Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Распределенные системы обработки информации»

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

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«Автоматизированная система работы груминг салона»**

|  |  |
| --- | --- |
|  | Выполнил студент группы 074002  Рощина Ирина Сергеевна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc134462146)

[1 Анализ и моделирование предметной области программного средства 7](#_Toc134462147)

[1.1 Описание предметной области 7](#_Toc134462148)

[1.2 Разработка функциональной модели предметной области 8](#_Toc134462149)

[1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 11](#_Toc134462150)

[1.4 Разработка информационной модели предметной области 13](#_Toc134462151)

[1.5 UML-модели представления программного средства и их описания 15](#_Toc134462152)

[2 Проектирование и конструирование программного средства 18](#_Toc134462153)

[2.1 Постановка задачи 18](#_Toc134462154)

[2.2 Обоснование выбора компонентов и технологий для реализации программного средства 18](#_Toc134462155)

[2.3 Архитектурные решения 22](#_Toc134462156)

[2.4 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства 25](#_Toc134462157)

[2.5 Проектирование пользовательского интерфейса 27](#_Toc134462158)

[3 Тестирование и проверка работоспособности программного средства 28](#_Toc134462159)

[4 Руководство по развертыванию и использованию программного средства 33](#_Toc134462160)

[4.1 Руководство по установке (развертыванию) программного средства 33](#_Toc134462161)

[4.2 Руководство пользователя 33](#_Toc134462162)

[Заключение 42](#_Toc134462163)

[Список использованных источников 43](#_Toc134462164)

[Приложение А (обязательное) Отчет о проверке на заимствование в системе «Антиплагиат» 45](#_Toc134462165)

[Приложение Б (обязательное) Листинг кода алгоритмов, реализующих бизнес-логику 46](#_Toc134462166)

[Приложение В (обязательное) Листинг скрипта генерации базы данных 50](#_Toc134462167)

**ВВЕДЕНИЕ**

Веб-приложение, предназначенное для управления деятельностью компании, является важным и удобным инструментом для эффективной организации работы фирмы. С развитием информационных технологий возможно создание системы управления предприятием в виде сайта, что значительно упрощает работу компании.

В настоящее время, когда на рынке услуг и конкуренция очень высока, создание веб-приложения становится обязательным условием для поддержания высоких позиций и удержания клиентов. Оценка груминг салонов клиентами зависит не только от качества предоставляемых услуг, но и от уровня сервиса. Исследования показывают, что четыре из пяти потенциальных клиентов предпочтут конкурентов, если обслуживание будет недостаточно качественным.

Сайт груминг салона является одним из основных каналов продажи услуг. В интернете потенциальные клиенты ищут, оценивают салоны, получают необходимую информацию и принимают решение о выборе салона. Поэтому создание удобного для клиента, автоматизированного сайта является крайне важным.

Запись на услуги является ключевым этапом перед покупкой, и обычно это задача администрации груминг салона, которая также обязана встречать клиентов, составлять расписание работы сотрудников, консультировать потенциальных клиентов и составлять отчетность. Однако все эти задачи могут быть ускорены и упрощены с помощью автоматизации процессов с помощью веб-приложения. Инструменты, такие как онлайн-запись в груминг салон, отправка заявки на запись с последующим звонком от администрации салона и онлайн-консультации, становятся все более популярными. Компании, которые не используют эти возможности, рискуют потерять огромное количество потенциальных клиентов, а значит, упускают прибыль и проигрывают в конкурентной борьбе тем, которые в полной мере используют потенциал интернет-технологий.

Разработка веб-приложения для груминг салона имеет целый ряд преимуществ, таких как: привлечение целевой аудитории и увеличение узнаваемости салона, возможность онлайн-записи, напоминание клиентам о предстоящих посещениях услуг, предоставление актуальной информации в режиме онлайн, сбор подробной статистики о потребностях и востребованности услуг, а также о популярности грумеров и т.д. Кроме того, с помощью разработанного веб-приложения можно создать удобное онлайн-расписание работы для грумеров студии.

Основной инструмент веб-приложения для груминг салона заключается в системе онлайн-записи на услуги, которая позволяет клиентам самостоятельно выбрать удобное время приёма и записаться на услуги через интерактивный календарь работы грумеров. В результате использования такой системы улучшаются рабочие процессы администрации и автоматизируются задачи по учёту и записи клиентов.

Цель курсового проекта – разработка веб-приложения для автоматизации работы груминг салона. Разработанное приложение позволит повысить эффективность деятельности салона.

С целью достижения цели курсового проекта необходимо выполнить ряд перечисленных ниже задач:

* проанализировать предметную область работы груминг салона в сфере оказания услуг и взаимодействия с клиентами;
* выбрать подходящую технологию для создания веб-приложения с архитектурой «клиент-сервер», которая будет располагать возможностями для реализации поставленной цели;
* разработать концептуальную модель будущего проекта;
* разработать веб-приложение, позволяющее автоматизировать работу груминг салона;
* отладить и протестировать созданный программный продукт на соответствие всем заявленным требованиям.

