Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина средства и технологии анализа и разработки информационных систем

|  |
| --- |
| *К защите допустить*: |
| Руководитель курсового проекта |
| \_\_\_\_\_\_\_\_\_\_\_Ю. В. Сильванович  « \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_ 2023 г. |

**Пояснительная записка**к курсовому проекту  
на тему

**ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АВТОМАТИЗАЦИИ ОСНОВНЫХ БИЗНЕС-ПРОЦЕССОВ ПАРИКМАХЕРСКОЙ**

БГУИР КП 1-40 05 01-02 012 ПЗ

|  |
| --- |
| Выполнила студентка группы 072304  Ладкина Мария-Ирина Ильинична  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
| Курсовая работа представлена на проверку « \_\_\_\_ » \_\_\_\_\_\_\_\_\_ 2023 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**РЕФЕРАТ**

БГУИР КП 1-40 05 01-02 012 ПЗ

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АВТОМАТИЗАЦИИ основных бизнес-процессов парикмахерской: курсовой проект / М-И. И. Ладкина. – Минск: БГУИР, 2023, – п. з. 89 с., чертежей (плакатов) – 5 л. формата А4.

Пояснительная записка 89 с., 54 рис., 14 источников, 5 приложений.

**Ключевые слова:** парикмахерская, веб-сервер, веб-сервис, база данных, модели системы, модель доступа к данным, администратор, сотрудник, клиент, разработка программного средства.

*Целью курсового проекта* является совершенствование и оптимизация процесса работы со складским учетом, а также автоматизации бизнес-процессов, непосредственно связанных с управлением парикмахерской (отслеживание прибыли, подача заявок поставщикам и др.), путем разработки программного средства для автоматизации работы парикмахерской.

*Объект исследования* – парикмахерская и системы автоматизации работы парикмахерской.

*Предмет исследования* – программное средство для автоматизации работы парикмахерской.

*Актуальность работы:* программное средство для автоматизации работы парикмахерской требует углублений в рамках мобильности его использования и наличия возможности формирования документации и статистики. Для таких задач целесообразна автоматизация работы парикмахерской.

*Методология проведения работы*: в процессе разработки системы использованы методы анализа данных, современные подходы к обработке данных, функциональный анализ процессов, принципы построения баз данных, моделирование системы с помощью UML-диаграмм.

*Результаты работы*: создано web-приложение, являющееся программным средством осуществления работы парикмахерской. Проанализированы основные бизнес-процессы предметной области. Разработан интуитивно понятный интерфейс программного продукта, не вводящий пользователя в заблуждение. Техническое обоснование разработки программного модуля показало целесообразность его внедрения.

**Содержание**

[Перечень условных обозначений, символов и терминов 6](#_Toc118988508)

[Введение 8](#_Toc118988509)

[1 Анализ и моделирование предметной области программного средства 11](#_Toc118988510)

[1.1 Описание предметной области 11](#_Toc118988511)

[1.2 Разработка функциональной модели предметной области 17](#_Toc118988512)

[1.3 [Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 25](#_Toc118988524)](#_Toc118988513)

[1.4 Разработка информационной модели предметной области 29](#_Toc118988513)

[1.5 UML-модели представления программного средства и их описание](#_Toc118988513) 32

2 Проектирование и конструирование программного средства 43

[2.1 Постановка задачи](#_Toc118988517) 43

[2.2 Обоснование выбора компонентов и технологий для реализации программного средства 44](#_Toc118988518)

[2.3 Архитектурные решения 45](#_Toc118988518)

[2.4 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства 46](#_Toc118988519)

[2.5 Проектирование пользовательского интерфейса](#_Toc118988523) 48

[2.6 Методы и средства, используемые для обеспечения безопасности данных](#_Toc118988523) 50

[3 Тестирование и проверка работоспособности программного средства 51](#_Toc118988525)

[4 Руководство по развертыванию и использованию программного средства 54](#_Toc118988526)

[4.1 Руководство по установке (развертыванию) программного средства 54](#_Toc118988524)

[4.2 Руководство пользователя 54](#_Toc118988524)

[Заключение 68](#_Toc118988527)

[Список использованных источников 69](#_Toc118988528)

[Приложение А (обязательное) Отчет о проверке на заимствования в системе «Антиплагиат» 70](#_Toc118988531)

[Приложение Б (обязательное) Листинг скрипта генерации базы данных 71](#_Toc118988532)

[Приложение В (обязательное) Листинг кода алгоритмов, реализующих основную бизнес-логику 73](#_Toc118988536)

[Приложение Г (обязательное) Схемы алгоритмов работы системы 86](#_Toc118988535)

[Приложение Д (обязательное) Диаграмма классов 88](#_Toc118988536)

[Ведомость документов курсового проекта 89](#_Toc118988537)

# Перечень условных обозначений, символов и терминов

|  |  |
| --- | --- |
| *ТЗ (техническое задание)*  *ПС (парикмахерский салон)*  *Веб-сервер* | – документ, содержащий требования заказчика к объекту разработки, определяющий порядок и условия её проведения;  – это предприятие, занимающееся предоставлением услуг для населения по уходу за волосами в оборудованном специально для этого помещении. Как правило, в парикмахерских дополнительно оказываются следующие виды услуг: маникюр, педикюр, косметические услуги и услуги визажиста;  – сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными; |
| *Среда выполнения*  *БД (база данных)* | – вычислительное окружение, необходимое для выполнения компьютерной программы и доступное во время выполнения компьютерной программы;  – представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины; |
| *СУБД (система управления базами данных)* | – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных; |
| *Маркетплейс*  *GUI (Graphical User Interface)* | – особый вид торговой площадки, куда выставляются товары от разных продавцов. По сути, маркетплейс – всего лишь посредник, который осуществляет связь покупателя и продавца;  – графический интерфейс пользователя – это разновидность пользовательского интерфейса, в котором все элементы (кнопки, меню, пиктограммы, списки) представленные пользователю на дисплее, выполнены в виде картинок, графики; |
| *UML* (*Unified Modeling Language*)  *API* (*application programming interface*) | – язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур;  – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой; |
| *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; |
| *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. |

**ВВЕДЕНИЕ**

Современный мир стремительно меняется, в результате чего современные технологии стали играть важную роль в различных сферах человеческой деятельности. И парикмахерские салоны не стали исключением из этого правила.

Парикмахерская – это место, куда каждый день сотни клиентов приходят, чтобы изменить свой имидж. Но при этом, управление бизнесом стало более сложным и требует от салона не только высококвалифицированных специалистов, но и использования новых технологий для эффективного управления своими процессами.

В связи с этим, программное средство для автоматизации основных бизнес-процессов парикмахерской становится важной составляющей в управлении салоном и его успешности. Это сочетание аппаратно-программного обеспечения, которое помогает владельцам парикмахерских автоматизировать бизнес-процессы, сократить временные затраты на рутинные операции, повысить качество услуг и удовлетворение клиентов.

Программное средство для автоматизации парикмахерской может быть разработано под различные задачи и нужды салона. Например, программа может помочь управлять расписанием работы сотрудников, контролировать запасы товаров и инвентаря, анализировать финансовые отчеты и т.д.

Кроме того, такое программное обеспечение позволяет сэкономить время на поиске и учете информации, ведение книги учета, управление базой данных клиентов и многое другое. В результате парикмахерский салон может сосредоточиться на своем основном бизнесе – предоставлении качественных услуг клиентам, не тратя силы и время на мелкие, но важные рутинные операции.

Таким образом, применение программного средства для автоматизации основных бизнес-процессов парикмахерской – это эффективный способ управления салоном, повышения качества услуг, удовлетворения клиентов и привлечения новых клиентов. Автоматизация бизнес-процессов с помощью программного обеспечения, теперь является необходимостью для современных парикмахерских салонов. Это инвестиция, которая выигрывает со временем, приносит выгоду и помогает оставаться конкурентоспособным на рынке.

Актуальность работы: программное средство для автоматизации работы парикмахерской требует углублений в рамках мобильности его использования и наличия возможности формирования документации и статистики. Для таких задач целесообразна автоматизация работы парикмахерской.

Объект исследования – парикмахерская и системы автоматизации работы парикмахерской.

Предмет исследования – программное средство для автоматизации работы парикмахерской.

Целью данного курсового проекта является совершенствование и оптимизация процесса работы со складским учетом, а также автоматизации бизнес-процессов, непосредственно связанных с управлением парикмахерской (отслеживание прибыли, подача заявок поставщикам и др.), путем разработки программного средства для автоматизации работы парикмахерской.

Поставленная цель потребовала решения следующих задач:

* сделать анализ и модель предметной области реализации товаров и услуг в парикмахерских салонах;
* выволнить проектирование и конструирование программного средства для автоматизации работы парикмахерской;
* провести тестирование и проверку работоспособности программного средства для автоматизации работы парикмахерской;
* предоставить руководство по развертыванию и использованию программного средства для автоматизации работы парикмахерские салона пользователем.

Курсовой проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности составляет 94,13%, что соответствует норме, установленной кафедрой экономической информатики (приложение А, рисунок А.1). Цитирования обозначены ссылками на публикации, указанные в «Списке использованных источников».

# Анализ и моделирование предметной области программного средства

Парикмахерская – это очень популярный вид деятельности. Сейчас все больше людей начинают обращать внимание на свою внешность и ухоженность, поэтому услуги парикмахеров становятся все более востребованными. Однако, как и в любом другом бизнесе, существует огромная конкуренция. Чтобы выделиться на рынке и привлечь большое количество клиентов, необходимо иметь качественное программное средство.

* 1. **Описание предметной области**

Основная цель парикмахерского салона – это предоставление высококачественных услуг и удовлетворение потребностей клиентов в уходе за волосами. Эта цель может быть достигнута только благодаря комплексному подходу к управлению салоном.

В области предоставляемых услуг можно выделить несколько основных направлений:

* стрижка и укладка волос;
* окрашивание и тонирование волос;
* уход за волосами и кожей головы;
* наращивание волос и наращивание ресниц;
* маникюр и педикюр.

Каждое из направлений имеет свои особенности и требует от парикмахеров и мастеров высокого уровня профессионализма и квалификации.

Стрижка и укладка волос включает в себя широкий спектр услуг, начиная от классических стрижек и причесок, заканчивая модными укладками и прическами для особых случаев. В рамках данного направления парикмахеры должны уметь правильно подбирать стрижки и укладки в зависимости от формы лица и типа волос клиента.

Окрашивание и тонирование волос позволяет изменить цвет и оттенок волос, придавая им яркость и насыщенность. Кроме того, окрашивание позволяет скрыть седину и исправить недостатки цвета волос. Важно учитывать особенности клиента, его цветотип и цвет глаз, чтобы подобрать наиболее подходящий цвет волос.

Уход за волосами и кожей головы включает в себя широкий спектр процедур, таких как массаж головы, глубокое увлажнение и питание волос, а также уход за кожей головы. Парикмахеры должны знать основные принципы ухода за волосами и кожей головы, чтобы правильно подбирать процедуры и средства для каждого клиента.

Наращивание волос и ресниц – это процедуры, которые позволяют изменить длину и объем волос и ресниц. В рамках данного направления парикмахеры должны уметь работать с различными материалами и технологиями, чтобы обеспечить наилучший результат для клиента.

Маникюр и педикюр – это процедуры по уходу за ногтями на руках и ногах. Мастера парикмахерского салона должны обладать навыками и знаниями в области маникюра и педикюра, включая процедуры очистки ногтей, удаления кутикулы, обработки кожи рук и ног, а также нанесение лака и украшений.

Далее мы рассмотрим организационная структура парикмахерского салона.

Организационная структура парикмахерского салона состоит из:

* руководство;
* финансовый отдел;
* отдел развития;
* административный отдел.

Руководство парикмахерского салона является высшим управленческим органом компании. Они устанавливают стратегические цели и направления развития бизнеса, а также контролируют и координируют работу всех отделов. Они принимают ключевые решения, связанные с финансами, управлением ресурсами, маркетингом и другими важными аспектами деятельности компании. Руководство также отвечает за управление рисками и принятие мер по предотвращению потенциальных проблем.

Финансовый отдел парикмахерского салона отвечает за управление финансами компании. Они занимаются составлением бюджета, учетом доходов и расходов, финансовым анализом и подготовкой финансовых отчетов. Финансовый отдел также контролирует расходы, управляет налогами, планирует инвестиции и управляет финансовыми рисками. Они помогают руководству компании принимать обоснованные решения, связанные с финансовыми вопросами.

Отдел развития парикмахерского салона занимается исследованием рынка, конкурентов и клиентов, анализирует тренды и прогнозирует будущие изменения в отрасли. Они разрабатывают стратегии развития бизнеса, планируют новые услуги, исследуют потребности клиентов и укрепляют имидж бренда. Отдел развития также отвечает за оценку эффективности новых услуг, разработку программ лояльности и маркетинговых акций для привлечения новых клиентов.

Краткая организационная структура парикмахерского салона проиллюстрирована ниже, на рисунке 1.1.



Рисунок 1.1 – Краткая организационная структура парикмахерского салона

Административный отдел парикмахерского салона занимается управлением ресурсами компании. Они отвечают за управление кадрами, закупки, обеспечение безопасности и управление инфраструктурой. Административный отдел также отвечает за разработку и внедрение политик и процедур, направленных на улучшение эффективности работы компании и повышение качества предоставляемых услуг. Они также обеспечивают своевременное обслуживание клиентов и управляют процессом бронирования.

Каждый из этих отделов играет важную роль в успешном функционировании парикмахерского салона. Они работают в тесном взаимодействии друг с другом, чтобы обеспечить качество услуг и удовлетворение потребностей клиентов, а также эффективное управление бизнесом.

Далее сделаем детальный обзор всех основных бизнес-процессов парикмахерского, которые обычно выполняются в рамках работы парикмахерского салона. Парикмахерский салон – это не только место, где клиенты могут получить услуги по уходу за волосами и ногтями, но и бизнес, который требует эффективного управления и оптимизации процессов для достижения успеха и прибыли. Поэтому мы рассмотрим наиболее важные бизнес-процессы и опишем их детально, чтобы помочь владельцам парикмахерских салонов и их менеджерам улучшить эффективность своих операций и увеличить прибыль. На рисунке 1.2 приведена схема основных бизнес-процессов парикмахерского салона.



Рисунок 1.2 – Схема основных бизнес-процессов парикмахерского салон

Управление персоналом:

* рекрутинг и найм персонала: создание вакансий, публикация вакансий, оценка резюме, проведение интервью, выбор подходящего кандидата и оформление документов;
* обучение и развитие персонала: обучение новых сотрудников, проведение тренингов и семинаров для развития навыков, подготовка к повышению квалификации;
* оценка и управление производительностью: оценка производительности сотрудников, установление целей и планирование достижения результатов, разработка и внедрение системы стимулирования.

Управление закупками и инвентаризацией:

* закупки материалов и оборудования: оценка потребностей в материалах и оборудовании, поиск поставщиков, сравнение цен и условий, оформление заказов;
* управление инвентаризацией: учет и контроль за запасами, периодическая инвентаризация, решение проблем, связанных с нехваткой или переизбытком материалов и оборудования.

Управление клиентами:

* управление бронированием и записью на услуги: прием заявок на бронирование и запись на услуги, управление расписанием, уведомление клиентов об изменениях;
* организация обслуживания клиентов: прием и организация обслуживания клиентов, учет предпочтений и пожеланий клиентов, оказание дополнительных услуг.
* управление обратной связью: сбор и анализ отзывов клиентов, поиск возможностей для улучшения качества обслуживания, удовлетворение потребностей клиентов.

Проведение услуг:

* проведение стрижек и укладок: выполнение различных видов стрижек и укладок в соответствии с желаниями клиентов, рекомендации по уходу за волосами;
* проведение косметических процедур: проведение различных процедур для ухода за кожей, ногтями, волосами, подбор индивидуальных средств ухода;
* услуги массажа и и SPA-процедуры: предоставление услуг массажа для тела, лица, рук и ног, а также проведение SPA-процедур для расслабления и улучшения состояния кожи.

Финансовый учет:

* управление кассой: включает в себя регистрацию всех операций, проводимых через кассу салона, таких как продажа услуг, продажа товаров, оплата аренды и других расходов;
* управление бухгалтерскими записями: включает в себя ведение документации, отражающей все финансовые операции, произведенные в салоне, такие как счета-фактуры, договоры и другие документы;
* финансовый анализ: позволяет проанализировать финансовые результаты деятельности салона, оценить эффективность использования ресурсов и определить направления для повышения прибыли.

Все основные бизнес-процессы парикмахерского салона направлены на предоставление клиентам качественных и профессиональных услуг в области ухода за внешностью и создания модных образов. Они включают в себя все этапы, начиная от приема и консультации клиентов и заканчивая оказанием услуг и финансовым учетом.

Процессы, связанные с приемом и консультацией клиентов, играют важную роль в создании доверительных отношений между салоном и клиентом. Это важно, так как хорошие взаимоотношения с клиентами позволяют салону не только привлекать новых клиентов, но и удерживать постоянную клиентуру.

Процессы, связанные с выполнением услуг, определяют качество и эффективность работы парикмахерского салона. Профессиональные навыки и опыт сотрудников салона позволяют обеспечить наилучший результат для клиента, что повышает уровень удовлетворенности клиентов и увеличивает вероятность их возвращения.

Финансовые процессы также играют важную роль в успешной работе парикмахерского салона. Управление кассой и бухгалтерскими записями позволяет контролировать финансовые потоки и снижать риски возникновения ошибок в учете, что в свою очередь позволяет салону экономить время и ресурсы. Финансовый анализ позволяет управлять финансовыми ресурсами салона и принимать решения, направленные на повышение эффективности бизнеса и увеличение прибыли.

В целом, эффективное управление всеми бизнес-процессами парикмахерского салона позволяет салону успешно конкурировать на рынке, привлекать новых клиентов и удерживать постоянную клиентуру, повышать уровень удовлетворенности клиентов и обеспечивать стабильный доход.

Но, а чтобы привлекать новых клиентов и поддерживать связь с существующими парикмахерские салону необходимо имеют свой веб-сайт. Поэтому далее будет проведен сравнительный анализ веб-сайтов парикмахерских салонов, которые представлены на рынке.

Сравнительный анализ веб-сайтов парикмахерских салонов:

1. Сайт "Hair&Beauty":
   * удобный и простой интерфейс;
   * интерактивные элементы (калькулятор цен, онлайн-запись и т.д.);
   * недостаточно информации о процедурах и услугах.
2. Сайт "HairStyle":
   * красивый и стильный дизайн;
   * информативные статьи и блог;
   * неудобный поиск информации о ценах и услугах;
   * отсутствие онлайн-записи.
3. Сайт "Мастерская красоты":
   * подробное описание процедур и услуг;
   * удобная онлайн-запись;
   * старомодный и устаревший дизайн;
   * медленная скорость загрузки страниц.
4. Сайт "HairCut":
   * интерактивная онлайн-консультация;
   * подробные отзывы и рейтинги мастеров;
   * сложный и запутанный интерфейс;
   * ограниченный функционал.

Наиболее успешным решением для автоматизации основных бизнес-процессов парикмахерского салона является разработка индивидуального программного средства, учитывающего все плюсы и минусы существующих программных средств. Это средство должно включать в себя удобный и понятный интерфейс, подробное описание услуг и процедур, удобный поиск информации о ценах и услугах, онлайн-запись и интерактивные элементы. Также важным фактором является быстрая скорость загрузки страниц и простота навигации.

Результатом исследования стал разбор предметной области ведения парикмахерского салона; выбор сферы деятельности парикмахерского салона; проведен сравнительный анализ функционала аналогов, с разбором плюсов и минусов каждого, и вследствие, выбор необходимых модулей для создаваемого программного средства.

Начало формы

Конец формы

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

Исходя из задания данного курсового проекта, целью моделирования системы является описание ее функциональности для дальнейшего использования созданной модели при разработке информационной модели.

Методология IDEF0 (Integration Definition for Function Modeling) - это инструмент, используемый для анализа и моделирования бизнес-процессов. Она была разработана в 1970-х годах в рамках программы Integrated Computer-Aided Manufacturing (ICAM) в США и с тех пор широко используется в различных отраслях, включая промышленность, транспорт, здравоохранение и государственное управление.

