываемому программному продукту:

* авторизация в системе и сохранение личных данных;
* поиск свободных окошек на выбранную услугу;
* возможность записи на услугу;
* вывод отчета по проведенной работе.

Техническое обеспечение должно гарантировать высокую надежность технических средств управления, организацию режимов работы.

Система должна позволить централизовать управление всеми процессами, связанными с записью на услугу и выбором процедуры, наличие доступа к управлению услугами и актуальными ценами, позволить администратору своевременно получать достоверную информацию, и исходя из этого, строить правильную систему работы салона красоты.

Система будет спроектирована в виде GUI приложения с использованием стандартных библиотек пользовательского интерфейса.

## **2.2 Архитектурные решения**

Диаграммы классов используются при моделировании программных средств наиболее часто. На диаграммах классов показываются классы, интерфейсы и отношения между ними. В данном курсовом проекте были реализованы классы: *User*, *Client*, *Admin, Master, Shedule, Service, Person*. Все они используются для хранения информации о сущностях, присутствующих в системе. Сущность User используется для регистрации нового пользователя. Сущность *Client* нужна для уже авторизованного пользователя, который записан на процедуру. Сущность *Master* используется для заполнения анкеты мастера при добавлении его в систему. Сущность *Shedule* представляет собой данные о расписании услуг. Сущность *Service* помогает сохранить данные о добавленной услуге. (Рисунок 2.1)

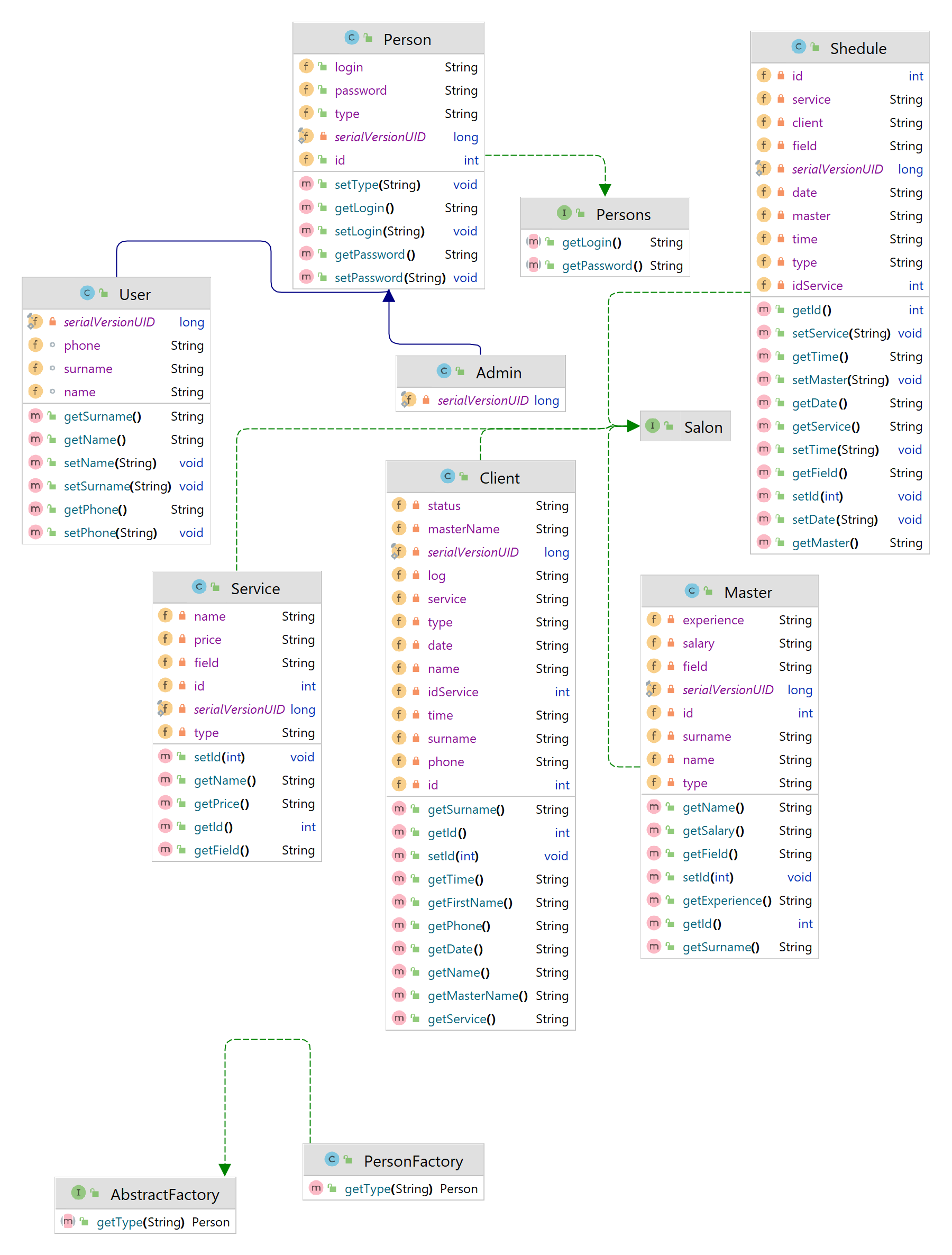


Рисунок 2.1 – Диаграмма классов пакета *classes*

Разрабатываемый программный продукт представляет собой клиент-серверное приложение для работы с СУБД *MySQL Server* через веб-сервис. Предполагается развертывание в организации, имеющей локальную вычислительную сеть с выделенным сервером. В таком случае на выделенном сервере устанавливается СУБД, на которой развертывается база данных системы, и создается директория для хранения документов. Экземпляры клиентских приложений размещаются на машинах сотрудников организации. Доступ клиентских приложений к базе данных осуществляется при помощи драйвера *jdbc data provider for MySQL Server*, входящего в состав *JDK*. Данные, необходимые для установки соединения с сервером, хранятся в конфигурационном файле клиентского приложения и доступны для настройки пользователем. Настройки, используемые всеми приложениями системы, хранятся в одной из таблиц базы данных. Это позволяет производить изменения, которые распространяются сразу на все экземпляры приложений, с другой стороны, доступ к настройкам системы закрыт для рядовых пользователей.

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

## **2.3 Описание алгоритмов, реализующих бизнес-логику разрабатываемой системы автоматизации работы косметического салона**

При запуске открывается главное окно, в котором пользователь должен пройти авторизацию. При успешной авторизации в зависимости от уровня доступа, пользователь получает разный набор возможностей для работы с программой. Если выполнен вход под администратором, то пользователь имеет доступ к базам данных клиентов, мастеров, услуг и расписания. Если же была выполнена авторизация в качестве пользователя, то функционал программы предоставляет возможность записаться на нужную услугу. (Рисунок 2.3)

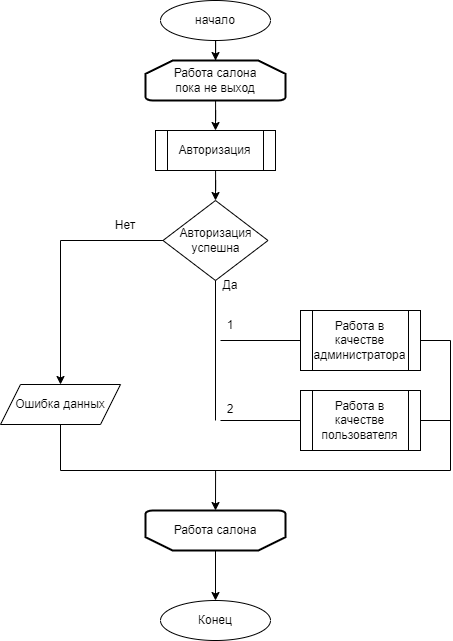


Рисунок 2.3 – Блок-схема алгоритма авторизации

Алгоритм записи клиента является одним из целей данного проекта.. Данный алгоритм изображён на рисунке 2.4. Пользователь проход регистрацию либо авторизацию, если его данные уже есть в базе. Далее он может выбрать любое из предложенных окошек, если же данное окошко после проверки наличия в базе данных не обнаруживается, то выводится сообщение об ошибке, если окошко найдено, то клиент на него записывается.