Методология IDEF0 предоставляет ряд преимуществ для анализа бизнес-процессов, включая возможность учета всех функций и связей между ними, а также возможность описания процессов на разных уровнях детализации. Она также обеспечивает легкость визуализации и понимания сложных процессов, а также возможность проведения анализа и оптимизации бизнес-процессов.

Однако, методология IDEF0 также имеет свои ограничения. В частности, она может быть достаточно сложной для использования без определенного уровня подготовки и обучения. Кроме того, она может быть ограничена в возможности учета некоторых аспектов бизнес-процессов, таких как контекстуальные факторы или социальные аспекты.

В целом, методология IDEF0 является эффективным инструментом для анализа и оптимизации бизнес-процессов. Однако, для достижения наилучших результатов, необходимо сочетать ее использование с другими методами и инструментами, а также учитывать конкретные особенности бизнес-процессов, которые подлежат моделированию.

Контекстная диаграмма представляет собой общее описание системы и пути её взаимодействия с пользователями.

На рисунке 1.3 представлена контекстная диаграмма верхнего уровня с функциональной моделью «Организовать работу парикмахерского салона», а также определены потоки входных и выходных данных, механизмы ограничения и управления данными. Таким образом, исходя из потребности в оказании услуги и информации о клиенте, информация ТМЦ, информация о сотрудниках, информация о расходах и доходах, осуществляется реализация товаров и услуг; налаженные поставки ТМЦ; база данных клиентов; найм персонала и оценка работы персонала; а также выставление счета и обработка платежей.

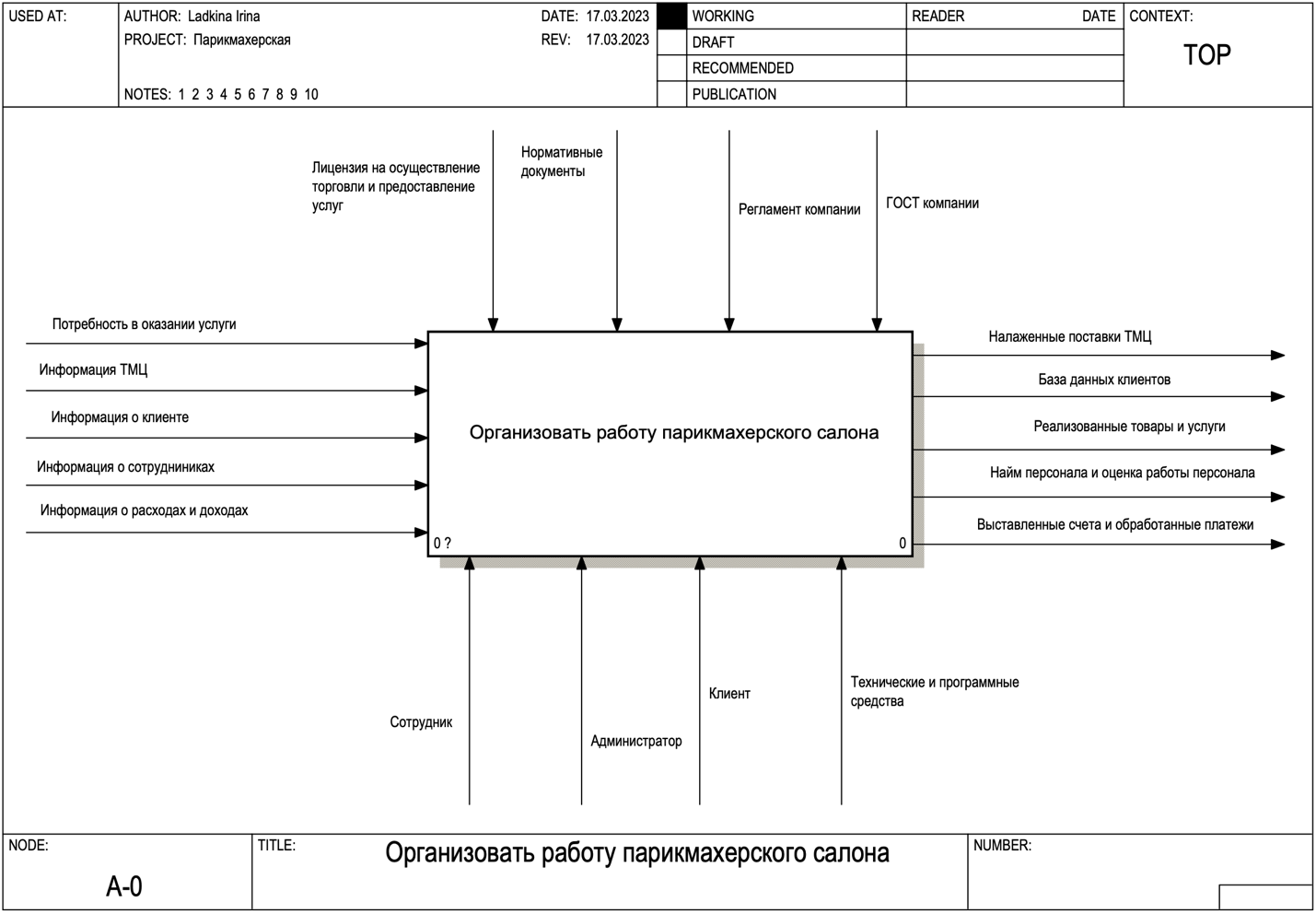


Рисунок 1.3 – Контекстная диаграмма «Организовать работу парикмахерского салона»

На рисунке 1.4 отображена декомпозиция контекстной диаграммы, состоящая из пяти главных подпроцессов: «Управлять клиентами», «Предоставить услуги», «Управлять персоналом», «Управлять инвентарем», «Управлять финансами».

Именно эти процессы, находящиеся под воздействием лицензии на осуществление торговли и предоставление услуг, ГОСТов и регламента компании и другой нормативной документации, а также управляемые, непосредственно, администрацией, сотрудниками, клиентами и программно-техническими средствами, позволяют точно определить последовательность действий для корректной деятельности парикмахерского салона.



Рисунок 1.4 – Декомпозиция контекстной диаграммы «Организовать работу парикмахерского салона»

На рисунке 1.5 отображена декомпозиция блока «Управлять персоналом», состоящая из четырех блоков: «Нанять персонал», «Организовать работу персонала», «Обучить персонал», «Оценить работу персонала».

Первый подблок, "Нанять персонал", описывает процесс подбора новых сотрудников для парикмахерского салона. Это может включать в себя планирование численности персонала, размещение объявлений о вакансиях, проведение интервью и выбор подходящих кандидатов.

Второй подблок, "Организовать работу персонала", описывает процессы, связанные с организацией работы персонала парикмахерского салона. Это может включать в себя планирование графиков работы, распределение обязанностей, управление рабочим временем, а также контроль за выполнением работниками своих обязанностей.

Третий подблок, "Обучить персонал", описывает процессы, связанные с обучением персонала парикмахерского салона. Это может включать в себя проведение обучающих курсов, тренингов, семинаров и обучение на рабочем месте. Целью этого подблока является обеспечение высокого уровня профессионализма и квалификации персонала.

Четвертый подблок, "Оценить работу персонала", описывает процессы, связанные с оценкой работы персонала. Это может включать в себя проведение аттестации, оценку результативности работы, обратную связь и мотивацию сотрудников. Целью этого подблока является обеспечение высокого уровня производительности и эффективности работы персонала.



Рисунок 1.5 – Декомпозиция блока «Управлять персоналом»

На рисунке 1.6 отображена декомпозиция блока «Управлять инвентарем», состоящая из четырех блоков: «Соотнести товар с его текущим количеством», «Определить остаток товаров на складе и определить расходы», «Отправить запрос на пополнение» и «Договориться о времени и количестве поставок».

Первый блок "Соотнести товар с его текущим количеством" указывает на необходимость отслеживания количества и наличия товаров на складе, чтобы иметь представление о том, какие товары требуют пополнения, а какие уже имеются в достаточном количестве.

Второй блок "Определить остаток товаров на складе и определить расходы" описывает процесс определения остатков товаров на складе и расходов на их приобретение. Этот блок позволяет более точно планировать закупки товаров.

Третий блок "Отправить запрос на пополнение" представляет процесс запроса на пополнение запасов товаров, когда количество товаров на складе достигает предельного уровня. Этот блок предполагает взаимодействие с поставщиками.

Четвертый блок "Договориться о времени и количестве поставок" представляет процесс договоренности с поставщиками о времени и количестве поставок. Этот блок позволяет обеспечить поставки товаров вовремя и в нужном количестве.

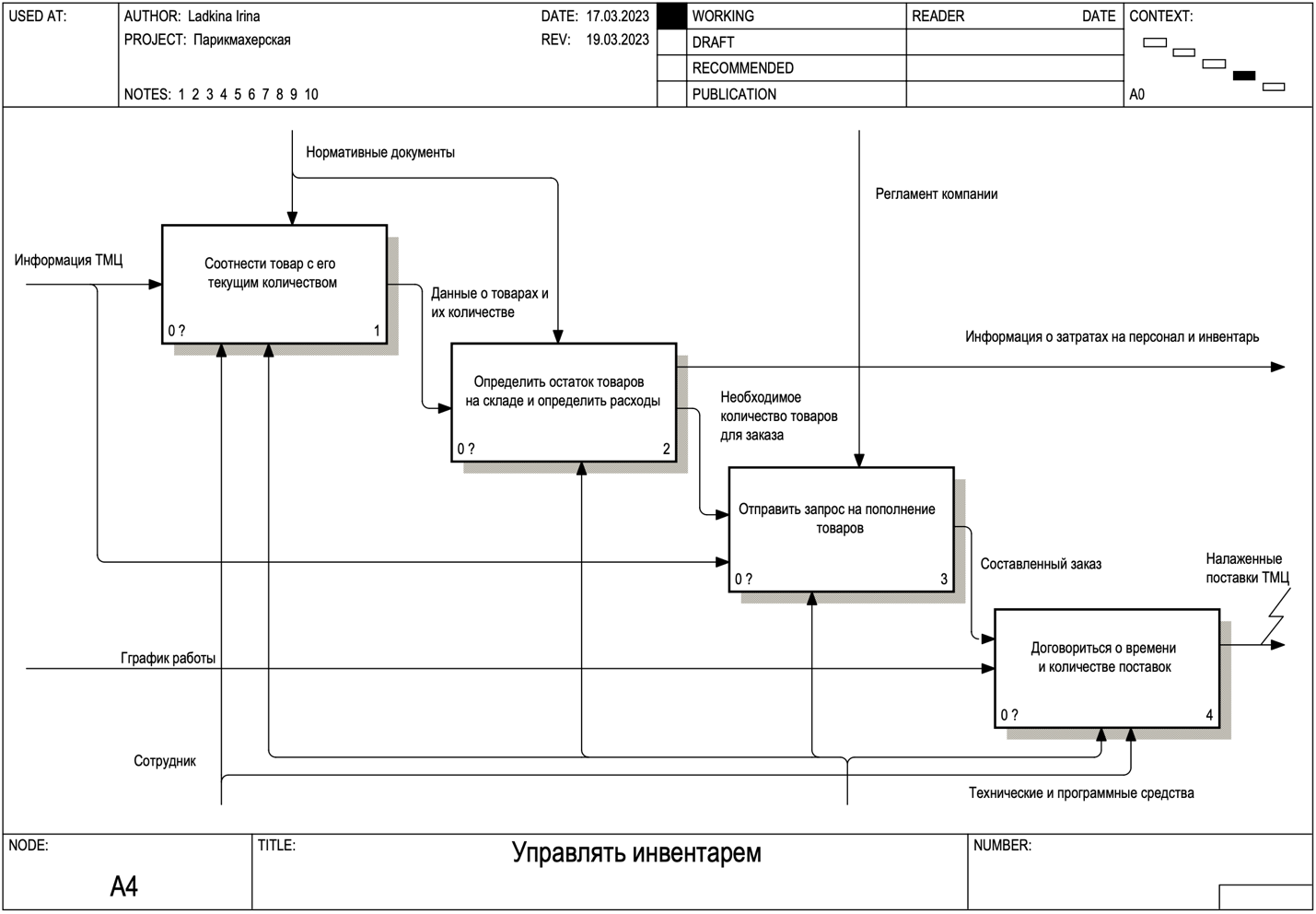


Рисунок 1.6 – Декомпозиция блока «Управлять инвентарем»

Для подробного исследования области ведения складского учета товаров рассмотрим декомпозиции главных подпроцессов данного блока.

На рисунке 1.7 отображена декомпозиция блока «Управлять финансами», состоящая из трех блоков: «Выставить счет», «Обработка платежа», «Управлять бухгалтерией».

Первый блок, «Выставить счет», отвечает за составление и выставление счетов клиентам за оказанные услуги. Это может включать в себя создание счетов-фактур, определение стоимости услуг, расчет налогов и других финансовых аспектов, связанных с предоставлением услуг.

Второй блок, «Обработка платежа», отвечает за обработку платежей клиентов. Это может включать в себя проверку и принятие платежей, учет финансовых транзакций, связанных с оплатой услуг, и т.д.

Третий блок, «Управлять бухгалтерией», отвечает за ведение бухгалтерского учета и отчетности. Это может включать в себя подготовку отчетов о финансовом состоянии парикмахерского салона, учет расходов и доходов, налоговые отчеты и другие финансовые аспекты.



Рисунок 1.7 – Декомпозиция блока «Управлять финансами»

Для подробного исследования области ведения складского учета товаров рассмотрим декомпозиции главных подпроцессов данного блока.

На рисунке 1.8 отображена декомпозиция блока «Управлять клиентами», состоящая из трех блоков: «Принять клиента», «Обслужить клиента», «Учесть клиента».

Блок «Принять клиента» отвечает за прием клиента, регистрацию его данных и определение его потребностей.

Блок «Обслужить клиента» отвечает за предоставление клиенту необходимой информации и оказание ему услуг.

Блок «Учесть клиента» отвечает за фиксацию факта обслуживания клиента, внесение изменений в его профиль и учет его потребностей для будущего обслуживания.

Таким образом, данная декомпозиция позволяет более детально рассмотреть процесс управления клиентами и понять, какие шаги необходимо выполнить для обслуживания клиента.

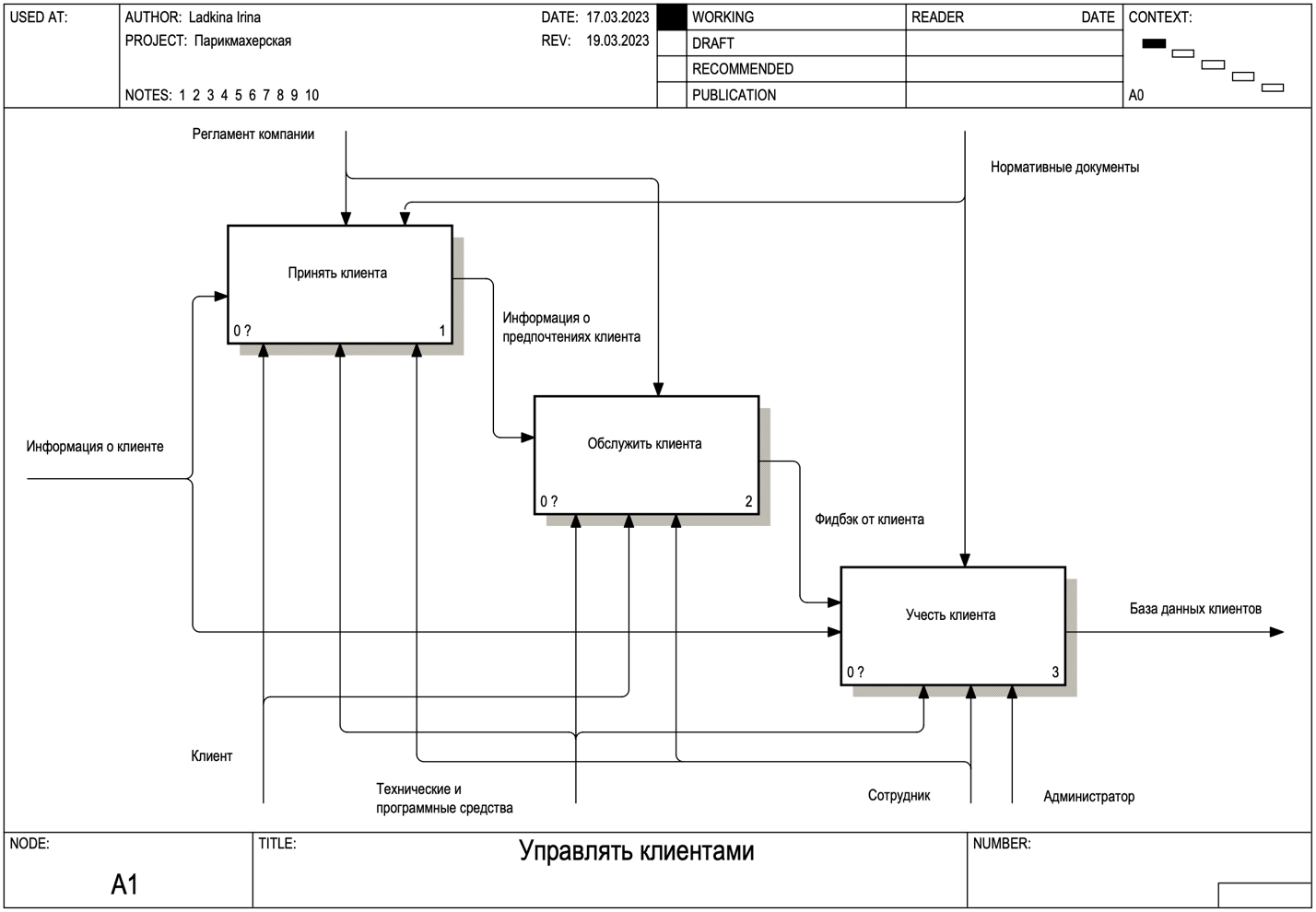


Рисунок 1.8 – Декомпозиция блока «Управлять клиентами»

Таким образом было проведено описание основного процесса предметной области: «Организовать работу парикмахерского салона».

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

Любая компания, в том числе и парикмахерская, должна предоставлять свои услуги наиболее удобным клиенту образом, чтобы оказанный сервис удовлетворял каждого клиента. В свою очередь, процесс обслуживания клиентов должен быть организован таким образом, чтобы позволить компании реализовывать продукцию без каких-либо затруднений. Также при обработке пользовательских данных или принятии каких-либо технических решений сотрудниками фирмы важно свести к минимуму возможность допущения ошибок, для чего удобно использовать различные технические и программные средства.

В основе любой программной системы, а также процесса ее разработки лежат требования к программному обеспечению.

Программное обеспечение для парикмахерской должно быть простым и понятным в использовании, функциональным и эффективном в ряде задач, стоящих перед администрацией компании.

В ходе работы над данным курсовым проектом была поставлена задача реализовать web-приложение с базой данных типа «Клиент-Сервер» с удобным интерфейсом и возможностью безопасного хранения данных.

В данном курсовом проекте для решения были поставлены следующие задачи:

* создание профилей клиентов и сотрудников;
* хранение истории услуг, предоставленных клиентам;
* озможность записи клиентов на прием к парикмахеру;
* уведомление клиентов о предстоящих записях и изменениях;
* просмотр расписания работ сотрудников;
* возможность назначения работы каждому сотруднику;
* ведение учета рабочего времени;
* учет количества и состояния инвентаря;
* ведение учета затрат на инвентарь;
* разграничение доступа к функционалу программы в зависимости от уровня пользователя;
* быстрая и стабильная работа программы при работе с большим объемом данных;
* минимальные временные задержки при выполнении операций и формировании отчетов;
* масштабируемость системы, позволяющая ее расширение и улучшение с ростом бизнеса;
* генерировать отчеты и статистику о выполненных услугах, остатках товаров и других важных показателях;
* предоставлять возможность добавления новых услуг и товаров, а также изменения цен на них;
* сохранять историю выполненных услуг и продаж товаров, чтобы иметь возможность отслеживать и анализировать их в будущем;
* обеспечивать доступ к базе данных только авторизованным пользователям с определенными уровнями доступа;
* интуитивно понятный и удобный пользовательский интерфейс, позволяющий быстро и легко освоить функционал программы;
* реализовать приложение в виде веб-сайта.

Однако главной целью курсового проекта является создание программного средства для автоматизации ведения основных бизнес-процессов парикмахерской.

Для отражения спецификации функциональных требований была создана диаграмма вариантов использования системы парикмахерского салона.

Диаграмма вариантов использования (Use Case Diagram) - это одна из самых распространенных и удобных форм документирования требований в системной инженерии и разработке программного обеспечения. Она представляет собой графическое представление сценариев использования системы, которые называются вариантами использования (Use Case).

Диаграмма вариантов использования включает в себя два типа объектов: актеры (Actors) и варианты использования (Use Cases). Актеры - это внешние сущности, которые взаимодействуют с системой, например, пользователи, другие системы или устройства. Варианты использования - это описания функциональности системы, которые объединены в связанные группы.

На диаграмме вариантов использования актеры представлены в виде стикеров, а варианты использования - в виде овалов. Взаимодействие между актерами и вариантами использования изображается стрелками, которые указывают направление взаимодействия.

Цель диаграммы вариантов использования – документирование всех возможных вариантов использования системы, а также уточнение и описание требований к функциональности системы. Это позволяет разработчикам лучше понимать потребности пользователей и создавать более удобные и функциональные системы.

На рисунках 1.8 – 1.9 представлены диаграммы вариантов использования (Use Case Diagram) парикмахерской.

Рассмотрим подробно все действия, которые могут выполнять данные роли.

Далее рассмотрим варианты использования администратора. Для начала администратор должен зайти в аккаунт, пройдя обязательную авторизацию. После успешной авторизации ему предоставляется 2 основных направления для функционирования: управление данными и анализ данных (рисунок 1.8).

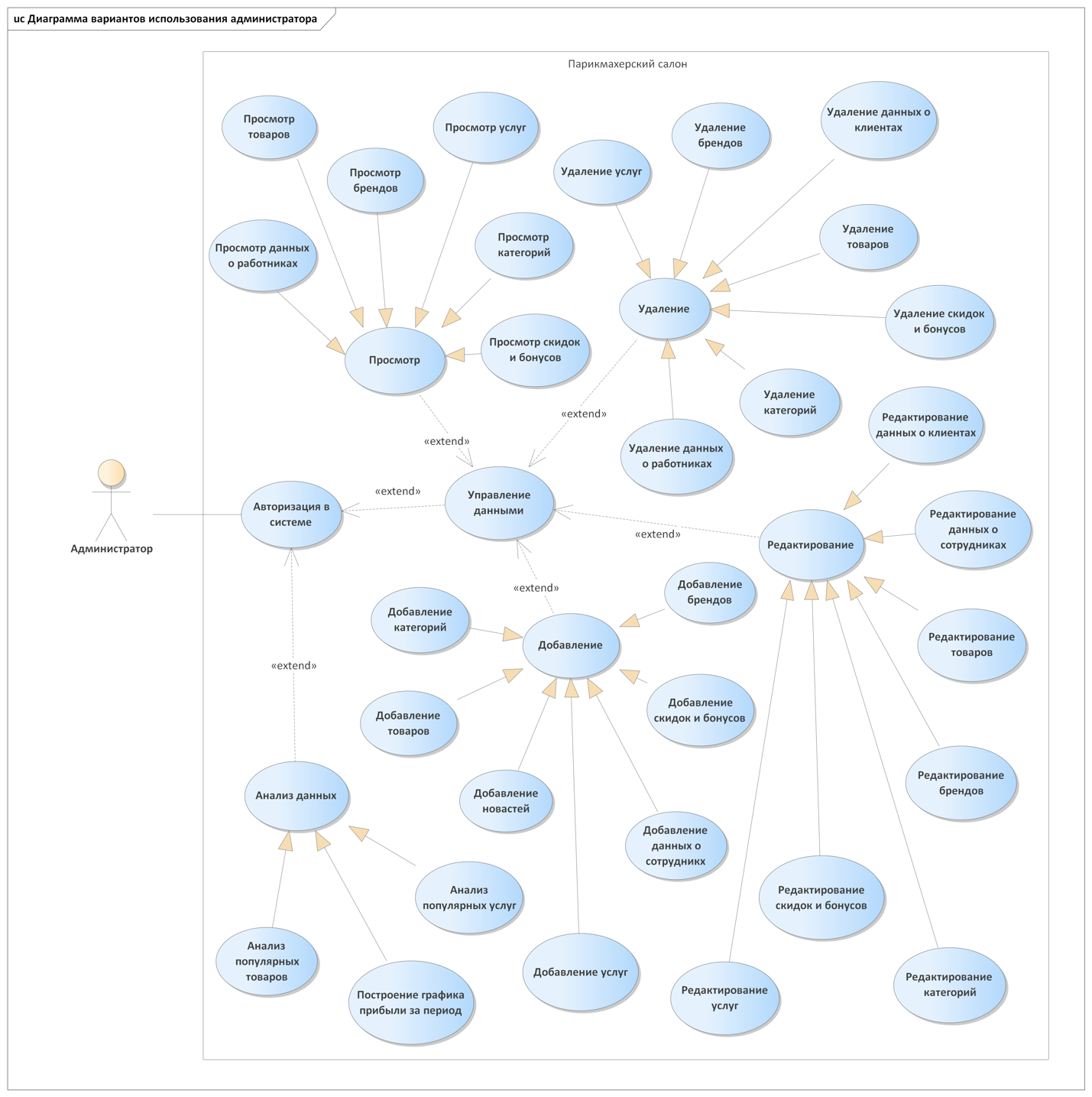


Рисунок 1.8 – Диаграмма вариантов использования администратора

Далее рассмотрим варианты использования клиента. Клиенту не обязательно авторизовываться, ведь ПС позволяет клиенту, будучи гостем, просматривать каталог товаров, пользоваться функциями фильтрации, сортировки и поиска товаров и услуг по параметрам.

Для перехода в личный кабинет клиенту сначала необходимо либо успешно авторизоваться, либо пройти регистрацию аккаунта (рисунок 1.15).



Рисунок 1.9 – Диаграмма вариантов использования клиента

Соответственно автоматизированная система должна быть разработана с учётом всех упомянутых ранее нюансов, принимая во внимание тип аккаунта и соответственно предоставленные ему возможности.

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

Разработка информационной модели предметной области является одним из ключевых этапов проектирования информационных систем и баз данных. Эта модель представляет собой формальное описание данных, которые будут использоваться в системе, и их взаимосвязей.

Информационная модель предметной области помогает проектировщикам информационных систем лучше понимать предметную область, в которой работает система. Она позволяет описать все сущности, атрибуты и связи между ними, которые будут использоваться в системе. Кроме того, информационная модель предметной области позволяет лучше понимать процессы и бизнес-логику, которые будут поддерживаться системой.

Разработка информационной модели предметной области является ключевым шагом при проектировании информационных систем и баз данных. Она позволяет лучше понимать предметную область, на которую ориентирована система, и обеспечивает более эффективное использование данных в системе.

Для отображения структуры хранимых данных библиотеки была выбрана информационная методология IDEF1.X.

IDEF1.X (Integrated Definition 1) является методом для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для удобного построения концептуальной схемы.

Связи в IDEF1.X представляют собой ссылки, соединения и ассоциации между сущностями. Связи – суть глаголы, которые показывают, как соотносятся сущности между собой. Первая сущность называется родительской, а вторая - дочерней.

Данная модель состоит из сущностей, представляющих собой множество объектов, похожих по свойствам, но отличаемых друг от друга по некоторым признакам, связей между ними и соответствующих атрибутов. Сущность представляется в виде прямоугольника, разделённого горизонтальной чертой для разграничения первичных и других (альтернативных) атрибутов. Первичными атрибутами, или же первичными ключами, называются атрибуты родительской сущности, которые передаются затем в дочернюю. Для дочерней сущности данные ключи являются внешними ключами (Foreign Key).

Между собой сущности соединены связями, которые разделяются на два вида отношений: идентифицирующие (сущность становится зависимой) и не идентифицирующие (сущность является независимой).

Отношение – это ассоциация или связь между сущностями, при которой каждый экземпляр одной сущности, называемой родительской сущностью, ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, называемой сущностью-потомком, а каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя.

Отношения в IDEF1.X бывают следующих типов:

* идентифицирующие отношения (первичный ключ одной сущности наследуется другой сущностью и является её ключевым атрибутом);
* не идентифицирующие отношения (первичный ключ одной сущности наследуется другой сущностью, но не является её ключевым атрибутом);
* отношения «многие ко многим» (любой экземпляр сущности «A» может иметь связь с несколькими экземплярами сущности «В», и наоборот);
* отношения категоризации (каждый экземпляр одной сущности связан с одним экземпляром категориальной сущности) [14].

При проектировании системы было принято решение использовать следующие 5 таблиц:

* таблицы admins для хранения данных об администраторах, включая их идентификатор, логин и пароль;
* таблицы clients для хранения данных о клиентах, включая их идентификатор, электронную почту, имя, фамилию и пароль;
* таблицы hairdressers для хранения данных о парикмахерах, включая их идентификатор, опыт работы, имя и фамилию;
* таблицы services для хранения данных об услугах, включая их идентификатор, стоимость и название;
* таблицы sessions для хранения данных о сеансах стрижки, включая их идентификатор, дату и время, идентификатор парикмахера, идентификатор клиента и идентификатор услуги.

Графическое представление информационной модели для системы работы парикмахерского салона представлено на рисунке 1.10.



Рисунок 1.10 – Информационная модель системы парикмахерского салона

В данной схеме присутствуют четыре первичных ключа:

1. PK\_admins: первичный ключ таблицы admins, состоящий из столбца id, который имеет тип int и автоматически инкрементируется при добавлении новой записи. Данный ключ уникально идентифицирует каждую запись в таблице admins.
2. PK\_clients: первичный ключ таблицы clients, состоящий из столбца id, который имеет тип int и автоматически инкрементируется при добавлении новой записи. Данный ключ уникально идентифицирует каждую запись в таблице clients.
3. PK\_hairdressers: первичный ключ таблицы hairdressers, состоящий из столбца id, который имеет тип int и автоматически инкрементируется при добавлении новой записи. Данный ключ уникально идентифицирует каждую запись в таблице hairdressers.
4. PK\_services: первичный ключ таблицы services, состоящий из столбца id, который имеет тип int и автоматически инкрементируется при добавлении новой записи. Данный ключ уникально идентифицирует каждую запись в таблице services.

База данных приведена к 3 нормальной форме, так как требование 3 нормальной формы (3NF) заключается в том, чтобы в таблицах отсутствовала транзитивная зависимость. Транзитивная зависимость представляет собой зависимость неключевых столбцов одной таблицы от значений неключевых столбцов другой таблицы.

SQL-скрипт для генерации базы данных приведен в приложении Б.

1.5 UML-модели представления программного средства и их описание

Для модели представления системы парикмахерского салона был выбран унифицированный язык моделирования.

Нотация UML представляет собой совокупность графических элементов, которые применяются в моделях. Выбор средств для построения моделей сложных систем основывается на нескольких основных принципах: принцип абстрагирования, принцип многомодельности и принцип иерархического построения.

Принцип абстрагирования устанавливает включение в модель только тех аспекты проектируемой системы, которые имеют непосредственное отношение к выполнению целевого предназначения системой.

Использование принципа многомодельности предполагает, что никакое представление системы не является достаточным для полноценного выражения всех её особенностей. Принцип иерархического построения моделей в свою очередь предписывает рассматривать процесс построения модели на всевозможных уровнях абстрагирования, детализации в рамках фиксированных представлений [11].

Различают 2 основных типа UML-диаграмм: структурные и поведенческие.

Структурные диаграммы отражают элементы, из которых состоит система. Поведенческие модели описывают процессы, протекающие в системе.

**1.5.1** Описание диаграммы последовательности системы парикмахерского салона. Современный парикмахерский салон является сложной системой, в которой необходимо эффективно управлять множеством бизнес-процессов, связанных с оказанием услуг, контролем качества, управлением персоналом и многими другими аспектами. Одним из инструментов автоматизации бизнес-процессов в парикмахерском салоне являются диаграммы последовательности.

Диаграммы последовательности являются частным случаем диаграмм взаимодействия и служат для моделирования взаимодействия объектов во времени. Они позволяют проиллюстрировать последовательность действий, выполняемых системой в ответ на запрос пользователя или другую внешнюю стимуляцию. Они позволяют детально описать, как новые функции будут работать в рамках существующей системы и как они будут взаимодействовать с другими компонентами.

Для системы парикмахерского салона диаграммы последовательности могут использоваться для моделирования различных бизнес-процессов. Например, для процесса записи клиента на прием к парикмахеру или управление персоналом и другие. Но в данном курсовом проекте была смоделирована диаграмма последовательности совершения транзакции пользователем парикмахерского салона.

Составленная диаграмма последовательности проиллюстрирована на рисунке 1.18.

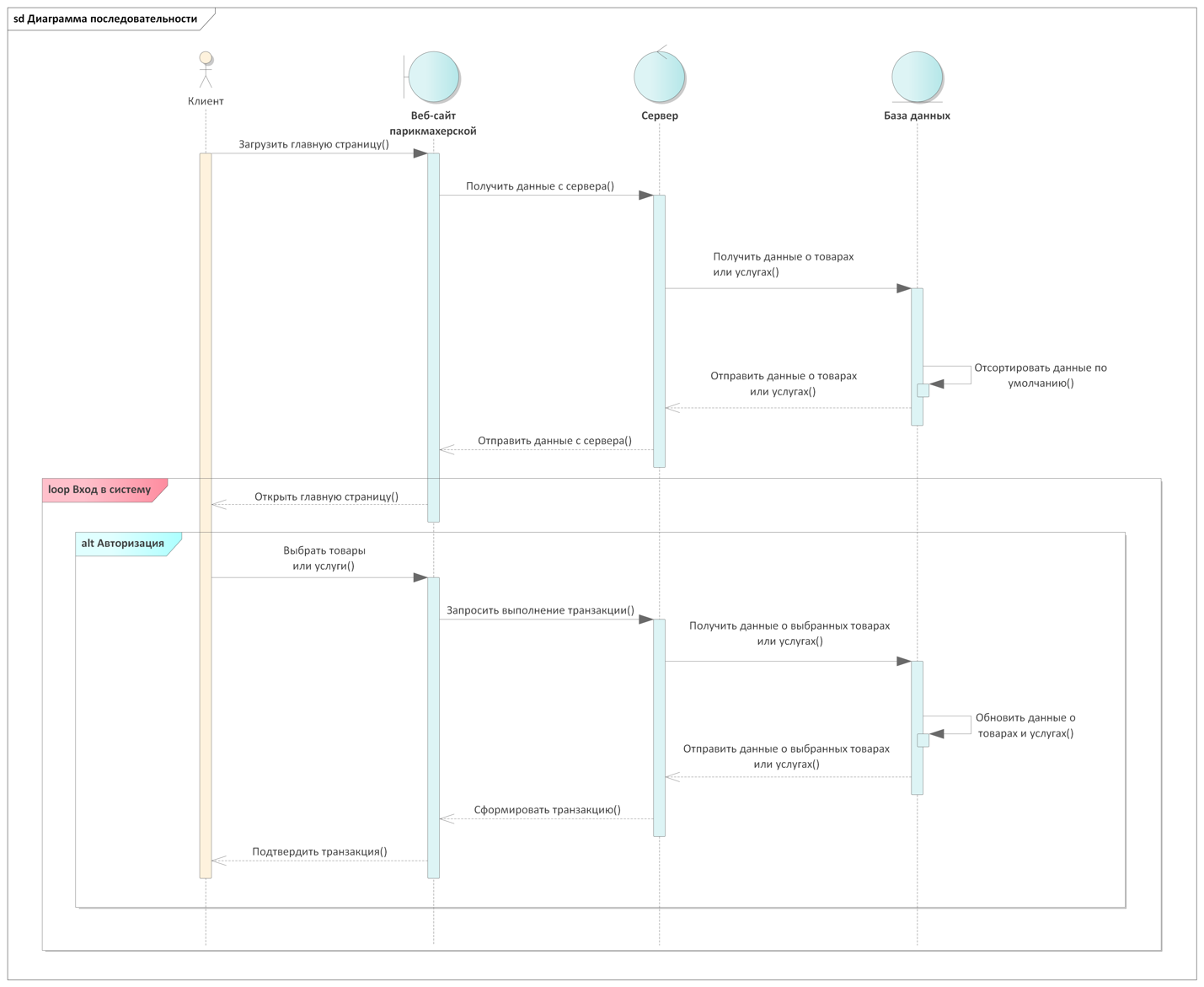


Рисунок 1.11 – Диаграмма последовательности совершения транзакции пользователем парикмахерского салона

Для данного программного средства характерна следующая последовательность действий при отправке клиентом запроса на совершение транзакции. Вначале клиент, зайдя под гостем, может просмотреть товары и услуги, которые ему интересны. Однако выбрать товары для покупки и, как следствие, совершить транзакцию клиент может только после авторизации в системе. Все данные, отправленные или запрашиваемые клиентом, находятся непосредственно в БД.

Таким образом, диаграммы последовательности являются важным инструментом для анализа, проектирования и оптимизации системы парикмахерского салона. Они позволяют разработчикам и аналитикам понимать, как система работает, и выявлять узкие места, которые необходимо оптимизировать для улучшения производительности и качества обслуживания.

1.5.2 Описание диаграммы состояний системы парикмахерского салона. Диаграмма состояний – это диаграмма, показывающая нам все возможные состояния, в которых может находиться объект, а также процесс смены состояний в результате внешнего влияния.

Диаграмма состояний показывает, как объект переходит из одного состояния в другое. Она служат для моделирования динамических аспектов системы. Данная диаграмма полезна при моделировании жизненного цикла объекта. От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса (одного объекта), причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события.

Они могут быть использованы для моделирования различных бизнес-процессов в системе парикмахерского салона, таких как: процесс бронирования услуги, процесс оплаты, процесс оценки качества услуг и т.д. При разработке курсового проекта была составлена диаграмма состояний (рисунок 1.12), описывающая возможные состояния, которые возникают в результате процесса входа пользователя в систему. При вводе некорректных символов (зарезервированные символы программы) или отсутствии данного логина/пароль в БД, пользователь перейдет в состояния о некорректном вводе, если же пользователь ввел данные верно, то в конечном итоге он будет переведен в состояние «Прошедший аутентификацию пользователь».

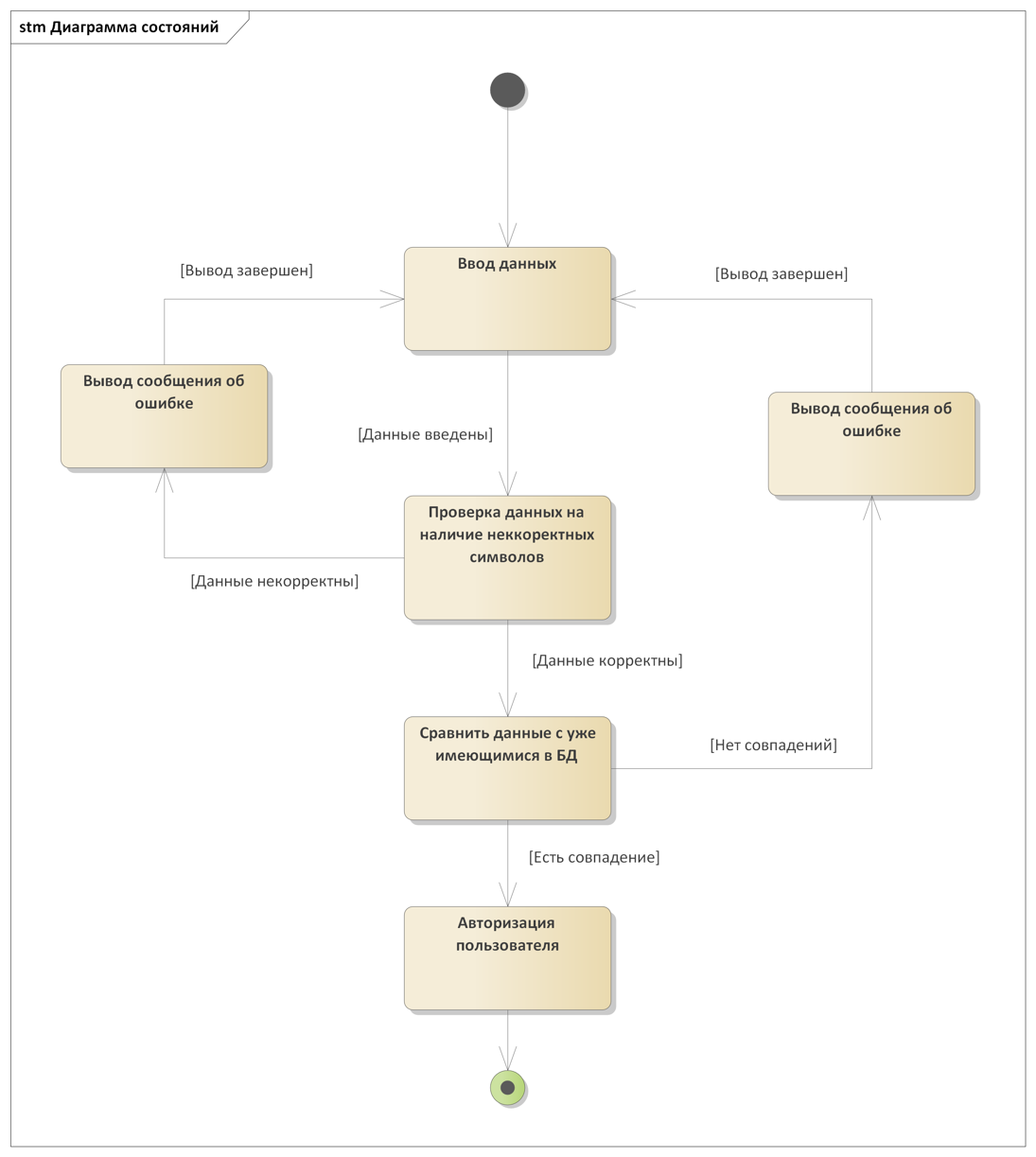


Рисунок 1.12 – Диаграмма состояний аутентификации пользователя

Диаграммы состояний являются важным инструментом для моделирования бизнес-процессов и жизненного цикла объектов в системе парикмахерского салона. Они позволяют определить возможные состояния объектов, действия, выполняемые в рамках каждого состояния, и переходы между состояниями. Это помогает улучшить понимание работы системы и оптимизировать бизнес-процессы.