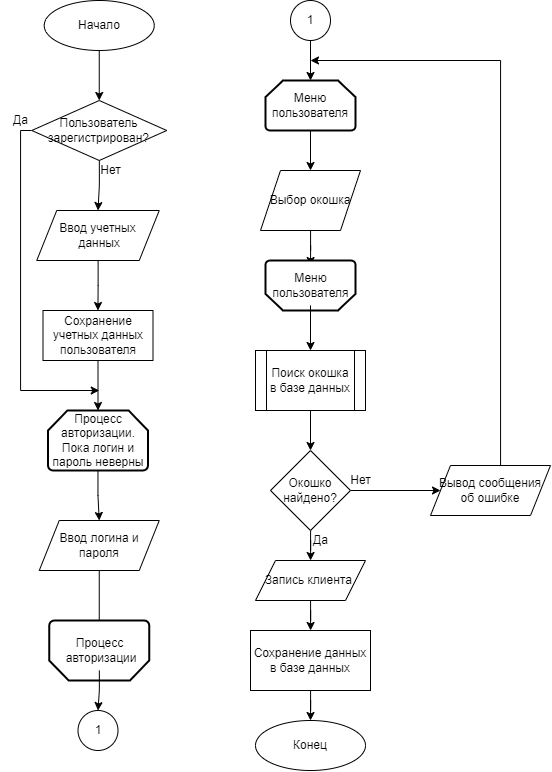


Рисунок 2.4 - Алгоритм записи клиента

Данные алгоритмы нам наглядно показывают логику работы приложения.

## **Проектирование пользовательского интерфейса**

Пользовательский интерфейс – это система средств взаимодействия пользователя с компьютером, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана (окон, значков, меню, кнопок, списков и т.п.). При этом, в отличие от интерфейса командной строки, пользователь имеет произвольный доступ (с помощью клавиатуры или указательного устройства ввода) ко всем видимым экранным объектам, а на экране реализуется модель мира в соответствии с некоторой метафорой и осуществляется прямое манипулирование.

Интерфейс помогает двум объектам понимать друг друга и обмениваться информацией. Интерфейс — это «язык общения», который понимают оба объекта, которые взаимодействуют друг с другом с целью решить определенный вопрос.

Если каждое приложение или программа, установленная на компьютере, планшете или смартфоне, — это помощник, то интерфейс — это способ общаться (взаимодействовать) с ней, чтобы она помогала в вашем деле на работе и в жизни.

В данном проекте пользовательский интерфейс выполнялся при помощи Scene Builder, Scene Builder — это инструмент, с помощью которого мы можем конструировать наши окна в виде графического интерфейса и после их сохранять, и эта программа на основании результата будет конструировать xml файлы, которые мы будем подтягивать в нашем приложении.

Программный интерфейс реализован в фирменных тонах бренда. На рисунке 2.5 мы можем увидеть окошко входа в систему.

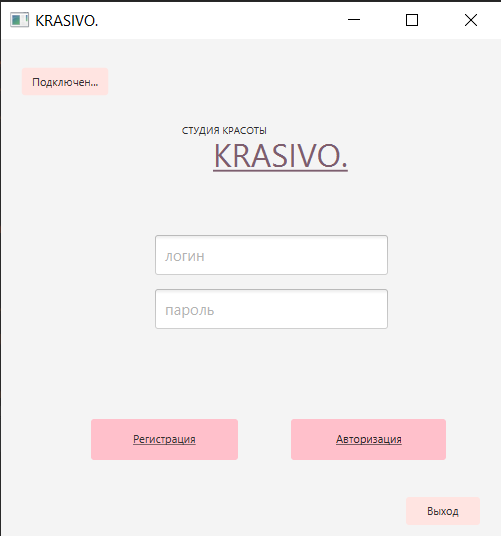


Рисунок 2.5 – Окно входа

Программный интерфейс для администратора реализован в виде так называемой таблице с кнопками для входа в меня 2-го уровня. Для начала администратор выбирает с кем или чем работать, а именно с мастерами, клиентами, услугами или расписанием. После выбирает нужное действие.



Рисунок 2.6 – Интерфейс администратора

Интерфейс пользователя немного проще. Здесь 2 основные таблицы, в верхней таблице есть доступные записи, а в нижней подтвержденные записи пользователя. Также есть поиск и персональная скидка. (Рисунок 2.7)

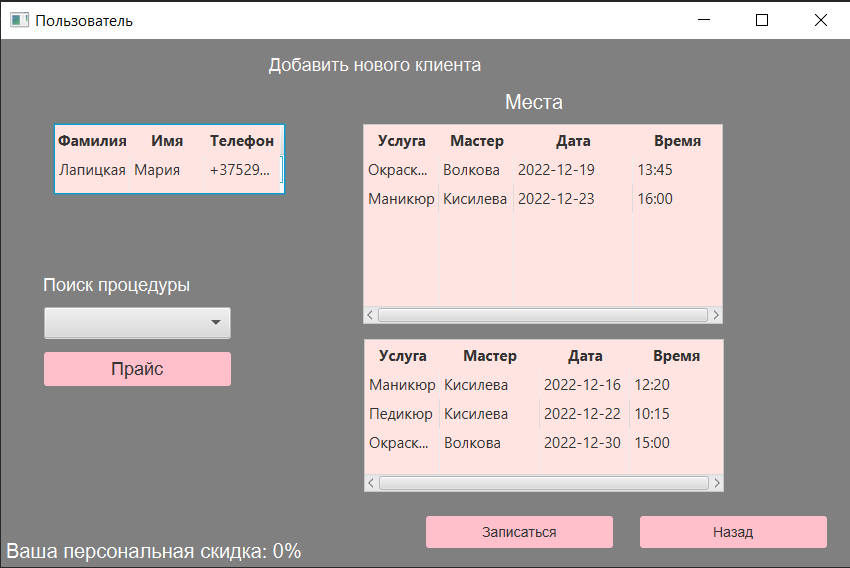


Рисунок 2.7 – Интерфейс пользователя

Данный интерфейс достаточно простой для пользователя, что поможет быстро выбрать нужное окошко и услугу, а также узнать о своей скидке.

## **Обоснование выбора компонентов и технологий для реализации системы автоматизации работы косметического салона**

Первым этапом разработки программного обеспечения является проектирование. Для более наглядного представления деятельности организации, а именно создания контекстной диаграммы и диаграммы декомпозиции, использовалась программа Allfusion Process Modeler или BPwin.