**1.5.3** Описание диаграммы развертывания программного средства. Диаграмма развертывания – диаграмма, на которой представлены узлы выполнения программных компонентов реального времени, а также процессов и объектов [13].

Диаграмма развертывания применяется для представления общей конфигурации и топологии распределенной программной системы и содержит изображение размещения компонентов по отдельным узлам системы. Кроме того, диаграмма развертывания показывает наличие физических соединений-маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы.

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих только на этапе ее исполнения. При этом представляются только те компоненты программы, которые являются исполнимыми файлами или динамическими библиотеками. Компоненты, не используемые на этапе исполнения, на диаграмме развертывания не показываются.

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. В отличие от диаграмм логического представления, диаграмма развертывания является единственной для системы в целом, поскольку должна отражать все особенности ее реализации. Диаграмма развертывания разрабатывается совместно системными аналитиками, сетевыми инженерами и системотехниками.

Диаграмма развёртывания приложения представлена на рисунке 1.13.

Как видно из диаграммы, для развертывания приложения необходимо предоставить базу данных, запустить серверное приложение, затем запустить клиентское приложение на устройстве. Для полноценной работы системы необходимы три основных вида окружения: бекенд-окружение, веб-окружение и мобильная ОС Android.

Клиентское приложение запускается на мобильных устройствах, что дает клиентам мобильность, а также возможность интеграции с другими приложениями и сервисами благодаря возможностям ОС, в то время как приложение для сотрудника разворачивается на веб-окружении, что будет означать ресурсо-эффективность, поддержку большого количества устройств, удобный интерфейс для стационарных машин, как правило, размещаемых в торговых точках, и гибкость в установке и использовании в различных организациях, торговых точках.

Приложение руководителя также запускается в веб-окружении для обеспечения простоты и гибкости работы с ним. Приложение оптимизировано под настольные ПК и широкие экраны ввиду большого количества текстовой информации, комплексной навигации и динамического содержимого.

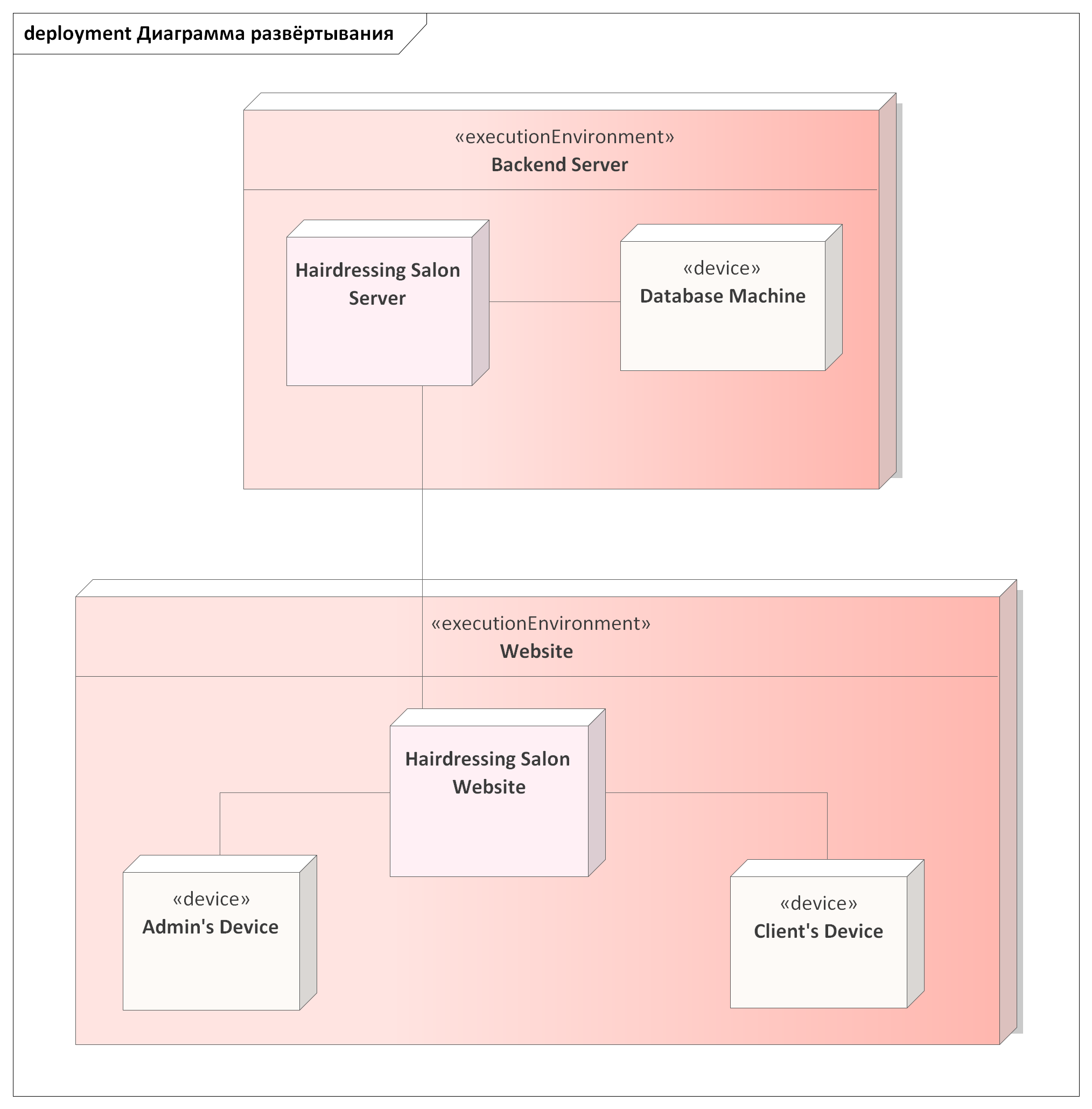


Рисунок 1.13 – Диаграмма развёртывания приложения

Диаграмма развёртывания является очень полезным инструментом для проектирования и разработки сложных систем. Она позволяет разработчикам и системным администраторам понять, как компоненты системы будут взаимодействовать друг с другом и как они будут работать в различных средах выполнения. Это помогает создать более эффективную и надёжную систему, которая будет соответствовать требованиям пользователей и бизнеса.

**1.5.4** Описание диаграммы классов системы парикмахерской. Диаграмма классов (Class diagram) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы.

Класс (Class) служит для обозначения множества объектов, которые обладают одинаковой структурой, поведением и отношениями с объектами из других классов. Графически класс изображается в виде прямоугольника, который дополнительно может быть разделен горизонтальными линиями на разделы или секции.

Имя класса должно быть уникальным в пределах пакета, который описывается некоторой совокупностью диаграмм классов (возможно, одной диаграммой). Оно указывается в первой верхней секции прямоугольника.

В первой секции обозначения класса могут находиться ссылки на стандартные шаблоны или абстрактные классы, от которых образован данный класс и, соответственно, от которых он наследует свойства и методы. В этой секции может приводиться информация о разработчике данного класса и статус состояния разработки, а также могут записываться и другие общие свойства этого класса, имеющие отношение к другим классам диаграммы или стандартным элементам языка UML.

Класс может не иметь экземпляров или объектов. В этом случае он называется абстрактным классом, а для обозначения его имени используется наклонный шрифт (курсив).

Кроме внутреннего устройства или структуры классов на соответствующей диаграмме указываются различные отношения между классами. При этом совокупность типов таких отношений фиксирована в языке UML и предопределена семантикой этих типов отношений. Базовыми отношениями или связями в языке UML являются:

* отношение зависимости (dependency relationship);
* отношение зависимости (association relationship);
* отношение обобщения (generalization relationship);
* отношение реализации (realization relationship).

Отношение зависимости в общем случае указывает некоторое семантическое отношение между двумя элементами модели или двумя множествами таких элементов, которое не является отношением ассоциации, обобщения или реализации. Оно касается только самих элементов модели и не требует множества отдельных примеров для пояснения своего смысла.

Отношение ассоциации соответствует наличию некоторого отношения между классами. Данное отношение обозначается сплошной линией с дополнительными специальными символами, которые характеризуют отдельные свойства конкретной ассоциации.

Отношение агрегации имеет место между несколькими классами в том случае, если один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности.

Отношение обобщения является обычным таксономическим отношением между более общим элементом (родителем или предком) и более частным или специальным элементом (дочерним или потомком). Данное отношение может использоваться для представления взаимосвязей между пакетами, классами, вариантами использования и другими элементами языка UML [11].

Полная диаграмма классов отображена в приложении Д (рисунок Д.1).

Рассмотрим последовательно несколько главных классов серверной части приложения.

Класс клиента, имеет следующие свойства:

* id (long): уникальный идентификатор клиента;
* surname (String): фамилия клиента;
* email (String): электронная почта клиента;
* password (String): пароль для учетной записи клиента;
* name (String): имя клиента.

В дополнение к этим свойствам класс клиента имеет несколько методов, которые можно использовать для взаимодействия с объектом клиента:

* getName(): возвращает имя клиента в виде строки (String);
* setEmail(String): устанавливает адрес электронной почты клиента на указанное значение типа String;
* getEmail(): возвращает адрес электронной почты клиента в виде строки (String);
* getSurname(): возвращает фамилию клиента в виде строки (String);
* getId(): возвращает уникальный идентификатор клиента в виде значения типа long;
* setId(long): устанавливает уникальный идентификатор клиента на указанное значение типа long;
* setPassword(String): устанавливает пароль для учетной записи клиента на указанное значение типа String;
* setName(String): устанавливает имя клиента на указанное значение типа String;
* setSurname(String): устанавливает фамилию клиента на указанное значение типа String;
* getPassword(): возвращает пароль для учетной записи клиента в виде строки (String);

Эти методы позволяют получать и изменять свойства объекта клиента по мере необходимости. Например, мы можем использовать метод setName(), чтобы обновить имя клиента, или setEmail(), чтобы изменить его адрес электронной почты.

Класс сервиса, имеет следующие свойства:

* id (long): уникальный идентификатор сервиса;
* name (String): название сервиса;
* cost (float): стоимость сервиса.

Класс сервиса имеет несколько методов, которые могут быть использованы для взаимодействия с объектом сервиса:

* setName(String): устанавливает название сервиса на указанное значение типа String;
* getCost(): возвращает стоимость сервиса в виде значения типа float;
* getName(): возвращает название сервиса в виде строки (String);
* setCost(float): устанавливает стоимость сервиса на указанное значение типа float;
* setId(long): устанавливает уникальный идентификатор сервиса на указанное значение типа long;
* getId(): возвращает уникальный идентификатор сервиса в виде значения типа long.

Эти методы позволяют получать и изменять свойства объекта сервиса по мере необходимости. Например, мы можем использовать метод setName(), чтобы обновить название сервиса, или setCost(), чтобы изменить его стоимость.

Класс парикмахера, который вы описали, имеет следующие свойства:

* experience (int): опыт работы парикмахера;
* id (long): уникальный идентификатор парикмахера;
* surname (String): фамилия парикмахера;
* name (String): имя парикмахера.

Класс парикмахера имеет несколько методов, которые можно использовать для взаимодействия с объектом парикмахера:

* setName(String): устанавливает имя парикмахера на указанное значение типа String;
* setSurname(String): устанавливает фамилию парикмахера на указанное значение типа String;
* getExperience(): возвращает опыт работы парикмахера в виде значения типа int;
* getId(): возвращает уникальный идентификатор парикмахера в виде значения типа long;
* setExperience(int): устанавливает опыт работы парикмахера на указанное значение типа int;
* getSurname(): возвращает фамилию парикмахера в виде строки (String);
* setId(long): устанавливает уникальный идентификатор парикмахера на указанное значение типа long.

Эти методы позволяют получать и изменять свойства объекта парикмахера по мере необходимости. Например, мы можем использовать метод setName(), чтобы обновить имя парикмахера, или setExperience(), чтобы изменить его опыт работы.

Класс сеанса, имеет следующие свойства:

* hairdresser (Hairdresser): парикмахер, назначенный на сеанс;
* service (Service): сервис, предоставляемый в рамках сеанса;
* client (Client): клиент, который забронировал сеанс;
* dateTime (LocalDateTime): дата и время проведения сеанса;
* id (long): уникальный идентификатор сеанса.

Класс сеанса имеет несколько методов, которые можно использовать для взаимодействия с объектом сеанса:

* setHairdresser(Hairdresser): устанавливает парикмахера на указанное значение типа Hairdresser;
* getService(): возвращает сервис в рамках сеанса в виде объекта типа Service;
* setClient(Client): устанавливает клиента на указанное значение типа Client;
* getClient(): возвращает клиента в рамках сеанса в виде объекта типа Client;
* setService(Service): устанавливает сервис на указанное значение типа Service;
* getDateTime(): возвращает дату и время проведения сеанса в виде объекта типа LocalDateTime;
* getHairdresser(): возвращает парикмахера в рамках сеанса в виде объекта типа Hairdresser;
* setDateTime(LocalDateTime): устанавливает дату и время проведения сеанса на указанное значение типа LocalDateTime.

Эти методы позволяют получать и изменять свойства объекта сеанса по мере необходимости.

Класс Admin, который вы описали, имеет следующие свойства:

* login (String): логин администратора;
* password (String): пароль администратора;
* id (long): уникальный идентификатор администратора.

Класс Admin имеет несколько методов, которые можно использовать для взаимодействия с объектом Admin:

* getLogin(): возвращает логин администратора в виде строки типа String;
* getId(): возвращает уникальный идентификатор администратора в виде числа типа long;
* setId(long): устанавливает уникальный идентификатор администратора на указанное значение типа long;
* setPassword(String): устанавливает пароль администратора на указанное значение типа String;
* setLogin(String): устанавливает логин администратора на указанное значение типа String;
* getPassword(): возвращает пароль администратора в виде строки типа String.

Эти методы позволяют получать и изменять свойства объекта Admin по мере необходимости.

Для данной главы была описана структура хранимых данных системы парикмахерского салона, а также построены следующие модели представления с использованием унифицированного языка моделирования: диаграмма вариантов использования, диаграмма последовательности, диаграмма состояний, диаграмма классов и диаграмма развертывания.

# Проектирование и конструирование программного средства

Проектирование программного обеспечения и конструирование программного обеспечения являются двумя важными дисциплинами программной инженерии.

Проектирование программного обеспечения – это процесс создания проекта ПО, который включает в себя определение требований к системе, анализ этих требований, определение внутренних свойств системы и детализацию ее внешних свойств. Целью проектирования является создание дизайна, который оптимизирует производительность, эффективность и надежность системы, а также обеспечивает удовлетворение потребностей пользователя.

Конструирование программного обеспечения – это дисциплина программной инженерии, которая связана со всеми другими дисциплинами программной инженерии и включает в себя создание работающего программного обеспечения с помощью комбинации кодирования, верификации, модульного тестирования, интеграционного тестирования и отладки. Целью конструирования ПО является создание программного обеспечения, которое соответствует спецификациям, быстро работает, легко поддерживается и легко расширяется.

Обе дисциплины очень важны для процесса разработки программного обеспечения, и успешное выполнение проекта ПО требует хорошо спланированного и реализованного процесса проектирования и конструирования ПО.

* 1. **Постановка задачи**

При проектировании данного курсового проекта для решения были поставлены следующие задачи:

* применить ряд решений для создания web-приложения;
* реализовать бесперебойную работу сайта для реализации продукции;
* обеспечить необходимый функционал для комфортного поиска/сортировки/фильтрации;
* создать приложение с архитектурой «Клиент-Сервер»;
* реализовать удобный пользовательский интерфейс и понятную логику работы программы;
* использовать базу данных для хранения данных и доступа к ним;
* обеспечить возможность ввода, редактирования и удаления данных;
* предусмотреть обработку исключительных ситуаций;
* разграничить функции пользователей;
* обеспечить лёгкий и быстрый доступ к необходимой запрашиваемой информации;
* предоставить администрации графическую отчётность в виде диаграмм и графиков;
* обеспечить формирование итогового отчета о полученной прибыли;
* реализовать приложение в виде веб-сайта.

Благодаря автоматизации разработанная программа позволяет решить

следующие задачи:

– повысить прибыль компании;

– облегчить процесс совершения покупок;

– защитить данные пользователей от несанкционированного доступа;

– предоставить доступ к своевременной и актуальной информации.

* 1. **Обоснование выбора компонентов и технологий для реализации программного средства**

В качестве основного средства разработки серверной части был выбран объектно-ориентированный язык программирования Java, так как он отлично подходит для разработки серверных приложений. В качестве его преимуществ можно выделить безопасность, независимость от платформы, стабильность и высокую производительность.

Для клиентской части были выбраны языки HTML/CSS/JavaScript и технологии Java Server Pages. С помощью данной комбинации можно создавать выразительные и понятные в использовании интерфейсы.

Основной средой разработки программного приложения является IntelliJ IDEA и VSCode. Сегодня IntelliJ IDEA – это наиболее интеллектуальная среда. Она по ходу написания и выполнения может проанализировать код, выявить ошибки и предложить достойное решение. Не менее важным инструментом является и графический редактор для создания интерфейса. Это значительно удобнее, чем постоянно писать шаблонный код. VSCode – один из самых популярных редакторов кода, потому что он бесплатный и открытый, его можно сделать каким угодно под свои задачи.

Серверное приложение было запущено на облачном сервере. При этом база данных, используемая в приложении, отделена от машины, на которой оно запускается, таким образом, можно запустить несколько различных поставщиков данных (приложений) на многих устройствах для повышения надежности. В реализации используется реляционная база данных MySQL.

Соединение происходит на основе ORM – платформенно независимого промышленного стандарта взаимодействия Java-приложений с различными СУБД [13]. **ORM**или**, Object-relational mapping,** – это технология программирования, которая позволяет преобразовывать несовместимые типы моделей в ООП, в частности, между хранилищем данных и объектами программирования. ORM используется для упрощения процесса сохранения объектов в реляционную базу данных и их извлечения, при этом ORM сама заботится о преобразовании данных между двумя несовместимыми состояниями.

Для преобразования Java-объектов в текстовый формат JSON (сериализация) и обратного преобразования (десереализация) была использована библиотека Google Gson.

**JSON (Java Script Object Notation)** – это формат обмена данными по сети, основанный на **JavaScript** (что следует из названия). Формат **JSON**очень прост и может легко использоваться при сетевом взаимодействии. Для преобразования из JSON-формата в строку используется метод **fromJson()** в качестве второго параметра передается тип нашей коллекции. Для парсинга в **JSON**-строку принцип тот же самый, только используется метод **Gson.toJson().**

В целом, выбранные архитектурные решения и технологии позволят создать надежное и функциональное программное средство, способное удовлетворить потребности пользователей и обеспечить успешное функционирование парикмахерского салона.

* 1. **Архитектурные решения**

Архитектурные решения играют важную роль в создании качественного программного продукта, который соответствует требованиям заказчика и потребностям пользователей. В данной подглаве будут рассмотрены основные архитектурные решения, выбранные для реализации разрабатываемого программного средства.

В данном курсовом проекте использовалась архитектура «Клиент-Сервер», в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически, клиент и сервер – это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, загрузка файлов посредством HTTP, FTP, BitTorrent, потоковое мультимедиа или работа с базами данных) или в виде сервисных функций (например, работа с электронной почтой, общение посредством систем мгновенного обмена сообщениями или просмотр web-страниц во всемирной паутине).

На рисунке 2.5 наглядно показана архитектура взаимодействия клиента, сервера и базы данных.

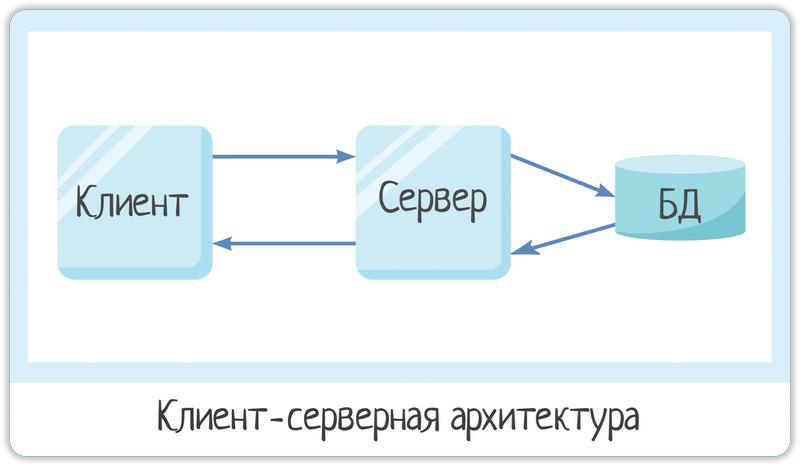


Рисунок 2.1 – Клиент-серверная архитектура

Клиенты – это так называемые объекты, которые запрашивают необходимую им информацию у серверов. Прикладной программный интерфейс – это набор всевозможных функций и программ, которые обеспечивают взаимодействие клиентов и сервера.

Для написания пользовательского интерфейса приложения были использованы языки верстки и стилей HTML, CSS и JavaScript. Эти языки были выбраны в реализации сайта, так как они позволяют добиться максимально понятного и выразительного интерфейса.

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

Для того чтобы реализовать необходимую бизнес-логику, прежде всего необходимо составить алгоритмы работы главных бизнес-процессов, которые определяют порядок работы приложения и позволяют достичь поставленных целей. Далее будут описаны алгоритмы, реализующие ключевую бизнес-логику разрабатываемого программного средства.

**2.4.1** Схема алгоритма создания графиков. Составление индивидуального графика о выполненных заказах – важная часть работы парикмахерского салона, так как именно график помогает определять продуктивность работников, справедливо распределять заработную плату.

Вначале выбирается тип графика, а именно, линейный, затем выбирается сотрудник и выполненные им заказы. Данные распределяются по дате записи клиента и, если данные не пусты, то дата сопоставляется с количеством заказов, строится график и выводится на экран.

Данный алгоритм проиллюстрирован на схеме, которая представлена в приложении Г (рисунок Г.1).

**2.4.2** Схема алгоритма обработки заявки в системе (создать заявку). Необходимо открыть страницу для добавления новой заявки. После чего начинается работа алгоритма пока заявка не будет добавлена в базу, или пока пользователь не уйдет со страницы.

Пользователю необходимо ввести данные, после чего отправить данные на сервер. После прихода данных на сервер они проверяются. Spring обеспечивает возможность предопределить ошибки (задать для полей объекта определенные условия, которым они должны соответствовать). После отправки сообщения на сервер Spring автоматически проверит их по заранее готовому шаблону, и, если ошибки будут иметь место, к ним можно будет обратиться. Далее проводится проверка на наличие ошибок ввода.

При отсутствии ошибок ввода происходит проверка на назначенного сотрудника. При наличии id администратора выполняется запрос к базе данных для его поиска по id и далее добавление в заявку.

При отсутствии id или его наличии проверяем, есть ли id заявителя. Если есть выполняем поиск заявителя в базе данных по id, если нет, значит выполняем поиск по фамилии. Далее добавляем в заявку заявителя. После чего, добавляем в заявку текущую дату и время создания.

Далее проверяется статус на закрыть ли заявку. Если статус закрыть, берем текущее время и дату и добавляем в заявку дату окончания. Данная функциональность предусмотрена для администраторов в случае, если заявку забудут зарегистрировать и сразу выполнят, после чего внесут в систему уже выполненной. Если статус не установлен как закрытый, продолжаем работу алгоритма.

После проверки статуса сохраняем заявку в базу данных. Далее происходит перевод пользователя на страницу со всеми заявками (администратор) или всеми моими заявками для клиента.

Данный алгоритм проиллюстрирован на схеме, которая представлена в приложении Г (рисунок Г.2).

Таким образом схемы алгоритмов работы приложения позволяют наглядно представить, как и в какой последовательности будет реализован проект или каким образом будет осуществляться работа различных функций. Поэтому их проектирование является неотъемлемой частью при разработке любого программного обеспечения.

* 1. **Проектирование пользовательского интерфейса**

Интерфейс – это вся видимая пользователю часть сервиса, с которой он взаимодействует, решая свои задачи.

Проектирование интерфейсов – это всегда поиск наиболее эффективного решения, которое основано на понимании задач, мотиваций и обстоятельств пользователей и в то же время учитывает цели, возможности и ограничения со стороны бизнеса и технологий.

Одним из требований к хорошему графическому интерфейсу программной системы является концепция «предсказуемости», чтобы система работала предсказуемо, чтобы пользователь заранее интуитивно понимал, какое действие выполнит программа после получения его команды.

При запуске веб-сайта от лица пользователя мы попадаем на главную страницу (рисунок 2.2).

Здесь расположены такие кнопки как:

* Домашняя страница: Это главная страница личного кабинета;
* Мастера: На этой странице пользователи могут ознакомиться с информацией о парикмахерах или стилистах, работающих в салоне. Здесь представлены фотографии мастеров, информация об их опыте и специализации;
* Услуги: На этой странице представлена информация об услугах парикмахерского салона. Пользователи могут узнать о различных видах стрижек, окрашиваний и других услугах, которые предлагает салон. Здесь также указаны цены на услуги и акции или специальные предложения;
* Записаться: Эта страница позволяет пользователям записаться на прием к парикмахеру или стилисту. Здесь можно выбрать дату и время визита, а также выбрать мастера и услугу;
* Мои записи: На этой странице пользователи могут просмотреть информацию о своих предстоящих или прошедших записях в парикмахерский салон. Здесь можно проверить дату и время визита;
* Информация об аккаунте: Эта страница содержит информацию об учетной записи пользователя. Здесь можно изменить личные данные, такие как имя, фамилия, адрес электронной почты, номер телефона и количество записей;
* О нас: На этой странице представлена информация о парикмахерском салоне, указан адрес салона, название и контактная информация;
* Выход: Эта кнопка позволяет выйти из учетной записи пользователя. После выхода пользователь будет перенаправлен на главную страницу сайта.

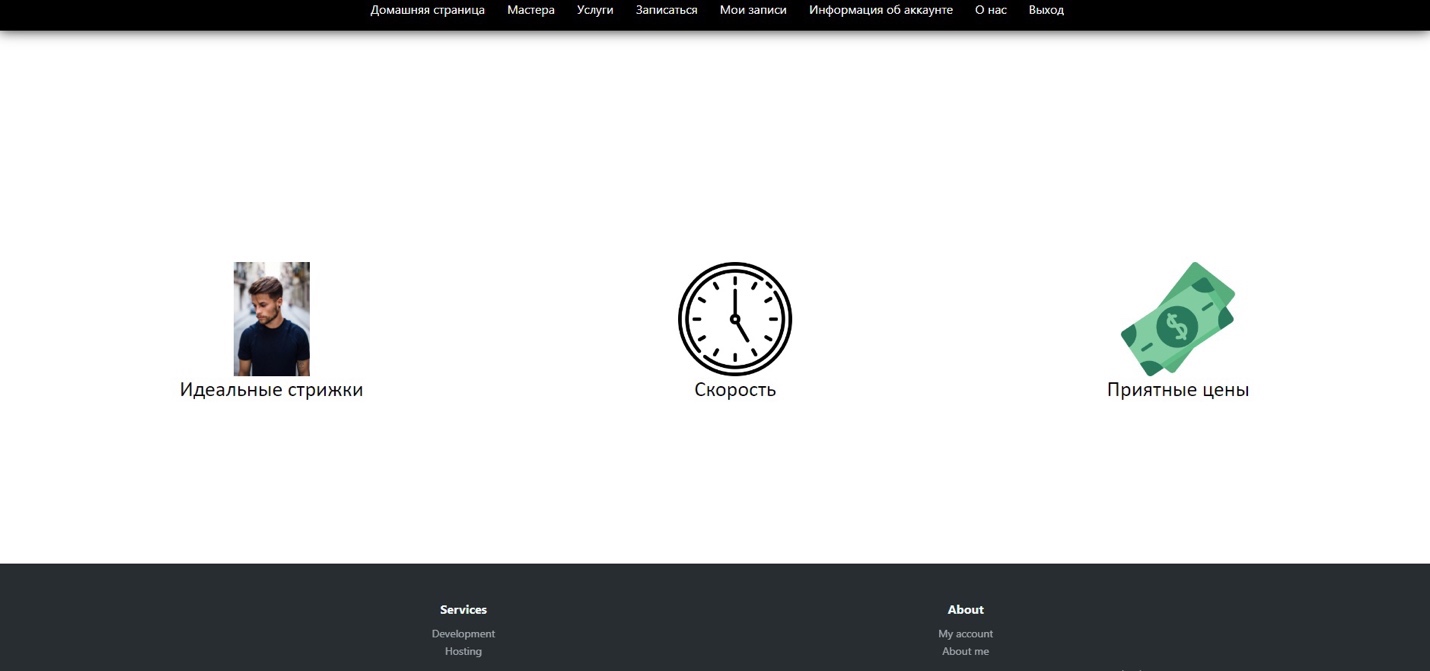


Рисунок 2.2 – Главная страница клиента

Все компоненты взаимодействуют друг с другом и связаны. Так же могут работать по отдельности. Разработанный интерфейс является user- friendly интерфейсом, так как он интуитивно понятен и примитивен.

* 1. **Методы и средства, используемые для обеспечения безопасности данных**

Важно учитывать требования к безопасности данных и обеспечить надежную защиту личной информации пользователей, а также информации о продажах и прибыли парикмахерского салона. Безопасность данных является одним из наиболее важных аспектов в разработке программного обеспечения для парикмахерского салона, так как нарушение безопасности данных может привести к утечке конфиденциальной информации о клиентах, потере денежных средств, а также негативно сказаться на репутации парикмахерского салона.

Один из ключевых методов обеспечения безопасности данных в интернет-магазине – это использование аутентификации пользователей. Аутентификация позволяет убедиться в том, что пользователь, который запрашивает доступ к системе, является тем, за кого себя выдает. Для этого можно использовать различные методы аутентификации, такие как логин и пароль, двухфакторную аутентификацию и т.д.

В данном проекте одной из важнейших задач является защищенная аутентификация в системе. Для этого был использован MD5 – это 128-битный алгоритм хеширования, разработанный профессором Рональдом Л. Ривестом из Массачусетского технологического института (MIT) в 1991 году. Он предназначен для создания «отпечатков» или дайджестов сообщения произвольной длины и последующей проверки их подлинности.

Один из способов использования MD5 – это хеширование паролей. Сайт хранит и сравнивает только отпечатки паролей, но не сами пароли в открытом виде. Таким образом, в случае, если злоумышленник украдет базу данных сайта – он не сможет узнать исходный пароль.

Один из плюсов использования MD5 – это проверка целостности информации. Так как алгоритм MD5 всегда выдает одинаковый результат для одного и того же входного значения, пользователи могут сравнить хеш исходного файла с новым хешем целевого файла, чтобы убедиться, что он остался нетронутым и неизменным.

В целом, обеспечение безопасности данных парикмахерского салона является очень важным аспектом разработки программного обеспечения, и требует применения различных методов и средств защиты, а также постоянного контроля. Поэтому важно следить за обновлением системы и использовать все доступные средства защиты для обеспечения безопасности данных и защиты личной информации клиентов.

# Тестирование и проверка работоспособности программного средства

Тестирование – процесс анализа программного средства и сопутствующей документации с целью поиска дефектов и повышения качества продукта [14].

Для проверки уровня базовых пользовательских требований будем использовать тестирование на основе тест-кейсов.

Тест-кейс – набор входных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения программного средства.

Оформленные основные тест-кейсы для данного программного приложения представлены в таблице 3.1.

Таблица 3.1 – Тест-кейсы для проверки базовых пользовательских требований

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Заглавие тест-кейса | Шаги тест-кейса | Ожидаемый результат | Статус тест-кейса |
| 1. | Регистрация нового пользователя в системе | * 1. Открыть форму регистрации.   2. Заполнить поля формы необходимыми данными | * + 1. Форма регистрации открылась, все поля доступны для ввода.     2. При вводе корректных данных пользователь успешно зарегистрирован. При неправильном вводе выводится сообщение об ошибке. | Passed |
| 2. | Авторизация пользователя в системе | * + - 1. Открыть форму авторизации.       2. Ввести данные. | * + - * 1. Форма авторизации открылась, все поля доступны.         2. Вход в аккаунт. При неправильном вводе данных вывод на экран сообщения об ошибке. | Passed |

Продолжение таблицы 3.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Заглавие тест-кейса | Шаги тест-кейса | Ожидаемый результат | Статус тест-кейса |
| 3. | Добавление новой услуги | Открыть форму добавления услуги.  Заполнить поля данными. | Форма добавления услуги открылась, все поля доступны для ввода.  При вводе корректных данных услуга успешно добавлена. При неверном вводе выводится сообщение об ошибке. | Passed |
| 4. | Удаление мастера-работника | 1. Нажать кнопку «Услуги».  2.Проверить, существуют ли зарегистрированные услуги мастера в системе.  3. Нажать кнопку «Мастера».  4. Нажать на знак «корзина» у нужного элемента. | 1. Вкладка услуг открылась.  2. Пользователь видит список всех существующих услуг.  3. Вкладка мастеров открылась.  4. Пользователь видит список всех существующих мастеров.  5. Мастер удален. При наличии не удаленных услуг вывод сообщения об ошибке. | Passed |
| 5. | Фильтрация услуг | 1. Открыть форму фильтрации услуг.  2. Заполнить поля необходимыми данными.  3. Дождаться вывода результата. | 1. Форма фильтрации услуг открылась.  2. Все поля для ввода доступны.  3. Пользователь видит доступные услуги для записи. | Passed |

Продолжение таблицы 3.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Заглавие тест-кейса | Шаги тест-кейса | Ожидаемый результат | Статус тест-кейса |
| 6. | Печать отчета о предоставленных услугах | 1. Нажать кнопку «Выписка». | 1. Открылся pdf-файл на текущую дату, содержащей название услуги, мастера, вид и количество. | Passed |
| 7. | Редактирование профиля пользователя | 1. Открыть форму редактирования профиля.  2. Ввести новые данные.  3. Подтвердить их. | 1. Форма редактирования профиля открылась, все поля доступны для ввода.  2. При верном вводе данных информация о профиле успешно отредактирована. При неправильном вводе выводится сообщение об ошибке. | Passed |
| 9. | Формирование заказа пользователя | 1. Нажать кнопку «Фильтрация».  2. Ввести данные для фильтрации.  3. Выбрать интересующие услуги. | 1. Форма фильтрации услуг открылась, все поля доступны для ввода.  2. Услуги успешно добавлены. | Passed |

В данной главе было описано тестирование программного приложения с помощью тест-кейсов. Все тест-кейсы были выполнены со статусом «Успешно». Таким образом, разработанные тест-кейсы помогли избежать потери данных и некорректной работы приложения при возникновении исключительных ситуаций.

# Руководство по развертыванию и использованию программного средства

# 

* 1. **Руководство по установке (развертыванию) программного средства**

Для развертывания данного приложения необходимо иметь ПК с доступом к интернету, в веб-браузере включить поддержку JavaScript, а также скачать репозиторий с клиентским интерфейсом, который содержит все необходимые файлы для запуска приложения.

После скачивания репозитория следует запустить файл index.html с помощью веб-браузера (например, Google Chrome, Safari, Microsoft Edge и т.д.) и далее пользователь увидит первую страницу сайта (рисунок 4.1). Где пользователь может авторизоваться, заполнив корректно предложенные поля, или пройти процесс регистрации.



Рисунок 4.1 – Главная страница сайта

Благодаря простоте установки и запуска приложения, пользователь может легко запустить данное программное приложение на любом устройстве, которое удовлетворяет требованиям к работе с веб-приложениями. Это позволяет обеспечить максимальную доступность и удобство использования приложения для пользователей.

* 1. **Руководство пользователя**

Руководство пользователя является важной частью программного продукта и предназначено для того, чтобы помочь пользователям освоиться с интерфейсом и функционалом программы. В данном курсовом проекте руководство пользователя описывает основные возможности парикмахерского салона и дает инструкции по их использованию.

Если пользователь на сайте впервые, то он должен зарегистрироваться на сайте (рисунки 4.2).

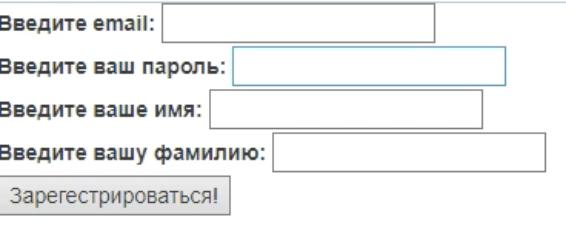


Рисунок 4.2 – Форма регистрации пользователя

После чего пользователь должен корректно заполнить все поля в форме, и нажать “Зарегистрироваться” (рисунки 4.3).

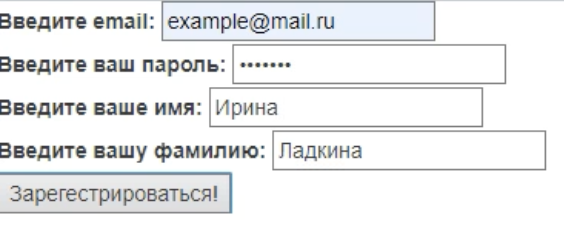


Рисунок 4.3 – Корректно заполненная форма регистрации пользователя

После чего пользователь попадает в личный кабинет, где он видит рекламный банер в центре, и панель навигации в верхней части экрана (рисунки 4.4).

На панели изображены кнопки: Домашняя страница, Мастера, Услуги, Записаться, Мои записи, Информация об аккаунте, О нас и Выход.