BPwin – программный продукт в области реализации средств CASE-технологий. Данная программа помогает документировать все важные аспекты бизнес-процесса, производить планирование и записывать способы достижения планов, исследовать необходимые для этого ресурсы, а также изображать достигаемые результаты. Process modeler является довольным эффективным инструментом для ведения бизнеса. Он помогает проводить анализ работы как больших предприятий, так и маленьких подразделений.[8]

Таким образом на первом этапе проектирования необходимо создать модели на основе стандартов IDEF0 и IDEF1X.

IDEF0 – методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является ее акцент на соподчиненность объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность (поток работ).

Второй этап при создании ПО – разработка. На данном этапе будет создан интерфейс приложения, программный код для серверной и клиентской части программы, также будет создана база данных и заполнена всей необходимой информацией

По заданию приложение должно быть реализовано на языке Java.

Java *–* строго типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Разработка ведётся сообществом, организованным через Java Community Process, язык и основные реализующие его технологии распространяются по лицензии GPL.

Для разработка была выбрана наиболее популярная и удобная среда разработки: IntelliJ IDEA. Хоть эта программа в первую очередь рассчитана для разработки на Java, но в ней есть поддержка таких языков как JavaScript, Kotlin, Python, PHP и других. IDEA имеет ряд преимуществ:

– по ходу написания кода программа анализирует его и предлагает, как может код быть дополнен, решены существующие ошибки;

– поддержка широкого круга фреймворков и платформ, а именно Spring Framework, Node.js, AngularJS, Java EE;

– интеграция с системами контроля версий, к примеру, Git, Subversion, Mercurial, CSV;

– инструменты для работы с базами данных;

– инструменты запуска тестов и анализа покрытия кода, включая JUnit, ScalaTest, Cucumber, TestNG;

– поддержка JavaFX и т.д.

Для создания графического интерфейса будет использована библиотека JavaFX.

JavaFX — платформа на основе Java обеспечивает создание мощного графического интерфейса пользователя (Graphical User Interface, GUI). Может использоваться как для создания настольных приложений так и для интернет приложений.

Архитектура «Клиент-Сервер» (также используются термины «сеть Клиент-Сервер» или «модель Клиент-Сервер») предусматривает разделение процессов предоставление услуг и отправки запросов на них на разных компьютерах в сети, каждый из которых выполняют свои задачи независимо от других.

В архитектуре «Клиент-Сервер» несколько компьютеров-клиентов (удалённые системы) посылают запросы и получают услуги от централизованной служебной машины – сервера (server – англ. «официант, обслуга»), которая также может называться хост-системой (host system, от host – англ. «хозяин», обычно гостиницы). При разработке информационной системы был использован системный структурный подход. Модели предоставляются в виде иерархично упорядоченных диаграмм.

В качестве используемой системы управления реляционными базами данных была выбрана MySQL. Она имеет открытый исходный код с моделью «клиент-сервер». MySQL является не единственной СУБД, но считается одной из самых популярных. К преимуществам данной СУБД можно отнести простоту в использовании, гибкость, относительную дешевизну, масштабируемость и производительность.

# **ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ СИСТЕМЫ АВТОМАТИЗАЦИИ РАБОТЫ КОСМЕТИЧЕСКОГО САЛОНА**

Тестирование – это процесс выполнения программы, целью которого является выявление ошибок.

Для того чтобы убедиться, что программное средство может корректно выполнять работу и устойчиво к ошибкам пользователя было проведено тестирование, отражающие все возможные исключительные ситуации.

Самая первая проверка- это проверка на правильность введенных данных авторизованных пользователей. В случае неудачи – выводится сообщение об ошибке. (Рисунок 3.1)

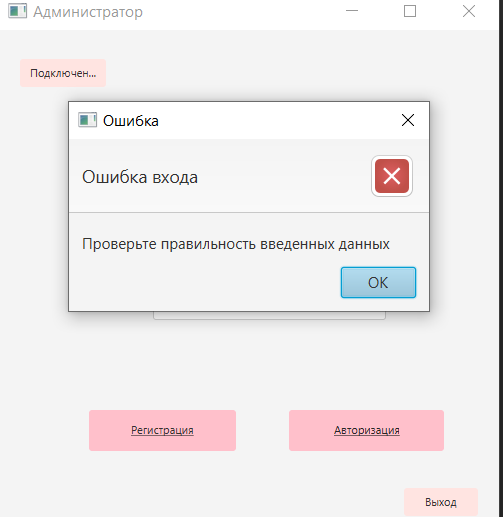


Рисунок 3.1 - Ошибка при вводе логина и пароля

Далее везде есть проверки на правильность введения номера телефона. (Рисунок 3.2).

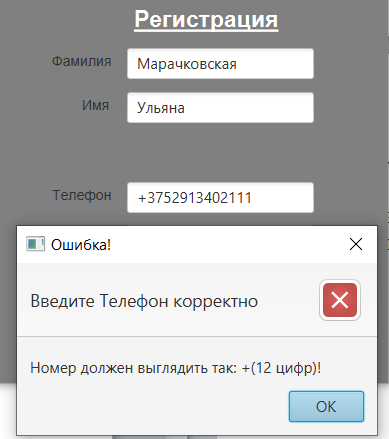


Рисунок 3.2 – Ошибка при вводе телефона

Также при вводе времени происходит проверка на корректность введения. (Рисунок 3.3).

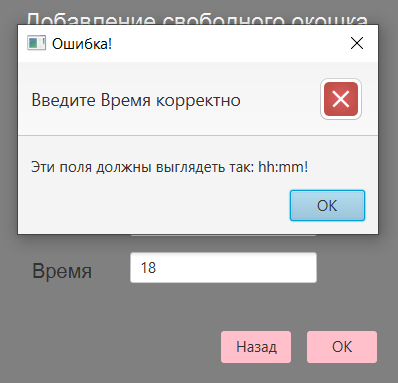


Рисунок 3.3 – Ошибка при вводе времени

При вводе даты происходит проверка на корректность введения. (Рисунок 3.4).

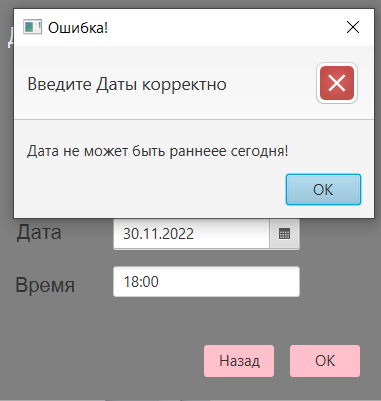


Рисунок 3.4 – Ошибка при вводе даты

При добавлении мастера нельзя использовать цифры при вводе фамилии или имени. (Рисунок 3.5).

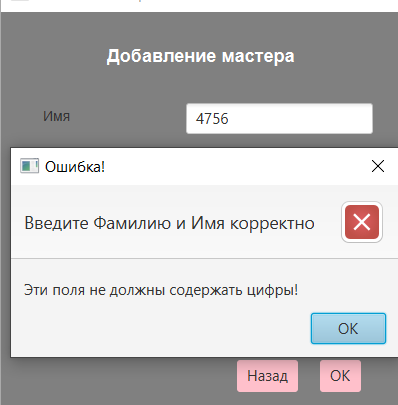


Рисунок 3.5 – Ошибка при вводе данных мастера

Зарплата и стаж мастера также должны быть введены корректно и быть больше нуля. (Рисунок 3.6).

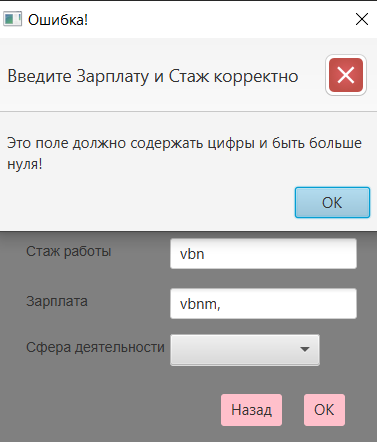


Рисунок 3.6 – Ошибка при вводе данных мастера

По результатам тестирования можно сделать вывод, что разработанное программное средство удовлетворяет функциональным требованиям и функции выполняются корректно.

# **4 РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ И ИСПОЛЬЗОВАНИЮ СИСТЕМЫ АВТОМАТИЗАЦИИ РАБОТЫ КОСМЕТИЧЕСКОГО САЛОНА**

Для запуска нашего проекта нужно поочередно запустить файлы server и main. (Рисунок 4.1)

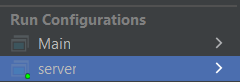


Рисунок 4.1 – Начало запуска приложения

Далее у нас появится окно входа нашего салона. (Рисунок 4.2) Нажав вверху кнопку «Подключение» происходит клиент-серверное взаимодействие, только после этого можно войти в программу.

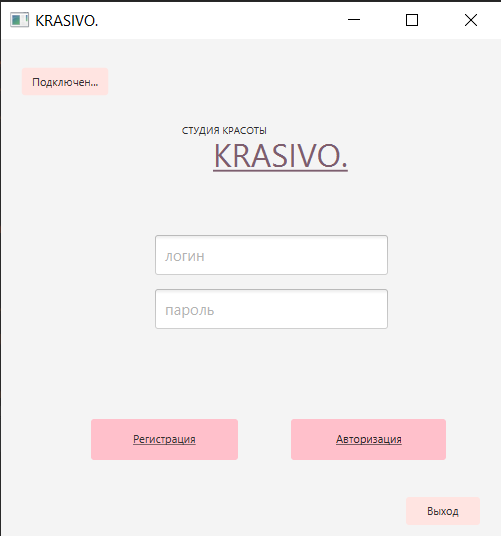


Рисунок 4.2 – Окно входа в приложение

Установка соединения с сервером изображена на рисунке 4.3.

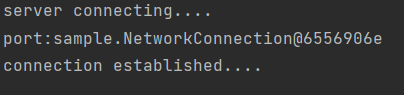


Рисунок 4.3 – Установка соединения

Далее можно пройти регистрацию для неавторизованных пользователей. (Рисунок 4.4) Либо ввести пароль и логин и оказаться в соответствующем меню.

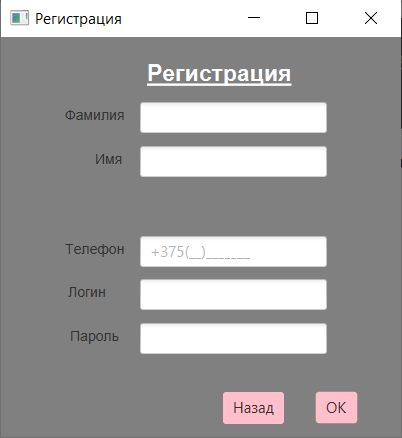


Рисунок 4.4 – Регистрация

Если же пользователь зашел под ролью администратора, то меню будет выглядеть следующим образом. Здесь можно работать с мастерами, клиентами, расписанием и услугами (Рисунок 4.5)



Рисунок 4.5 – Меню администратора

При работе с расписанием администратор может добавить свободное окошко в систему. При этом ему нужно указать название услуги из предложенного списка, фамилию мастера, дату и время свободного окошка (Рисунок 4.6)

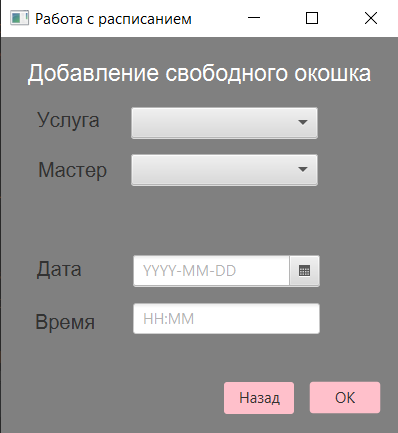


Рисунок 4.6 – Добавление свободного окошка

При работе с мастерами администратор может добавить нового мастера в систему. Для этого нужно указать фамилию, имя мастера, стаж работы, размер заработной платы и сферу деятельности. (Рисунок 4.7)

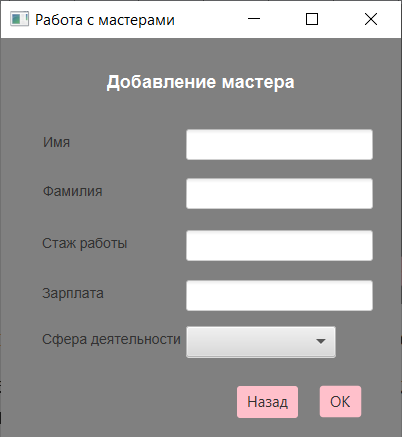


Рисунок 4.7 – Добавление нового мастера

При работе с услугами администратор может добавить новую услугу в систему. Добавить свободное окошко в расписание можно только после добавления услуги. Для добавления услуги нужно ввести название, примерную цену, сферу деятельности. (Рисунок 4.8)

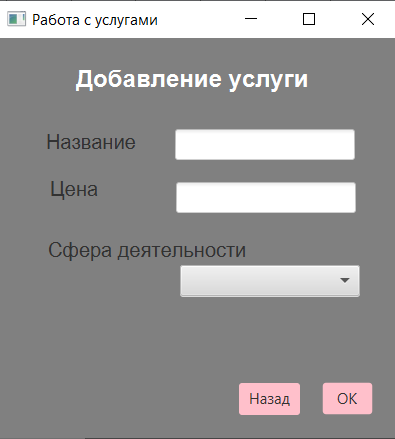


Рисунок 4.8 – Добавление новой услуги

Также администратор может записать нового клиента вручную. Но при подобной записи неавторизованный пользователь должен будет пройти регистрацию для дальнейшей записи. При добавлении клиента можно найти нужную процедуру и выбрать свободно окошко, далее ввести фамилию, имя и телефон клиента. (Рисунок 4.9)