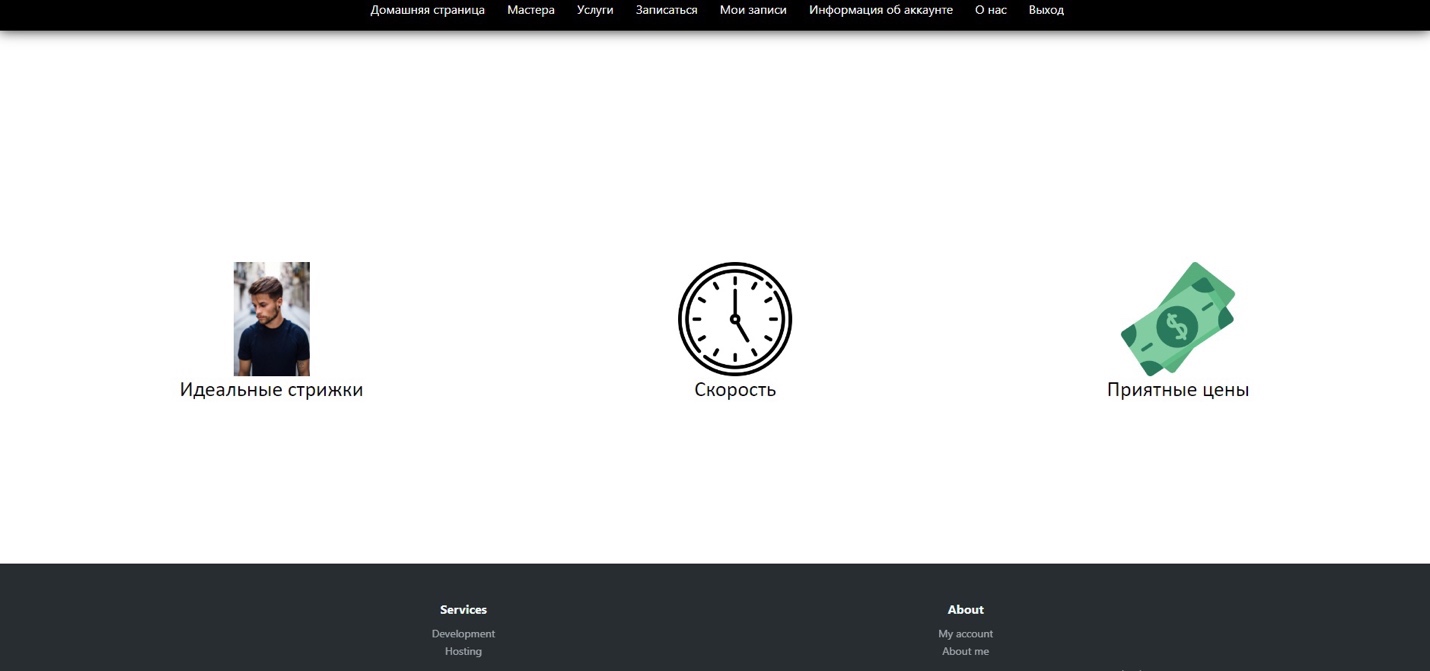


Рисунок 4.4 – Личный кабинет пользователя

При нажатии на кнопку «Домашняя страница» пользователь перейдет на начальную страницу кабинета пользователя (рисунки 4.4). При нажатии на кнопку «Мастера», пользователь попадает на страницу, где может ознакомиться с информацией о парикмахерах или стилистах, работающих в салоне (рисунки 4.5). Здесь представлена информация об их опыте и т.д.

# 

Рисунок 4.5 – Страница «Мастера»

При нажатии на кнопку «Услуги», пользователь попадает на страницу, где представлена информация об услугах парикмахерского салона (рисунки 4.6). Пользователи могут узнать о различных видах стрижек, окрашиваний и других услугах, которые предлагает салон. Здесь также указаны цены на услуги и акции или специальные предложения.

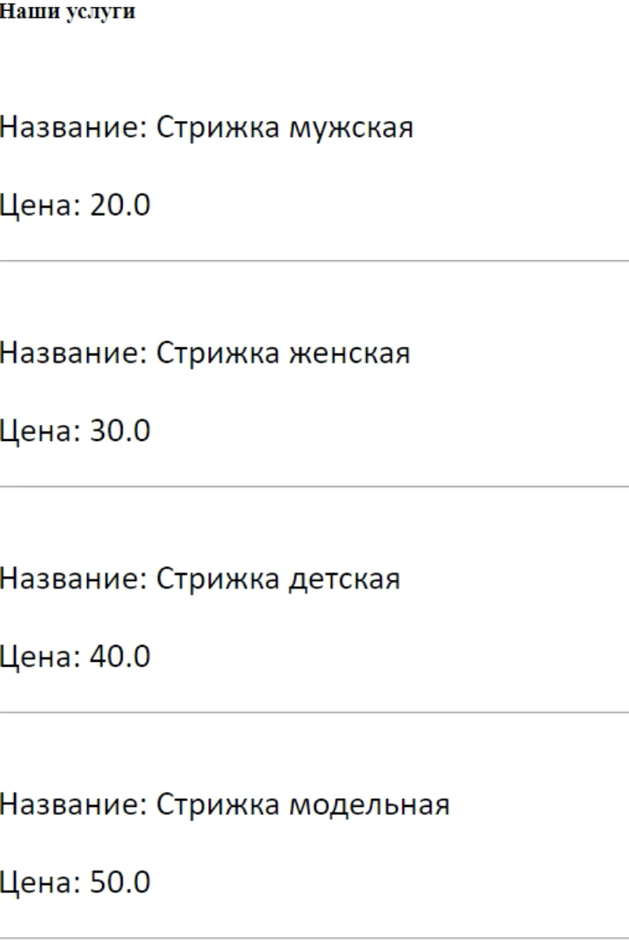


Рисунок 4.6 – Страница «Услуги»

При нажатии на кнопку «Записаться», пользователь попадает на страницу, где представлен интерактивный календарь салона (рисунки 4.7). Эта страница позволяет пользователям записаться на прием к парикмахеру или стилисту. Здесь можно выбрать дату и время визита, а также выбрать мастера и услугу.

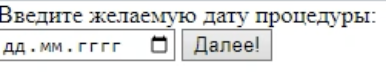


Рисунок 4.7 – Страница «Записаться»

После выбора желаемой даты, нажимаем на кнопку «Далее» (рисунки 4.8).

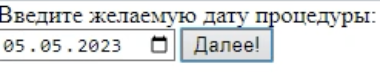


Рисунок 4.8 – Введенная дата записи

И далее появляется список со свободными на эту дату парикмахерами (рисунки 4.9). Так же он видит в какое время у них свободна запись и дату.

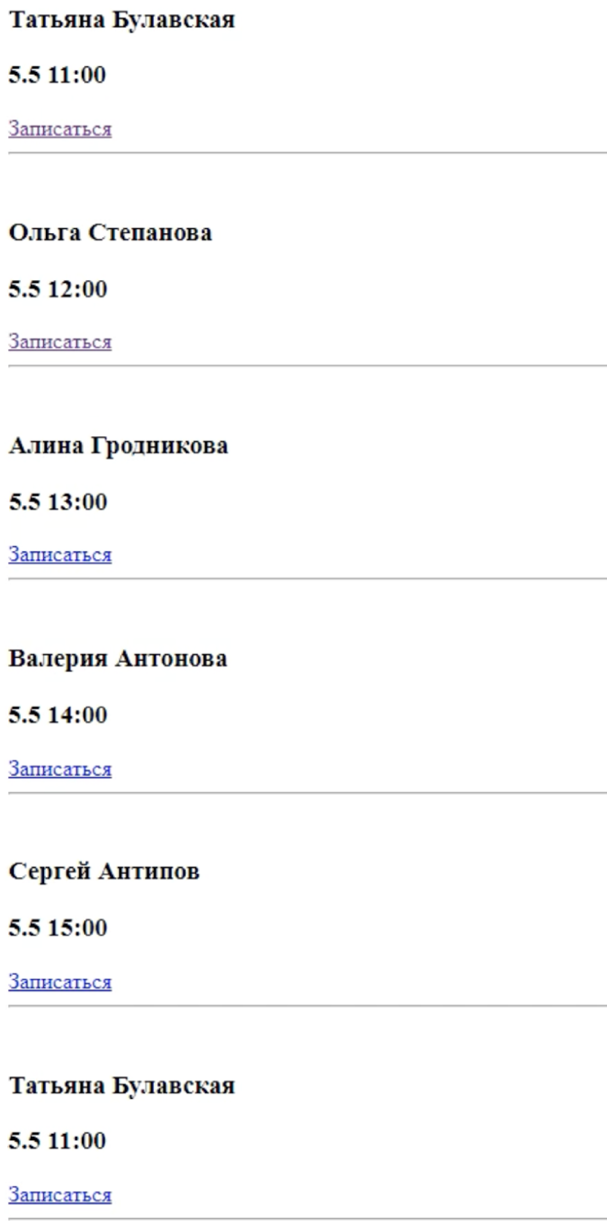


Рисунок 4.9 – Свободные парикмахеры

После выбора подходящего специалиста, необходимо нажать на кнопку «Записаться», после чего пользователь получит подтверждение, в виде текста изображенного на рисунке 4.10.

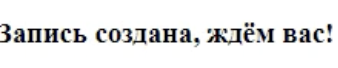


Рисунок 4.10 – Подтверждение записи к специалисту

При нажатии на кнопку «Мои записи», пользователь попадает на страницу, где представлена информация о своих предстоящих или прошедших записях в парикмахерский салон (рисунки 4.11). Здесь можно проверить дату и время визита, а так же тип услуги и парикмахера.

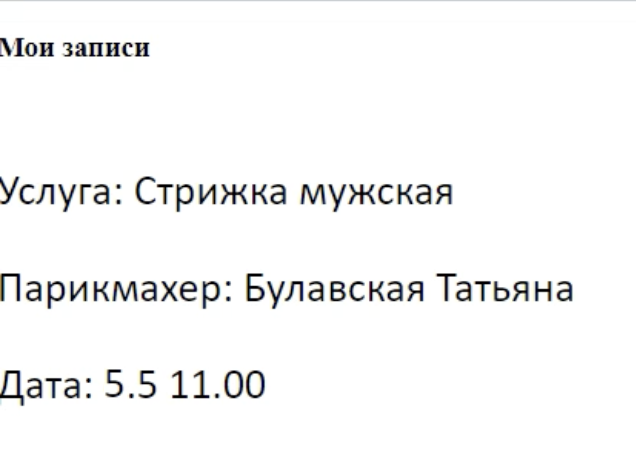


Рисунок 4.11 – Записи пользователя

При нажатии на кнопку «Информация об аккаунте», пользователь попадает на страницу, где представлена информация об учетной записи пользователя (рисунки 4.12). Здесь можно изменить личные данные, такие как имя, фамилия, адрес электронной почты, номер телефона и количество записей.



Рисунок 4.12 – Информация об аккаунте пользователя

При нажатии на кнопку «О нас», пользователь попадает на страницу, где представлена информация о парикмахерском салоне, указан адрес салона, название и контактная информация (рисунки 4.13).

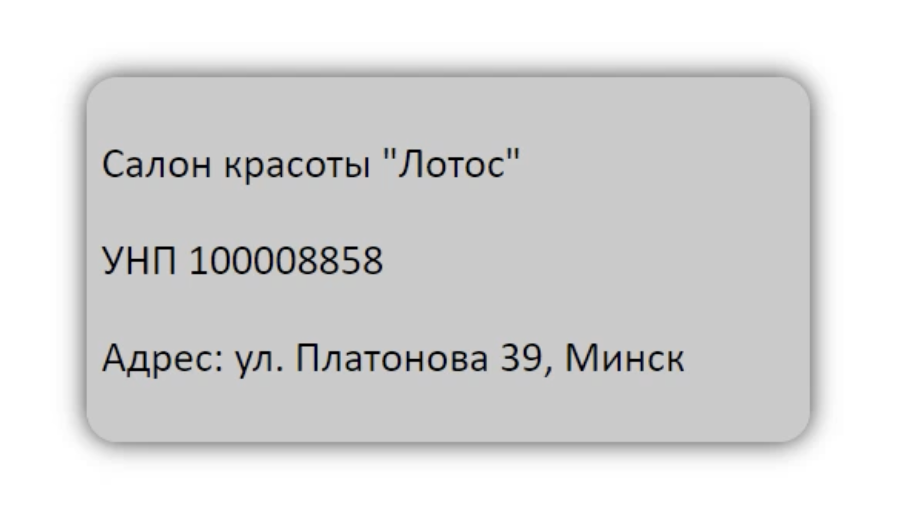


Рисунок 4.13 – Информация о парикмахерской

Кнопка «Выход», позволяет выйти из учетной записи пользователя. После выхода пользователь будет перенаправлен на главную страницу сайта (рисунки 4.1).

После заполняем форму и попадаем в личный кабинет администратора (рисунки 4.14). Где в панели навигации есть такие кнопки, как: Домашняя страница, Парикмахеры, Услуги, Записи, Клиенты, О нас и Выход.

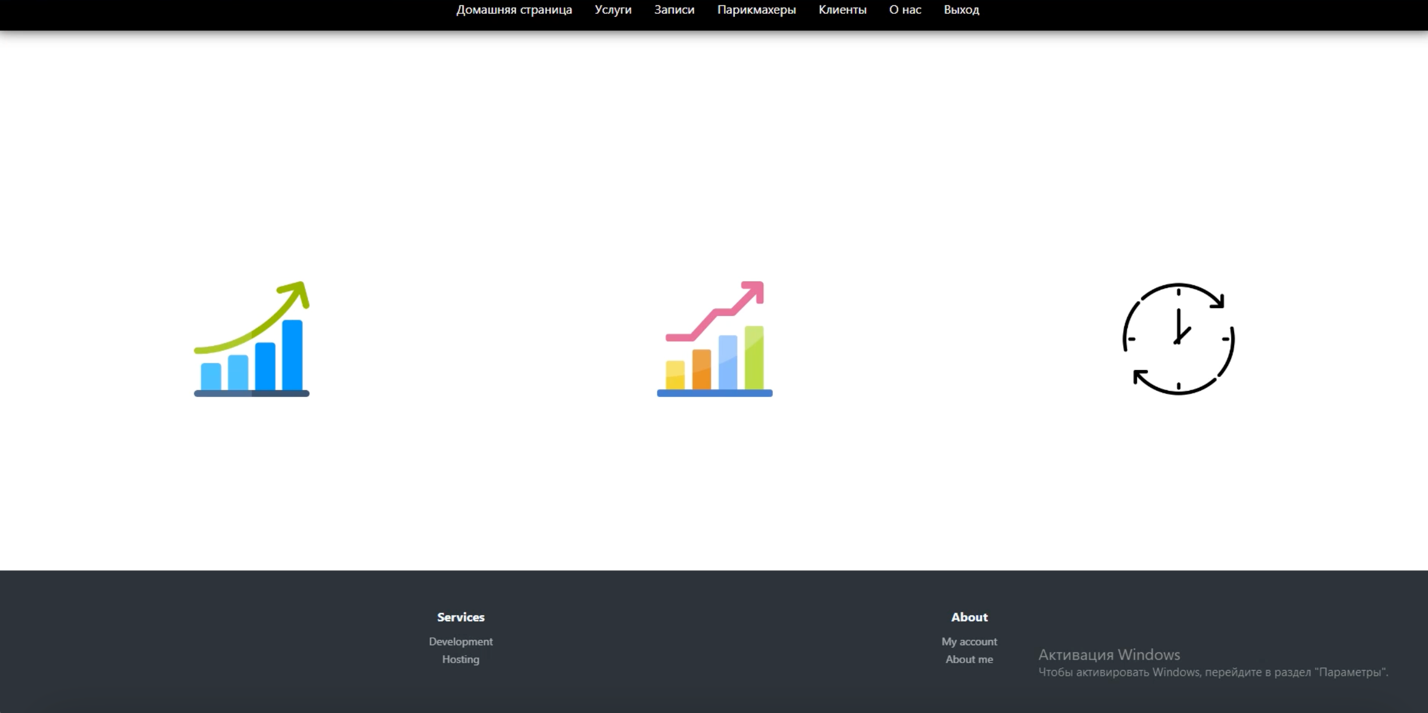


Рисунок 4.14 – Личный кабинет администратора

При нажатии на кнопку «Домашняя страница» администратор перейдет на начальную страницу кабинета администратора (рисунки 4.14). При нажатии на кнопку «Услуги», администратор попадает на страницу, с информацией об услугах парикмахерского салона (рисунки 4.15 – 4.16). Здесь администратор может добавлять или удалять услуги, а также устанавливать цены и акции.

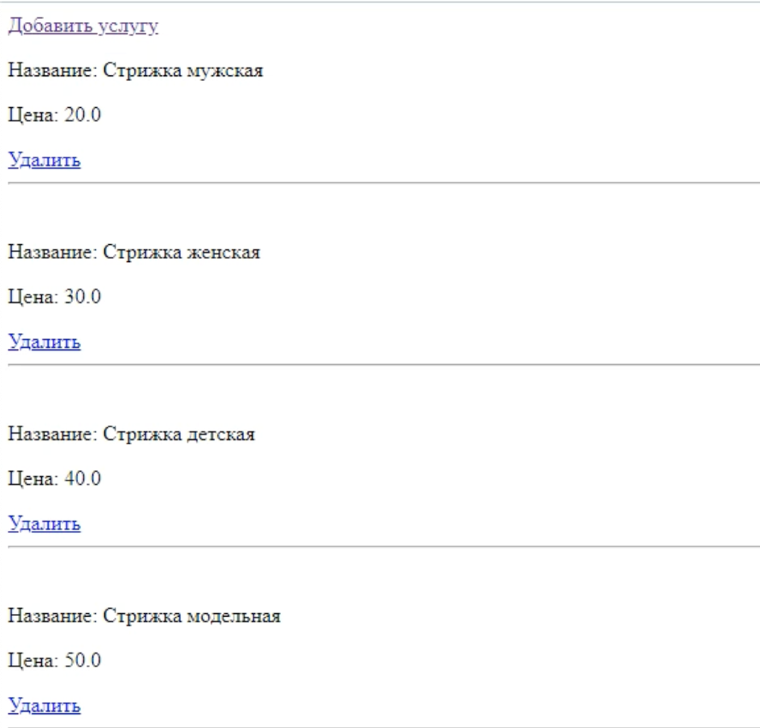


Рисунок 4.15 – Страница «Услуги»

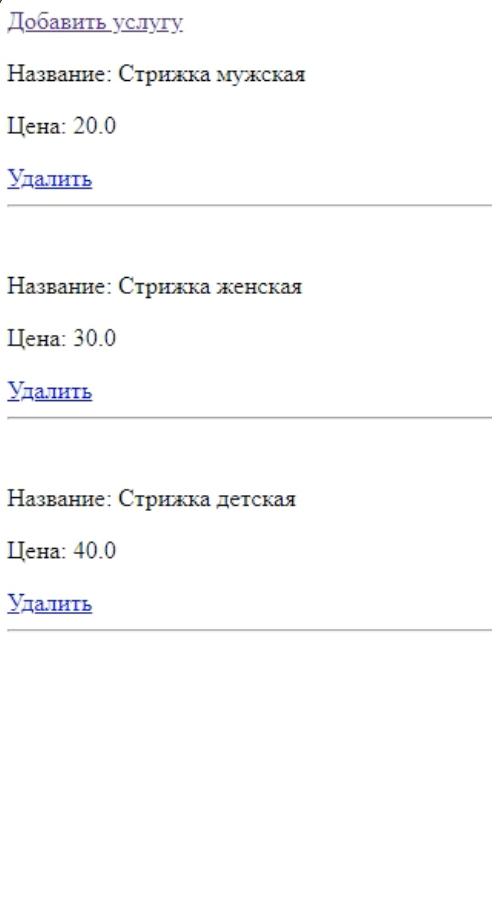


Рисунок 4.16 – Удаление услуги

При нажатии на кнопку «Записи», администратор попадает на страницу, с информацией о записях клиентов в парикмахерский салон (рисунки 4.17 – 4.18). Здесь администратор может просматривать, подтверждать или удалять записи.

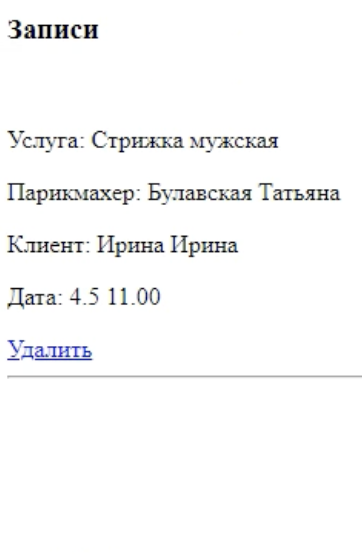


Рисунок 4.17 – Страница «Записи»



Рисунок 4.18 – Удаление записи

При нажатии на кнопку «Парикмахеры», администратор попадает на страницу, с информацией о парикмахерах или стилистах, работающих в салоне (рисунки 4.19). Здесь администратор может добавлять, редактировать или удалять информацию о мастерах.