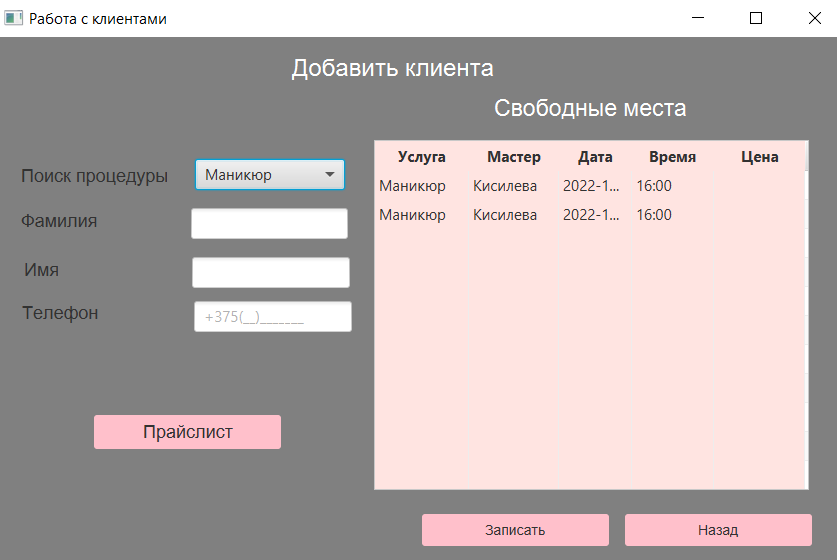


Рисунок 4.9 – Запись нового клиента

Также администратор может просмотреть прайс-лист услуг, чтобы достоверно проконсультировать клиента по цене. Отсортировать цены можно в порядке убывания и возрастания. (Рисунок 4.10)

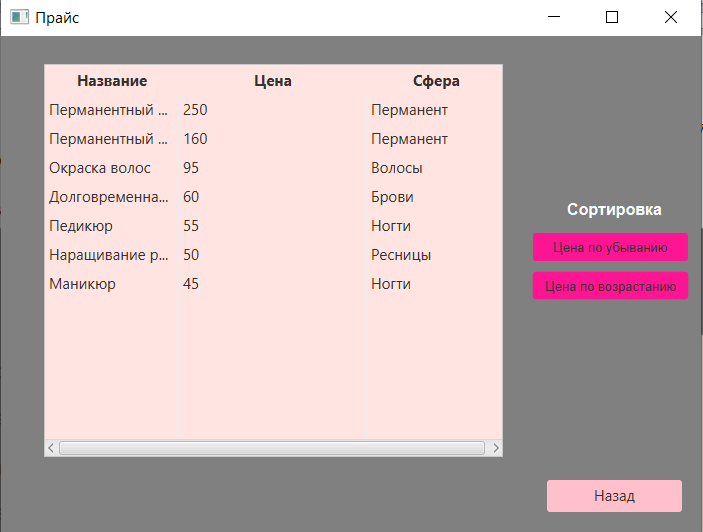


Рисунок 4.10 – Окно прайса и сортировки

Запись клиента также может быть отредактирована, для этого все данные переносятся в новые поля, где могут быть изменены. (Рисунок 4.11)

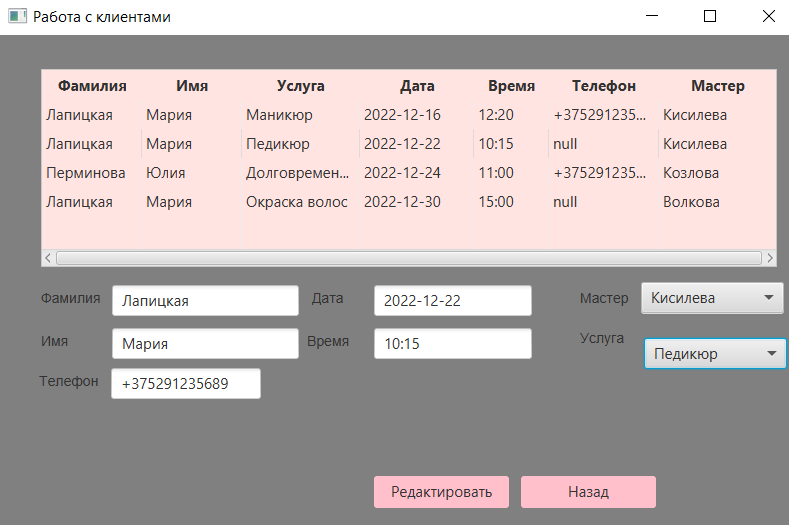


Рисунок 4.11 – Окно редактирования клиента

Также доступен просмотр расписания, клиентов, услуг и мастеров. (Рисунок 4.12)

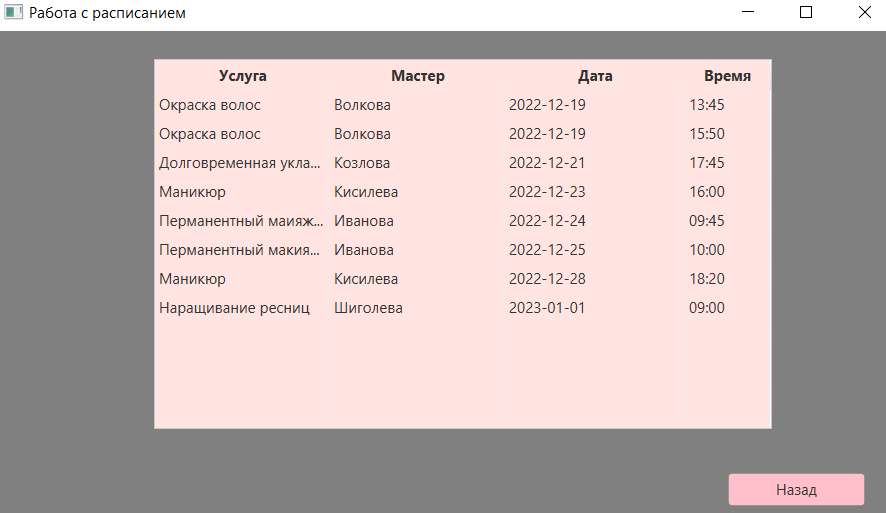


Рисунок 4.12 – Окно просмотра расписания

Для упрощения работы администратора при подведении итогов может быть создан текстовый отчет. (Рисунок 4.13)

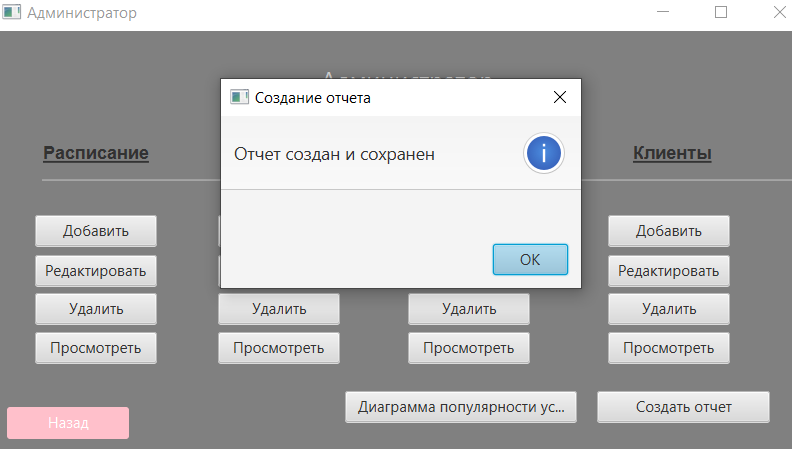


Рисунок 4.13– Окно создания отчета

Также для вывода популярности услуг была сделана диаграмма. (Рисунок 4.14)

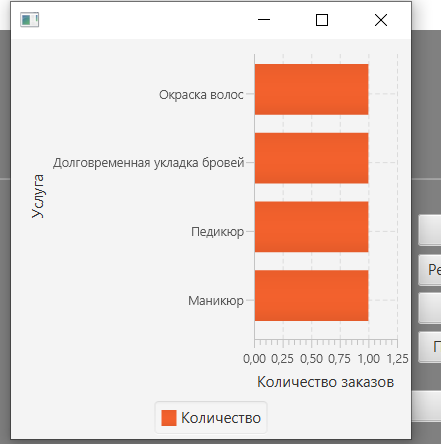


Рисунок 4.14 – Диаграмма популярности услуг

Если зайти под логином пользователя, то здесь можно записаться, просмотреть свои услуги и рассчитать свою скидку. (Рисунок 4.15)

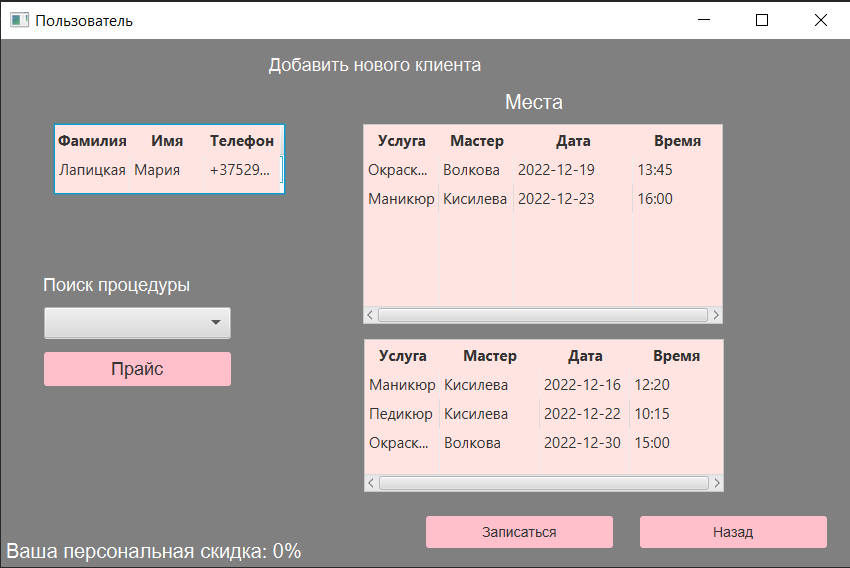


Рисунок 4.15 – Меню пользователя

Данная глава показывает корректность работы программы и то, как ее использовать.

# **Заключение**

В результате проделанной работы было создано программное средство, которое позволяет производить регистрацию и вести учёт клиентов салона красоты, что является доказательством того, что поставленная цель была достигнута.

Была изучена система работы косметического салона и рассмотрена запись клиентов в такой системе. Были освещены основные задачи, функции и структура деятельности салона. После рассмотрения предметной области появилась возможность сформировать требования к программной реализации курсового проекта. На основе требований были выбраны используемые технологии для реализации поставленных задач.

На этапе проектирования курсового проекта была изучена и применена методология IDEF0. Благодаря ей удалось полностью осветить все этапы и процессы в работе салона. Так же были разработаны диаграмма использования и диаграмма последовательности.

На этапе разработки были реализованы два компонента приложения: серверный, предназначенный для установки на локальный сервер организации, и клиентский, предназначенный для установки на рабочие компьютеры пользователей. Основные функции программного средства реализованы в соответствии поставленными перед разработкой требованиями. Для реализации поставленных задач был выбран язык программирования Java, для создания пользовательского интерфейса использовалась библиотека JavaFX, в качестве системы управления базами данных используется MySQL. Сервер поддерживает подключение к системе сразу нескольких сотрудников.

Тестирование прошло успешно, что говорит о том, что курсовая работа работает отлаженно и надёжно.

Данное программное средство создано с упором на масштабируемость. При его успешном внедрении в текущей версии, не составит труда расширить функциональные возможности приложения для автоматизации ещё некоторого количества действий.

# **Список использованных источников**

1. Ларман, К. Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ и проектирование. – 3-е изд. – СПб.: Вильямс, 2012. – 736 с.
2. Роб П., Коронел К. Системы баз данных: проектирование, реализация и управление – М: БХВ-Петербург, 2004. – 1040 с.
3. Харрингтон Джен Л. Проектирование реляционных баз данных – М: Лори, 2016. – 514 с.

[4] Перевод стандарта IDEF0 на русском языке [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/515466/

[5] Mysql [Электронный ресурс]. –­ Электронные данные. –­ Режим доступа: http://www.mysql.ru;

[6] Основы UML — диаграммы использования (use-case) [Электронный ресурс] – Режим доступа: https://pro-prof.com/archives/2594.

[7] Metanit. javafx [Электронный ресурс]. – Режим доступа: https://metanit.com/java/javafx/1.1.php.

[8] BPWin [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.bpwin.ru/.

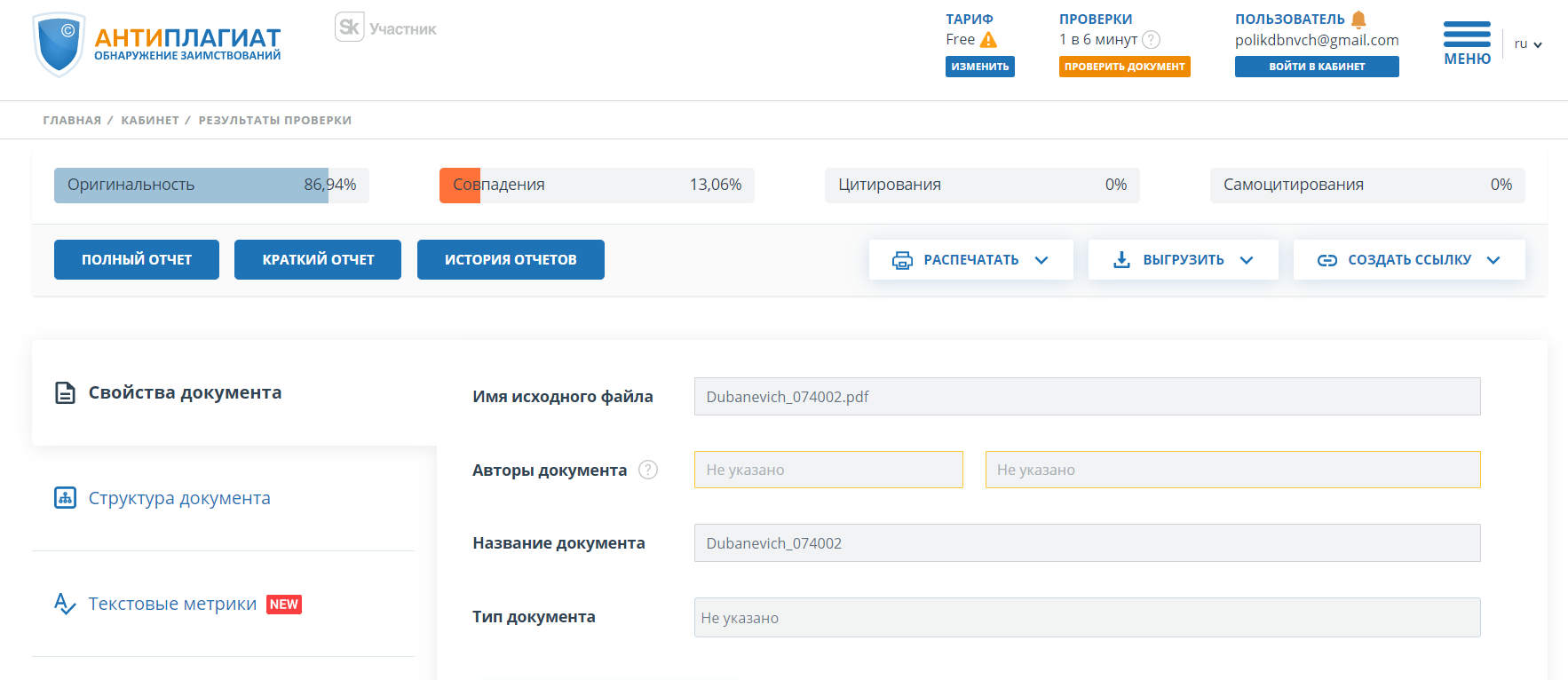
[9] Бизнес – план салона красоты [Электронный ресурс] – Режим доступа: https://www.business.ru/article/1685-biznes-plan-salona-krasoty