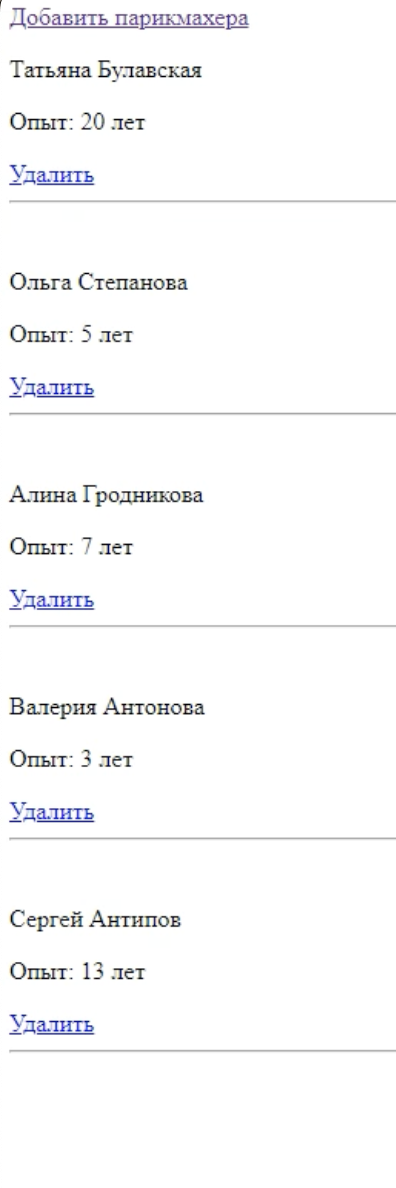


Рисунок 4.19 – Страница «Парикмахеры»

Для того, чтобы добавить парикмахера, необходимо нажать кнопку «Добавить парикмахера» и корректно заполнить форму, появившуюся на странице (рисунки 4.20 – 4.21).

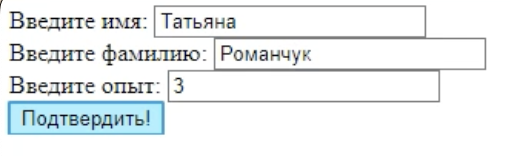


Рисунок 4.20 – Заполнение формы парикмахера



Рисунок 4.21 – Успешное добавление парикмахера

Так же, можно легко удалить парикмахера, нажав на кнопку «Удалить» (рисунки 4.22).

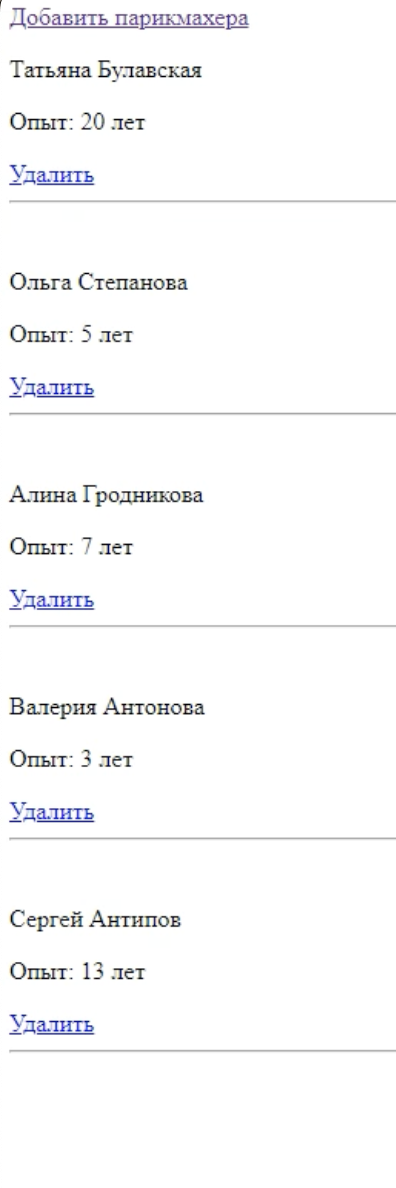


Рисунок 4.22 – Удаление парикмахера

При нажатии на кнопку «Клиенты», администратор попадает на страницу, с информацией о клиентах парикмахерского салона (рисунки 4.23). Здесь администратор может просматривать информацию о клиентах, их записях и удалять их.

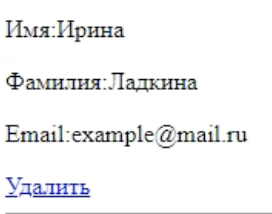


Рисунок 4.23 – Страница «Клиенты»

При нажатии на кнопку «О нас», администратор попадает на страницу, где представлена информация о парикмахерском салоне, указан адрес салона, название и контактная информация (рисунки 4.24).

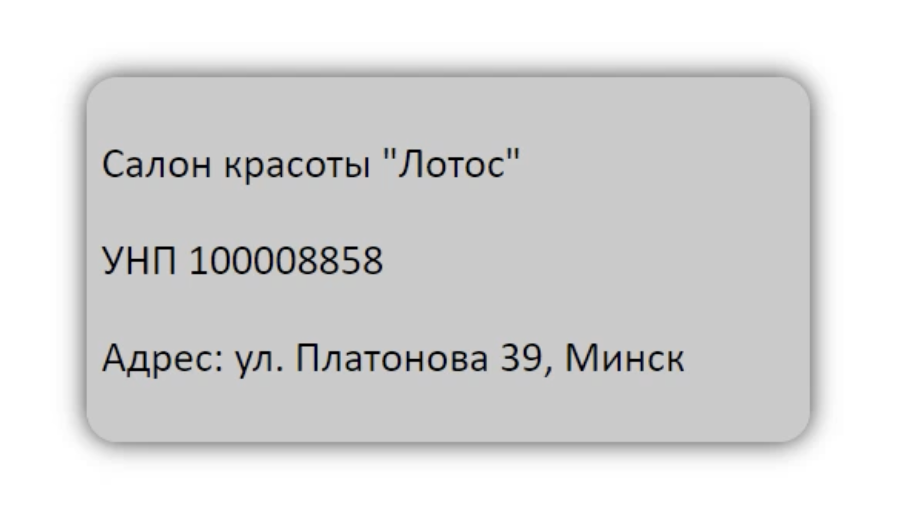


Рисунок 4.24 – Информация о парикмахерской

Кнопка «Выход», позволяет выйти из учетной записи пользователя. После выхода пользователь будет перенаправлен на главную страницу сайта (рисунки 4.1).

В целом, функциональные возможности и инструкции по установке и использованию программного средства позволяют пользователям быстро овладеть его функционалом и эффективно работать с программным средством, что делает ее более доступной и привлекательной для использования.

# ЗАКЛЮЧЕНИЕ

В ходе данного курсового проекта были достигнуты поставленные цели и реализованы необходимые задачи. Так, например, была изучена область деятельности парикмахерского салона. А результатом выполнения проекта стало программное средство, которое позволяет данной компании автоматизировать большое количество процессов, увеличить производительность и др.

Кроме создания приложения можно отметить, что результатом изучения процесса деятельности парикмахерского салона стала диаграмма IDEF0, состоящая из контекстной диаграммы и декомпозиций 4 уровней.

Изученные деятельность организации и процесс принятия решения позволили разработать базу данных, хранящую данные о пользователях, товарах, находящихся на интернет-площадке и др.

Реализованы проверки на различные виды ввода, и при неправильном вводе пользователь получает оповещение-подсказку, помогающую ему сориентироваться и исправить ввод.

Для разработки программного средства были смоделированы алгоритмы, реализующие бизнес-логику программного средства. Для улучшения взаимодействия пользователя с программным средством были предоставлены руководства по развёртыванию программного средства и его работе.

Программа оснащена понятным и удобным интерфейсом, а также простой системой навигации и необходимыми функциями для решения основной задачи курсового проекта. В данном приложении реализованы базовые принципы объектно-ориентированного программирования, использованы стандартные и пользовательские функции, была реализована работа с базами данных, взаимодействие между сервером и клиентом.

Для улучшения взаимодействия пользователя с программным средством были предоставлены руководства по развёртыванию программного средства и его работе.

По завершении разработки было проведено полное тестирование, в результате которого не было выявлено критических сбоев в работе программного средства, что подтверждает работоспособность программы.

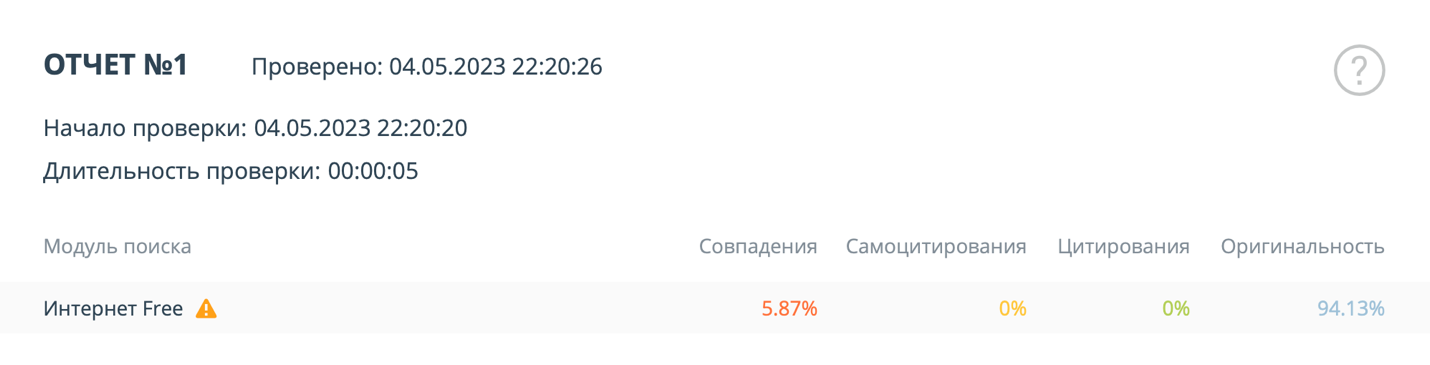
**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Клиент-сервер шаг-за-шагом [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/330676.
2. Прохоров Ю.К., Фролов В. В. Управленческие решения: Учебное пособие. – 2-е изд., испр. и доп. – СПб: СПбГУ ИТМО, 2011 – 138 с.
3. Шилдт, Герберт. Java. Полное руководство, 10-е изд.: Пер. с англ. -СПб. ООО "Альфа­книга"; 2019. - 1488 с.
4. REST API с использованием Spring Security и JWT [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/545610/.
5. Эккель Б. Философия Java. 4-е полное изд. – СПб.: Питер, 2015. – 1168 c.: ил. – (Серия «Классика computer science»).
6. Руководство по языку программирования Kotlin [Электронный ресурс]. Режим доступа: https://metanit.com/java/tutorial.
7. Фаулер M. UML. Основы, 3е издание. – Пер. с англ. – СПб: Символ Плюс, 2004 – 192 с.
8. Репин В. В. Разработка архитектуры бизнес-процессов компании. М.: Манн, Иванов и Фербер, 2019. – 142 с.
9. Базы данных [Электронный ресурс]. Режим доступа: <https://ru.hostings.info/schools/bazy-dannyh.html>.
10. Черемных С.В., Семенов И.О. Моделирование и анализ систем. IDEF – технологии: практикум. М.: Финансы и статистика, 2006. – 188 с.
11. Фаулер M. UML. Основы, 3е издание. – Пер. с англ. – СПб: Символ Плюс, 2004 – 192 с.
12. Черемных С.В., Семенов И.О. Моделирование и анализ систем. IDEF – технологии: практикум. М.: Финансы и статистика, 2006. – 188 с.
13. Ларман, К. Применение UML 2.0 и шаблонов проектирования. / К. Ларман - М.: Вильямс, 2016. – 736 с.
14. Диаграмма вариантов использования [Электронный ресурс]. – Режим доступа: http://www.informicus.ru/default.aspx.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

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



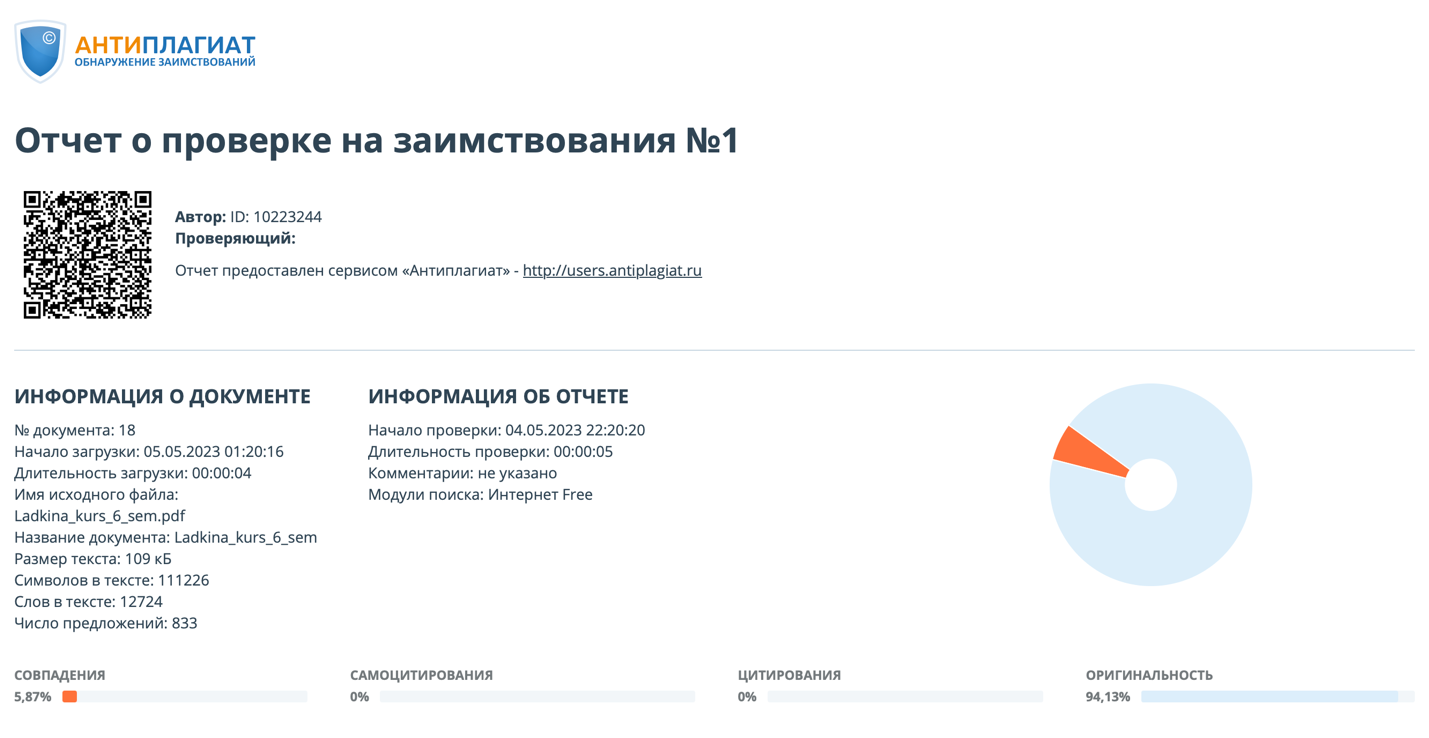


Рисунок А.1 – Отчет о проверке на заимствования в системе «Антиплагиат»

# ПРИЛОЖЕНИЕ Б

# (обязательное)

# Листинг скрипта генерации базы данных

create schema if not exists `barbershop\_automation\_system`;

use `barbershop\_automation\_system`;

create table if not exists `admins`(

`id` int not null auto\_increment,

`login` varchar(50) not null,

`password` varchar(50) not null,

constraint `PK\_admins` primary key (`id` ASC)

);

create table if not exists `clients`(

`id` int not null auto\_increment,

`email` varchar(50) not null,

`name` varchar(50) not null,

`surname` varchar(50) not null,

`password` varchar(50) not null,

constraint `PK\_clients` primary key (`id` ASC)

);

create table if not exists `hairdressers`(

`id` int not null auto\_increment,

`experience` int not null,

`name` varchar(50) not null,

`surname` varchar(50) not null,

constraint `PK\_hairdressers` primary key (`id` ASC)

);

create table if not exists `services`(

`id` int not null auto\_increment,

`cost` float not null,

`name` varchar(50) not null,

constraint `PK\_services` primary key (`id` ASC)

);

create table if not exists `sessions`(

`id` int not null auto\_increment,

`dateTime` datetime not null,

`hairdresserId` int not null,

`clientId` int not null,

`serviceId` int not null,

constraint `PK\_sessions` primary key (`id` ASC),

constraint `FK\_sessions\_hairdressers` FOREIGN KEY(`hairdresserId`) REFERENCES `hairdressers` (`id`),

Продолжение приложения Б

constraint `FK\_sessions\_clients` FOREIGN KEY(`clientId`) REFERENCES `clients` (`id`),

constraint `FK\_sessions\_services` FOREIGN KEY(`serviceId`) REFERENCES `services` (`id`)

);

**ПРИЛОЖЕНИЕ В**

**(обязательное)**

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

@Controller

@RequestMapping("/register")

public class RegisterController {

private final ClientsService clientsService;

private final AdminsService adminsService;

@Autowired

public RegisterController(ClientsService clientsService, AdminsService adminsService) {

this.clientsService = clientsService;

this.adminsService = adminsService;

}

@GetMapping("/index")

public String index(){

return "register/index";

}

@PostMapping(value = "/submit")

public String loginPost(@ModelAttribute ClientViewModel userDTO) {

try {

if (clientsService.isLoginExists(userDTO.getEmail()) || adminsService.isLoginExists(userDTO.getEmail())){

return "register/loginExists";

}

clientsService.createOrUpdateClient(userDTO);

} catch (Exception e) {

return "register/errorInput";

}

return "client/home";

}

}

@Controller

@RequestMapping(value = "/login")

public class LoginController {

private final ClientsService clientsService;

private final AdminsService adminsService;

@Autowired

public LoginController(ClientsService clientsService, AdminsService adminsService) {

this.clientsService = clientsService;

Продолжение приложения В

this.adminsService = adminsService;

}

@GetMapping("/index")

public String index() {

return "login/index";

}

@PostMapping(value = "/submit")

public String loginPost(HttpServletResponse response, @ModelAttribute ClientViewModel userDTO) {

try {

var client = clientsService.find(userDTO.getEmail(), userDTO.getPassword());

var typeCookie = new Cookie("typeCookie", String.valueOf(2));

typeCookie.setMaxAge(7 \* 24 \* 60 \* 60);

typeCookie.setDomain("localhost");

typeCookie.setPath("/client");

var idCookie = new Cookie("idCookie", String.valueOf(client.getId()));

idCookie.setMaxAge(7 \* 24 \* 60 \* 60);

idCookie.setDomain("localhost");

idCookie.setPath("/client");

response.addCookie(typeCookie);

response.addCookie(idCookie);

return "redirect:/client/home";

} catch (Exception ignored) {

System.out.println(ignored);

}

try {

var admin = adminsService.find(userDTO.getEmail(), userDTO.getPassword());

var typeCookie = new Cookie("typeCookie", String.valueOf(1));

typeCookie.setMaxAge(7 \* 24 \* 60 \* 60);

typeCookie.setDomain("localhost");

typeCookie.setPath("/admin");

var idCookie = new Cookie("idCookie", String.valueOf(admin.getId()));

idCookie.setMaxAge(7 \* 24 \* 60 \* 60);

idCookie.setDomain("localhost");

idCookie.setPath("/admin");

Продолжение приложения В

response.addCookie(typeCookie);

response.addCookie(idCookie);

return "redirect:/admin/home";

} catch (Exception ignored) {

System.out.println(ignored);

}

return "login/loginError";

}

}

@Controller

@RequestMapping("/client")

public class ClientsController {

private final ClientsService clientsService;

private final SessionsService sessionsService;

private final HairdressersService hairdressersService;

private final ServiceModelService serviceModelService;

public ClientsController(ClientsService clientsService, SessionsService sessionsService, HairdressersService hairdressersService, ServiceModelService serviceModelService) {

this.clientsService = clientsService;

this.sessionsService = sessionsService;

this.hairdressersService = hairdressersService;

this.serviceModelService = serviceModelService;

}

public Date localDateToDate(LocalDate dateToConvert) {

return java.util.Date.from(dateToConvert.atStartOfDay()

.atZone(ZoneId.systemDefault())

.toInstant());

}

public LocalDateTime dateToLocalDate(Date dateToConvert) {

return dateToConvert.toInstant()

.atZone(ZoneId.systemDefault())

.toLocalDateTime();

}

@GetMapping("/home")

public String home() throws Exception {

return "client/home";

}

Продолжение приложения В

@GetMapping("/viewServices")

public String viewServices(Model model) throws Exception {

var services = serviceModelService.getAllServices();

model.addAttribute("services", services);

return "client/clientServices";