[10] Система онлайн записи салона красоты [Электронный ресурс] – Режим доступа: https://nails-mag.ru/biznes/sistemy-zapisi-klientov/

# **ПРИЛОЖЕНИЕ А**

**(обязательное)**

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



# **Приложение Б**

# **(обязательное)**

# **Листинг кода алгоритмов, реализующих бизнес-логику**

public static void printInf() throws SQLException {  
  
 Connection con = DriverManager.*getConnection*(*HOST*, *ROOT*, *PASSWORD*);  
  
 List<Client> informationClients = new ArrayList();  
 List<Shedule> informationShedule = new ArrayList();  
 List<Service> informationServices = new ArrayList();  
 List<Master> informationMasters = new ArrayList();  
 List<User> informationUsers = new ArrayList();  
 List<Admin> informationAdmin = new ArrayList();  
  
 try (FileWriter writer = new FileWriter("print.doc", false)) {  
 Statement stmt = con.createStatement();  
 ResultSet rs = stmt.executeQuery("select \* from Client;");  
 while (rs.next()) {  
 informationClients.add(new Client(rs.getInt("IdClient"), rs.getString("Surname"), rs.getString("Name"),  
 rs.getString("Service"), rs.getString("MasterName"), rs.getString("date"), rs.getString("time"),  
 rs.getString("Phone")));  
 }  
 rs.close();  
 stmt.close();  
  
 stmt = con.createStatement();  
 rs = stmt.executeQuery("select \* from shedule;");  
 while (rs.next()) {  
 informationShedule.add(new Shedule(rs.getInt("IdShedule"), rs.getString("ServiceName"),  
 rs.getString("MasterName"), rs.getString("Date"),  
 rs.getString("Time")*/\*, rs.getInt("IdMaster")\*/*));  
 }  
 rs.close();  
 stmt.close();  
  
 stmt = con.createStatement();  
 rs = stmt.executeQuery("select \* from Service;");  
 while (rs.next()) {  
 informationServices.add(new Service(rs.getInt("IdService"), rs.getString("Name"),  
 rs.getString("Price"),  
 rs.getString("Field")*/\*, rs.getInt("IdMaster")\*/*));  
 }  
 rs.close();  
 stmt.close();  
  
 stmt = con.createStatement();  
 rs = stmt.executeQuery("select \* from Master;");  
 while (rs.next()) {  
 informationMasters.add(new Master(rs.getInt("idMaster"), rs.getString("Name"),  
 rs.getString("Surname"), rs.getString("Experience"),  
 rs.getString("Salary"), rs.getString("Field")));  
 }  
 rs.close();  
 stmt.close();  
  
 stmt = con.createStatement();  
 rs = stmt.executeQuery("select \* from Admin;");  
 while (rs.next()) {  
 informationAdmin.add(new Admin(rs.getString("Login"),  
 rs.getString("Password")));  
 }  
 rs.close();  
 stmt.close();  
  
 stmt = con.createStatement();  
 rs = stmt.executeQuery("select \* from User;");  
 while (rs.next()) {  
 informationUsers.add(new User(rs.getString("Login"),  
 rs.getString("Password")));  
 }  
 rs.close();  
 stmt.close();

static final int *DEFAULT\_PORT* = 8181;  
 static final String *ROOT* = "root";  
 static final String *PASSWORD* = "1234";  
 static final String *HOST* = "jdbc:mysql://localhost:3306/beauty?serverTimezone=Europe/Moscow&useSSL=false&allowPublicKeyRetrieval=true";  
  
 private static Socket *clientDialog*;  
  
 public server(Socket client) {  
 server.*clientDialog* = client;  
 }  
  
 static ExecutorService *executeIt* = Executors.*newFixedThreadPool*(2);  
 static ObjectInputStream *sois* = null;  
 static ObjectOutputStream *soos* = null;  
  
 public static void main(String[] args) {  
*//* try (ServerSocket server = new ServerSocket(*DEFAULT\_PORT*);  
 BufferedReader br = new BufferedReader(new InputStreamReader(System.*in*))) {  
 while (!server.isClosed()) {  
 if (br.ready()) {  
 String serverCommand = br.readLine();  
 if (serverCommand.equalsIgnoreCase("quit")) {  
 System.*out*.println("Main Server initiate exiting...");  
 server.close();  
 break;  
 }  
 }

*executeIt*.execute(new MonoThreadClientHandler(client));  
 }  
 *executeIt*.shutdown();  
 } catch (IOException e) {  
 System.*out*.print("Номер порта введен неверно");  
 e.printStackTrace();  
 }  
}

public class NetworkConnection {  
  
 public static Socket *clientSocket*;  
 public static ObjectOutputStream *coos*;  
 public static ObjectInputStream *cois*;  
  
  
 public void startConnection(int port) {  
  
 System.*out*.println("server connecting....");  
 try {  
  
 System.*out*.println("port:" + Controller.*connection*);  
 *clientSocket* = new Socket("127.0.0.1", port);  
 } catch (IOException e) {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Ошибка");  
 alert.setHeaderText("Ошибка подключения");  
 alert.setContentText("Проверьте работу сервера");  
 alert.showAndWait();  
 e.printStackTrace();  
 }  
 System.*out*.println("connection established....");  
 try {  
 *coos* = new ObjectOutputStream(*clientSocket*.getOutputStream());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 try {  
 *cois* = new ObjectInputStream(*clientSocket*.getInputStream());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
  
 public static List<Object> send(Object obj) throws Exception {  
 *coos*.writeObject(obj);  
 List<Object> list = (List<Object>) *cois*.readObject();  
 return list;  
 }  
  
 public void closeConnection() throws Exception {  
 *cois*.close();  
 *coos*.close();  
 *clientSocket*.close();  
 }  
}

public boolean ValidEnter(){  
 if(text\_surname.getText().matches("[\\d]+")|text\_name.getText().matches("[\\d]+")|  
 text\_surname.getText().matches("[\\s]+")|text\_name.getText().matches("[\\s]+"))  
 return true;  
 else return false;  
 }  
 public boolean ValidSalary(){  
 float salary=0,experience=0;  
 try {  
 salary= Float.*parseFloat*(text\_salary.getText());  
 System.*out*.println("SALARY "+salary);  
 experience=Integer.*parseInt*(text\_experience.getText());  
 }catch (NumberFormatException nfe){System.*out*.println("ОШИБКА");return false;}  
 return true;  
 }  
 public void AlertEnter(){  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Ошибка!");  
 alert.setHeaderText("Введите Фамилию и Имя корректно");  
 alert.setContentText("Эти поля не должны содержать цифры!");  
 alert.showAndWait();  
 }  
 public void AlertSalary(){  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Ошибка!");  
 alert.setHeaderText("Введите Зарплату и Стаж корректно");  
 alert.setContentText("Это поле должно содержать цифры и быть меньше нуля!");  
 alert.showAndWait();  
 }  
}

private void setButtonRemove(ActionEvent event) throws IOException {  
 Shedule shedule = new Shedule(clickedRow.getId(), "removeShedule");  
 List<Object> res = new ArrayList<Object>();  
 try {  
 res = (*send*(shedule));  
 } catch (IOException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
  
 if(Boolean.*parseBoolean*(res.get(0).toString())){  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle("Удаление места");  
 alert.setHeaderText("Место удалено успешно");  
 alert.showAndWait();  
 initialize();  
  
 } else {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Удаление места");  
 alert.setHeaderText("Место не удалено");  
 alert.showAndWait();  
 }  
 }  
}

private static final List<DiscountInfo> *NUMBER\_OF\_VISITORS\_FOR\_DISCOUNT* = Arrays.*asList*(  
 new DiscountInfo(0, 5,0),  
 new DiscountInfo(6, 10,1),  
 new DiscountInfo(11, 25,2),  
 new DiscountInfo(26, 50,5));

public List<Object> editClient(Connection con, Client client) {  
 List<Object> list = new ArrayList();  
 try {  
 PreparedStatement st = con.prepareStatement("DELETE FROM Client WHERE IdClient = " + client.getId() + ";");  
 st.executeUpdate();  
 st.close();  
 Statement stmt = con.createStatement();  
 String str\_update = "insert into Client(IdClient,Surname,Name, Service, MasterName,Date,Time, Phone) values (" +  
 getId() + ",'" + client.getFirstname() + "', '" + client.getName() + "', '" + client.getService() + "', '" +  
 client.getMasterName() + "', '" +  
 client.getDate() + "', '" + client.getTime() + "', '" + client.getPhone() + "')";  
 stmt.executeUpdate(str\_update);  
 stmt.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 con.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 list.add("true");  
 return list;  
}

private void setbuttonCreateReport(ActionEvent event) throws IOException {  
 Client client = new Client("getReport");  
 List<Object> res = new ArrayList<Object>();  
  
 try {  
 res = (*send*(client));  
 System.*out*.println(res.get(0).toString());  
 } catch (IOException e) {  
 e.printStackTrace();  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
  
 if(Boolean.*parseBoolean*(res.get(0).toString())){  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle("Создание отчета");  
 alert.setHeaderText("Отчет создан и сохранен");  
 alert.showAndWait();  
 } else {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle("Создание отчета");  
 alert.setHeaderText("Ошибка создания отчета");  
 alert.showAndWait();  
 }  
}

# **ПРИЛОЖЕНИЕ В**

**(обязательное)**

# **Листинг скрипта генерации базы данных**

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`service`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`service` (

`idservice` INT NOT NULL,

`NameS` VARCHAR(45) NULL,

`Price` VARCHAR(45) NULL,

`Field` VARCHAR(45) NULL,

PRIMARY KEY (`idservice`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`client`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`client` (

`idclient` INT NOT NULL,

`Surname` VARCHAR(45) NULL,

`Name` VARCHAR(45) NULL,

`Service` VARCHAR(45) NULL,

`MasterName` VARCHAR(45) NULL,

`date` VARCHAR(45) NULL,

`time` VARCHAR(45) NULL,

`Phone` VARCHAR(45) NULL,

`service\_idservice` INT NOT NULL,

PRIMARY KEY (`idclient`),

INDEX `fk\_client\_service1\_idx` (`service\_idservice` ASC) VISIBLE,

CONSTRAINT `fk\_client\_service1`

FOREIGN KEY (`service\_idservice`)

REFERENCES `mydb`.`service` (`idservice`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`shedule`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`shedule` (

`idshedule` INT NOT NULL,

`ServiceName` VARCHAR(45) NULL,

`MasterName` VARCHAR(45) NULL,

`Date` VARCHAR(45) NULL,

`Time` VARCHAR(45) NULL,

`clientName` VARCHAR(45) NULL,

`service\_idservice` INT NOT NULL,

PRIMARY KEY (`idshedule`),

INDEX `fk\_shedule\_service1\_idx` (`service\_idservice` ASC) VISIBLE,

CONSTRAINT `fk\_shedule\_service1`

FOREIGN KEY (`service\_idservice`)

REFERENCES `mydb`.`service` (`idservice`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`master`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`master` (

`idmaster` INT NOT NULL,

`MasterName` VARCHAR(45) NULL,

`Surname` VARCHAR(45) NULL,

`Experience` VARCHAR(45) NULL,

`Salary` VARCHAR(45) NULL,

`field` VARCHAR(45) NULL,

`shedule\_idshedule` INT NOT NULL,

PRIMARY KEY (`idmaster`),

INDEX `fk\_master\_shedule1\_idx` (`shedule\_idshedule` ASC) VISIBLE,

CONSTRAINT `fk\_master\_shedule1`

FOREIGN KEY (`shedule\_idshedule`)

REFERENCES `mydb`.`shedule` (`idshedule`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`admin`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`admin` (

`IdAdmin` INT NOT NULL,

`master\_idmaster` INT NOT NULL,

`service\_idservice` INT NOT NULL,

`status` VARCHAR(45) NULL,

PRIMARY KEY (`IdAdmin`),

INDEX `fk\_admin\_master1\_idx` (`master\_idmaster` ASC) VISIBLE,

INDEX `fk\_admin\_service1\_idx` (`service\_idservice` ASC) VISIBLE,

CONSTRAINT `fk\_admin\_master1`

FOREIGN KEY (`master\_idmaster`)

REFERENCES `mydb`.`master` (`idmaster`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_admin\_service1`

FOREIGN KEY (`service\_idservice`)

REFERENCES `mydb`.`service` (`idservice`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`user`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `mydb`.`user` (

`iduser` INT NOT NULL,

`Login` VARCHAR(45) NULL,

`Password` VARCHAR(45) NULL,

`client\_idclient` INT NOT NULL,

`admin\_IdAdmin` INT NOT NULL,

PRIMARY KEY (`iduser`, `client\_idclient`, `admin\_IdAdmin`),

INDEX `fk\_user\_client\_idx` (`client\_idclient` ASC) VISIBLE,

INDEX `fk\_user\_admin1\_idx` (`admin\_IdAdmin` ASC) VISIBLE,

CONSTRAINT `fk\_user\_client`

FOREIGN KEY (`client\_idclient`)

REFERENCES `mydb`.`client` (`idclient`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_user\_admin1`

FOREIGN KEY (`admin\_IdAdmin`)

REFERENCES `mydb`.`admin` (`IdAdmin`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;