}

@GetMapping("/viewMySessions")

public String viewMySessions(HttpServletRequest request, Model model) throws Exception {

var cookies = request.getCookies();

var id = Long.parseLong(cookies[1].getValue());

var sessions = sessionsService.getAllSessions();

var buf = sessions.stream().filter(x -> x.getClient().getId() == id).toList();

model.addAttribute("sessions", buf);

return "client/clientSessions";

}

@GetMapping("/viewMasters")

public String viewMasters(Model model) throws Exception {

var masters = hairdressersService.getAllHairdressers();

model.addAttribute("masters", masters);

return "client/masters";

}

@GetMapping("/createSession/{h}.{s}.{m}.{d}.{hh}")

public String createSession(HttpServletRequest request, @PathVariable int d,

@PathVariable long h,

@PathVariable int hh,

@PathVariable int m,

@PathVariable int s) throws Exception {

var cookies = request.getCookies();

var client = clientsService.getClientById(Long.parseLong(cookies[1].getValue()));

var hairdresser = hairdressersService.getHairdresserById(h);

var service = serviceModelService.getServiceById(s);

var monthValue = m;

var day = d;

var hour = hh;

var dt = LocalDateTime.of(2023, monthValue, day, hour, 0);

var sessionVm = new SessionViewModel(0, client, hairdresser, service, dt);

sessionsService.createOrUpdateSession(sessionVm);

return "client/sessionCreated";

}

@GetMapping("/procedureDate")

Продолжение приложения В

public String procedureDate() {

return "client/chooseProcedureDate";

}

@PostMapping("/viewFreeServices")

public String viewFreeServices(HttpServletRequest request, Model model, @ModelAttribute Date dateTime) throws Exception {

var cookies = request.getCookies();

var id = Long.parseLong(cookies[1].getValue());

var client = clientsService.getClientById(id);

var localDateTime = dateToLocalDate(dateTime);

var services = serviceModelService.getAllServices();

var sessionsOnDate = sessionsService.getAllSessions().stream().filter(

x -> x.getDateTime().getDayOfMonth() == localDateTime.getDayOfMonth()

&& x.getDateTime().getMonthValue() == localDateTime.getMonthValue()

&& x.getDateTime().getYear() == localDateTime.getYear()).toList();

var hairdressers = hairdressersService.getAllHairdressers();

var startTime = 11;

var freeSessions = new ArrayList<SessionViewModel>();

for (var s : services) {

var availableSessions = new ArrayList<SessionViewModel>();

var unavailableSessions = sessionsOnDate.stream().filter(x -> x.getService().getId() == s.getId()).toList();

var mentionedHairdressers = new ArrayList<Long>();

for (int i = 0; i < 5; i++) {

var sessionTime = startTime + i;

if (unavailableSessions.stream().anyMatch(x -> x.getDateTime().getHour() == sessionTime)) {

continue;

}

HairdresserViewModel hd = null;

for (var item : hairdressers) {

if (sessionsOnDate.stream().anyMatch(

x -> x.getHairdresser().getId() == item.getId() && x.getDateTime().getHour() == sessionTime)) {

continue;

}

if (mentionedHairdressers.stream().anyMatch(k -> k == item.getId())) {

continue;

}

Продолжение приложения В

hd = item;

mentionedHairdressers.add(hd.getId());

break;

}

var dt = LocalDateTime.of(dateTime.getYear() + 1900, dateTime.getMonth() + 1, dateTime.getDay(), sessionTime, 0);

availableSessions.add(new SessionViewModel(0, client, hd, s, dt));

}

freeSessions.addAll(availableSessions);

}

model.addAttribute("freeSessions", freeSessions);

return "client/viewFreeServices";

}

@GetMapping("/profileInfo")

public String profileInfo(HttpServletRequest request, Model model) throws Exception {

var cookies = request.getCookies();

var client = clientsService.getClientById(Long.parseLong(cookies[1].getValue()));

model.addAttribute("user", client);

var amountOfRecords = sessionsService.getAllSessions().stream().filter(x->x.getClient().getId() == client.getId()).count();

model.addAttribute("amountOfRecords", amountOfRecords);

return "client/profileInfo";

}

@GetMapping("/aboutUs")

public String aboutUs() {

return "client/aboutUs";

}

}

@Controller

@RequestMapping("/admin")

public class AdminsController {

private final ClientsService clientsService;

private final SessionsService sessionsService;

private final HairdressersService hairdressersService;

private final ServiceModelService serviceModelService;

public AdminsController(ClientsService clientsService, SessionsService sessionsService, HairdressersService hairdressersService, ServiceModelService serviceModelService) {

Продолжение приложения В

this.clientsService = clientsService;

this.sessionsService = sessionsService;

this.hairdressersService = hairdressersService;

this.serviceModelService = serviceModelService;

}

@GetMapping("/home")

public String home(HttpServletRequest request) throws Exception {

return "admin/home";

}

@GetMapping("/getAllSessions")

public String getAllSessions(Model model) throws Exception {

var sessions = sessionsService.getAllSessions();

model.addAttribute("sessions", sessions);

return "admin/sessions";

}

@GetMapping("/deleteSession/{id}")

public String deleteSession(@PathVariable long id) throws Exception {

sessionsService.deleteSession(id);

return "redirect:/admin/getAllSessions";

}

@GetMapping("/getAllClients")

public String getAllClients(Model model) throws Exception {

var clients = clientsService.getAllClients();

model.addAttribute("clients", clients);

return "admin/clients";

}

@GetMapping("/deleteClient/{id}")

public String deleteClient(@PathVariable long id) throws Exception {

var sessions = sessionsService.getAllSessions().stream().filter(x -> x.getClient().getId() == id).toList();

for (var item : sessions) {

sessionsService.deleteSession(item.getId());

}

clientsService.deleteClient(id);

return "redirect:/admin/getAllClients";

}

@GetMapping("/getAllHairdressers")

public String getAllHairdressers(Model model) throws Exception {

var hairdressers = hairdressersService.getAllHairdressers();

model.addAttribute("hairdressers", hairdressers);

return "admin/hairdressers";

}

@GetMapping("/createHairdresser")

Продолжение приложения В

public String createOrUpdateHairdresser() throws Exception {

return "admin/createHairdresser";

}

@PostMapping("/createHairdresser")

public String createOrUpdateHairdresser(@ModelAttribute HairdresserViewModel hairdresser) throws Exception {

hairdressersService.createOrUpdateHairdresser(hairdresser);

return "redirect:/admin/getAllHairdressers";

}

@GetMapping("/deleteHairdresser/{id}")

public String deleteHairdresser(@PathVariable long id) throws Exception {

hairdressersService.deleteHairdresser(id);

return "redirect:/admin/getAllHairdressers";

}

@GetMapping("/getAllServices")

public String getAllServices(Model model) throws Exception {

var services = serviceModelService.getAllServices();

model.addAttribute("services", services);

return "admin/services";

}

@GetMapping("/createService")

public String createService() throws Exception {

return "admin/createService";

}

@PostMapping("/createService")

public String createService(@ModelAttribute ServiceViewModel model) throws Exception {

serviceModelService.createOrUpdateService(model);

return "redirect:/admin/getAllServices";

}

@GetMapping("/deleteService/{id}")

public String deleteService(@PathVariable long id) throws Exception {

var sessions = sessionsService.getAllSessions().stream().filter(x -> x.getService().getId() == id).toList();

for (var item : sessions) {

sessionsService.deleteSession(item.getId());

}

serviceModelService.deleteService(id);

return "redirect:/admin/getAllServices";

}

}

@Service

public class SessionsService {

Продолжение приложения В

private final SessionsRepository sessionsRepository;

private final DozerBeanMapper mapper;

@Autowired

public SessionsService(SessionsRepository adminRepository) {

this.sessionsRepository = adminRepository;

mapper = new DozerBeanMapper();

}

public List<SessionViewModel> getAllSessions() throws Exception {

List<SessionViewModel> list = new LinkedList<>();

var sessions = sessionsRepository.findAll();

for (var item: sessions) {

var sessionVM = new SessionViewModel();

sessionVM.setId(item.getId());

sessionVM.setService(mapper.map(item.getService(), ServiceViewModel.class));

sessionVM.setHairdresser(mapper.map(item.getHairdresser(), HairdresserViewModel.class));

sessionVM.setClient(mapper.map(item.getClient(), ClientViewModel.class));

sessionVM.setDateTime(item.getDateTime());

list.add(sessionVM);

}

//sessionsRepository.findAll().forEach(x -> list.add(mapper.map(x, SessionViewModel.class)));

return list;

}

public SessionViewModel getAdminById(long id) throws Exception {

var obj = sessionsRepository.findById(id);

return mapper.map(obj, SessionViewModel.class);

}

public void createOrUpdateSession(SessionViewModel objDTO) throws Exception {

//var obj = mapper.map(objDTO, Session.class);

var obj = new Session();

obj.setId(objDTO.getId());

obj.setService(mapper.map(objDTO.getService(), com.example.barbershopautomationsystem.DataLayer.Models.Service.class));

obj.setHairdresser(mapper.map(objDTO.getHairdresser(), Hairdresser.class));

obj.setClient(mapper.map(objDTO.getClient(), Client.class));

obj.setDateTime(objDTO.getDateTime());

sessionsRepository.save(obj);

}

public void deleteSession(long id) throws Exception {

Продолжение приложения В

sessionsRepository.deleteById(id);

}

}

@Service

public class ServiceModelService {

private final ServicesRepository servicesRepository;

private final DozerBeanMapper mapper;

@Autowired

public ServiceModelService(ServicesRepository servicesRepository) {

this.servicesRepository = servicesRepository;

mapper = new DozerBeanMapper();

}

public List<ServiceViewModel> getAllServices() throws Exception {

List<ServiceViewModel> list = new LinkedList<>();

servicesRepository.findAll().forEach(x -> list.add(mapper.map(x, ServiceViewModel.class)));

return list;

}

public ServiceViewModel getServiceById(long id) throws Exception {

var obj = servicesRepository.findById(id).get();

return mapper.map(obj, ServiceViewModel.class);

}

public void createOrUpdateService(ServiceViewModel objDTO) throws Exception {

var obj = mapper.map(objDTO, com.example.barbershopautomationsystem.DataLayer.Models.Service.class);

servicesRepository.save(obj);

}

public void deleteService(long id) throws Exception{

servicesRepository.deleteById(id);

}

}

@Service

public class HairdressersService {

private final HairdressersRepository hairdressersRepository;

private final DozerBeanMapper mapper;

@Autowired

public HairdressersService(HairdressersRepository hairdressersRepository)

Продолжение приложения В

{

this.hairdressersRepository = hairdressersRepository;

mapper = new DozerBeanMapper();

}

public List<HairdresserViewModel> getAllHairdressers() throws Exception {

List<HairdresserViewModel> list = new LinkedList<>();

hairdressersRepository.findAll().forEach(x -> list.add(mapper.map(x, HairdresserViewModel.class)));

return list;

}

public HairdresserViewModel getHairdresserById(long id) throws Exception {

var obj = hairdressersRepository.findById(id).get();

return mapper.map(obj, HairdresserViewModel.class);

}

public void createOrUpdateHairdresser(HairdresserViewModel objDTO) throws Exception {

var obj = mapper.map(objDTO, Hairdresser.class);

hairdressersRepository.save(obj);

}

public void deleteHairdresser(long id) throws Exception{

hairdressersRepository.deleteById(id);

}

}

@Service

public class ClientsService {

private final ClientRepository clientRepository;

private final DozerBeanMapper mapper;

@Autowired

public ClientsService(ClientRepository clientRepository) {

this.clientRepository = clientRepository;

mapper = new DozerBeanMapper();

}

public List<ClientViewModel> getAllClients() throws Exception {

List<ClientViewModel> list = new LinkedList<>();

clientRepository.findAll().forEach(x -> list.add(mapper.map(x, ClientViewModel.class)));

return list;

}

Продолжение приложения В

public ClientViewModel getClientById(long id) throws Exception {

var client = clientRepository.findById(id).get();

return mapper.map(client, ClientViewModel.class);

}

public void createOrUpdateClient(ClientViewModel clientViewModel) throws Exception {

var client = mapper.map(clientViewModel, Client.class);

clientRepository.save(client);

}

public void deleteClient(long id) throws Exception{

clientRepository.deleteById(id);

}

public ClientViewModel find(String login, String password) throws Exception{

var res = clientRepository.findAll().stream().filter(x-> x.getEmail().equals(login) && x.getPassword().equals(password)).toList();

if (res.size() != 1) throw new Exception();

return mapper.map(res.get(0), ClientViewModel.class);

}

public boolean isLoginExists(String login){

var res = clientRepository.findAll();

for (var i: res) {

if (i.getEmail().equals(login)) return true;

}

return false;

}

}

@Service

public class AdminsService {

private final AdminRepository adminRepository;

private final DozerBeanMapper mapper;

@Autowired

public AdminsService(AdminRepository adminRepository) {

this.adminRepository = adminRepository;

mapper = new DozerBeanMapper();

}

public List<AdminViewModel> getAllAdmins() throws Exception {

List<AdminViewModel> list = new LinkedList<>();

adminRepository.findAll().forEach(x -> list.add(mapper.map(x, AdminViewModel.class)));

return list;

Продолжение приложения В

}

public AdminViewModel getAdminById(long id) throws Exception {

var obj = adminRepository.findById(id);

return mapper.map(obj, AdminViewModel.class);

}

public void createOrUpdateAdmin(AdminViewModel objDTO) throws Exception {

var obj = mapper.map(objDTO, Admin.class);

adminRepository.save(obj);

}

public void deleteAdmin(long id) throws Exception{

adminRepository.deleteById(id);

}

public AdminViewModel find(String login, String password) throws Exception{

var res = adminRepository.findAll().stream().filter(x-> x.getLogin().equals(login) && x.getPassword().equals(password)).toList();

if (res.size() != 1) throw new Exception();

return mapper.map(res.get(0), AdminViewModel.class);

}

public boolean isLoginExists(String login){

var res = adminRepository.findAll();

for (var i: res) {

if (i.getLogin().equals(login)) return true;

}

return false;

}

}

**ПРИЛОЖЕНИЕ Г  
(обязательное)**

**Схема алгоритма создания графика**

****

Рисунок Г.1 – Схема алгоритма создания графика

Продолжение приложения Г

**Схема алгоритма обработки заявки**

****

Рисунок Г.2 – Схема алгоритма обработки заявки

**ПРИЛОЖЕНИЕ Д**

**(обязательное)**

**Диаграмма классов**

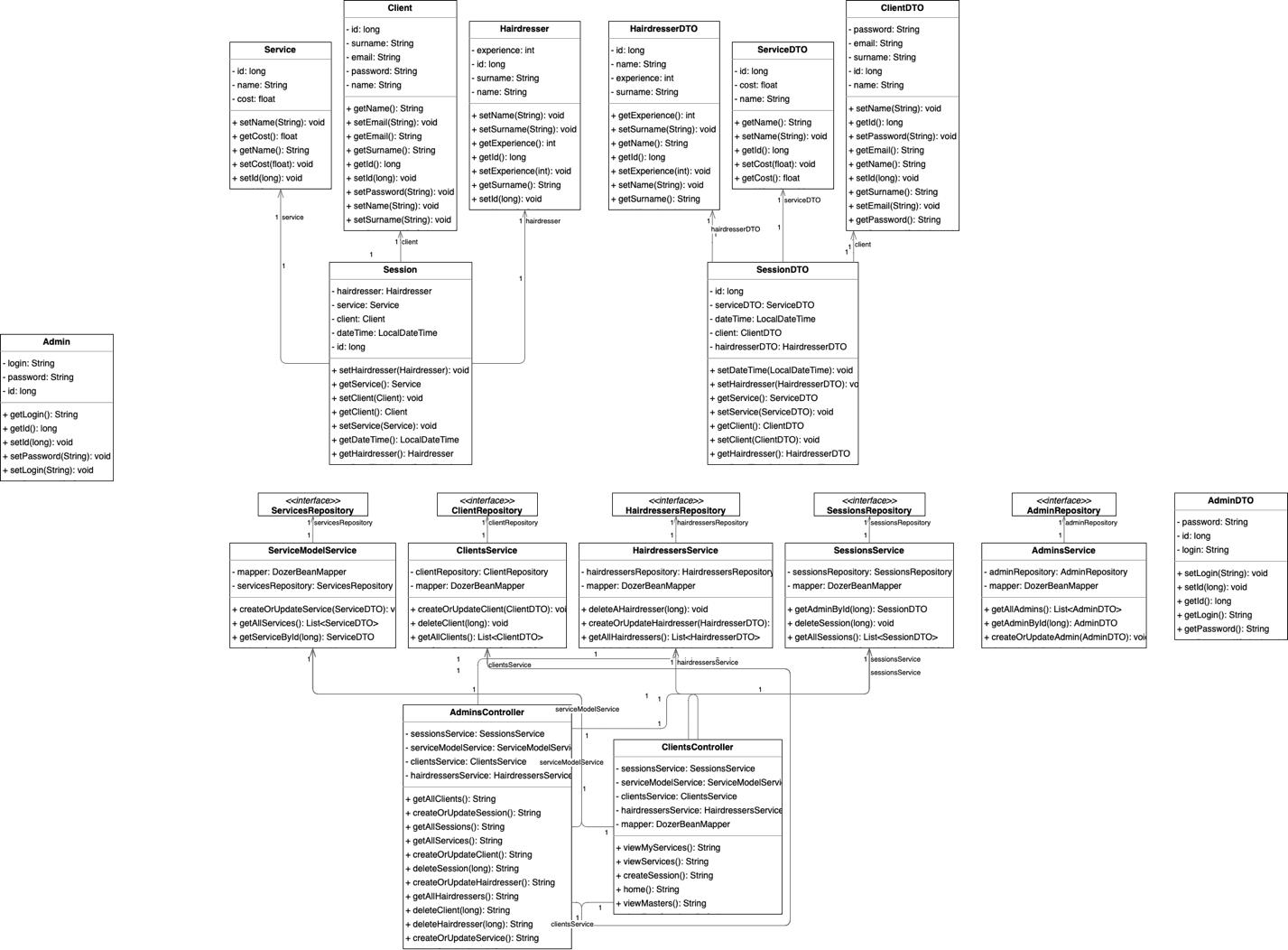
****

Рисунок Д.1 – Диаграмма классов приложения

# Ведомость документов курсового проекта

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | | | | *Дополнительные сведения* | | | |
|  | | | | | Текстовые документы | | | |  | | | |
|  | | | | |  | | | |  | | | |
| БГУИР КП 1-40 05 01-02 012 ПЗ | | | | | Пояснительная записка | | | | 89 с. | | | |
|  | | | | |  | | | |  | | | |
|  | | | | |  | | | |  | | | |
|  | | | | |  | | | |  | | | |
|  | | | | | Графические документы | | | |  | | | |
|  | | | | |  | | | |  | | | |
| ГУИР.425700.001ПД | | | | | Чертеж «IDEF0-модель процессов предметной области» | | | | Формат А4 | | | |
|  | | | | |  | | | |  | | | |
| ГУИР.425700.002ПД | | | | | Чертеж «Схема алгоритма, реализующая основную бизнес-логику программного средства» | | | | Формат А4 | | | |
|  | | | | |  | | | |  | | | |
| ГУИР.425700.003ПЛ | | | | | Плакат «UML диаграмма классов» | | | | Формат А4 | | | |
|  | | | | |  | | | |  | | | |
| ГУИР.425700.004ПЛ | | | | | Плакат «Модели представления программного средства» | | | | Формат А4 | | | |
|  | | | | |  | | | |  | | | |
| ГУИР.425700.005ПЛ | | | | | Плакат «Скриншоты рабочих окон программного средства» | | | | Формат А4 | | | |
|  | | | | |  | | | |  | | | |
|  | | | | |  | | | |  | | | |
|  | | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КП 1-40 05 01-02 012 Д1 | | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| *Изм* | *Л.* | *№ докум.* | *Подп* | *Дата* | *Программное средство для автоматизации работы парикмахерской*  *Ведомость курсового проекта* |  | | | | *Лист* | | *Листов* |
| *Разраб.* | | *Ладкина* |  |  |  | *Т* |  | | | *89* | *89* |
| *Пров.* | | *Сильванович* |  |  | *Кафедра ЭИ*  *гр. 072304* | | | | | | |
|  | |  |  |  |
|  | |  |  |  |