Данное веб-приложение будет способствовать достижению целей, связанных с улучшением качества обслуживания клиентов, более эффективным информированием целевой аудитории о предоставляемых услугах, сокращением времени на запись на услуги, что в результате приведёт к повышению конкурентоспособности груминг салона и увеличению числа клиентов и дохода.

# **АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА**

## **Описание предметной области**

Необходимость создания веб-приложения обуславливается необходимостью автоматизации существующих информационных процессов. Перед созданием веб-приложения необходимо поставить перед собой ряд вопросов: для достижения каких целей необходима разработка, насколько быстро нужно разработать систему, а также, какова величина затрат для того, чтобы осуществить его проектирование.

В интернете существует немало коммерческих подходов. В нем можно рекламировать стандартные услуги и продавать товары, он открывает реальные перспективы электронной коммерции. На современном этапе развития можно выделить два основных направления использования интернета в бизнесе: технологии интернета для бизнеса и бизнес в интернет-пространстве.

Салон красоты для животных предоставляет услуги, которые позволяют удовлетворить спрос людей, на уходовые процедуры для их питомцев.

Целевая аудитория салона для животных – люди, которые ухаживают за своими домашними животными и хотят обеспечить им качественный уход и удобства.

Чаще всего целевая аудитория включает людей, которые ищут профессиональные услуги для своих животных, такие как стрижка, мытье, уход за зубами и когтями, ветеринарные услуги и т.д.

Веб-приложение позволяет автоматизировать и структурировать необходимые для целевой аудитории услуги, обеспечить возможность быстрой записи на необходимую услугу, предоставить все возможный спектр услуг с их кратким описанием, также предоставить информацию о стоимости услуги и информацию о грумерах с отзывами клиентов. Помимо прочего веб-приложение выполняет информационно-ознакомительную функцию: приложение проводит ознакомление потенциальных клиентов с предоставляемыми услугами, контактными данными сотрудников. Веб-приложение представляет посетителям сайта основную информацию, позволяет хранить и удалять данные о грумерах и клиентах.

Можно выделить следующие услуги, предоставляемые веб-приложением:

* стрижка, данный комплекс включает в себя набор из нескольких услуг, таких как вычесывание колтунов, мойка, сушка, стрижка, этот комплекс рекомендуется посещать не реже чем раз в два месяца;
* мойка и сушка, данный комплекс отличается от предыдущего отсутствием стрижки, этот комплекс является более повседневным и рекомендуется к посещению раз в месяц;
* подстригание когтей, данная процедура подразумевает очистку лап от грязи и стрижку когтей, рекомендуется к посещению раз в месяц или чаще, если это необходимо;
* чистка зубов, данная процедура подразумевает гигиеническую чистку зубов, а также удаление зубного камня и налета с помощью ультразвука, если необходимо.

Организационная структура распределения рабочих процессов компании представлена на рисунке 1.1:

Рисунок 1.1 – Организационная структура салона красоты для животных

Схема взаимодействия клиентов и мастеров салона посредством связи через администрацию компании представлена на рисунке 1.2:

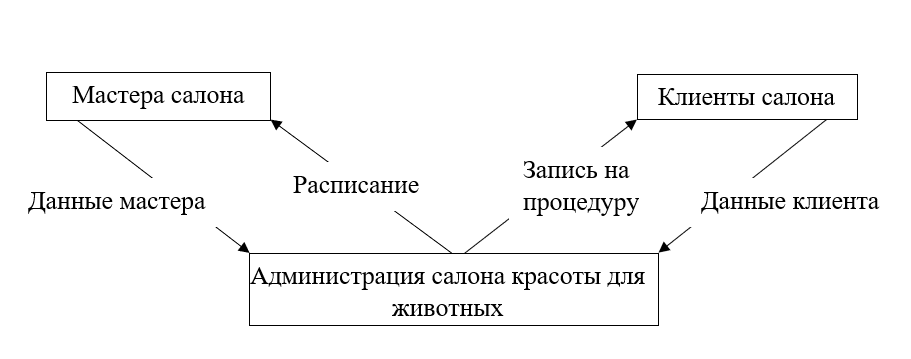


Рисунок 1.2 – Схема взаимодействий в салоне красоты для животных

## **Разработка функциональной модели предметной области**

Разработка функциональной модели предметной области – процесс создания описания того, как система или процесс должны функционировать в рамках определенной предметной области. Функциональная модель обычно состоит из блоков, которые описывают различные функции, процессы и задачи, выполняемые в предметной области.

В результате создания функциональной модели предметной области, получается диаграмма, которая показывает, как должна работать система или процесс в рамках данной предметной области. Это может быть полезно при проектировании системы или процесса, а также при оценке возможных изменений в них [1].

На контекстной диаграмме верхнего уровня представлена функциональная модель «Деятельность салона» (рисунок 1.3).

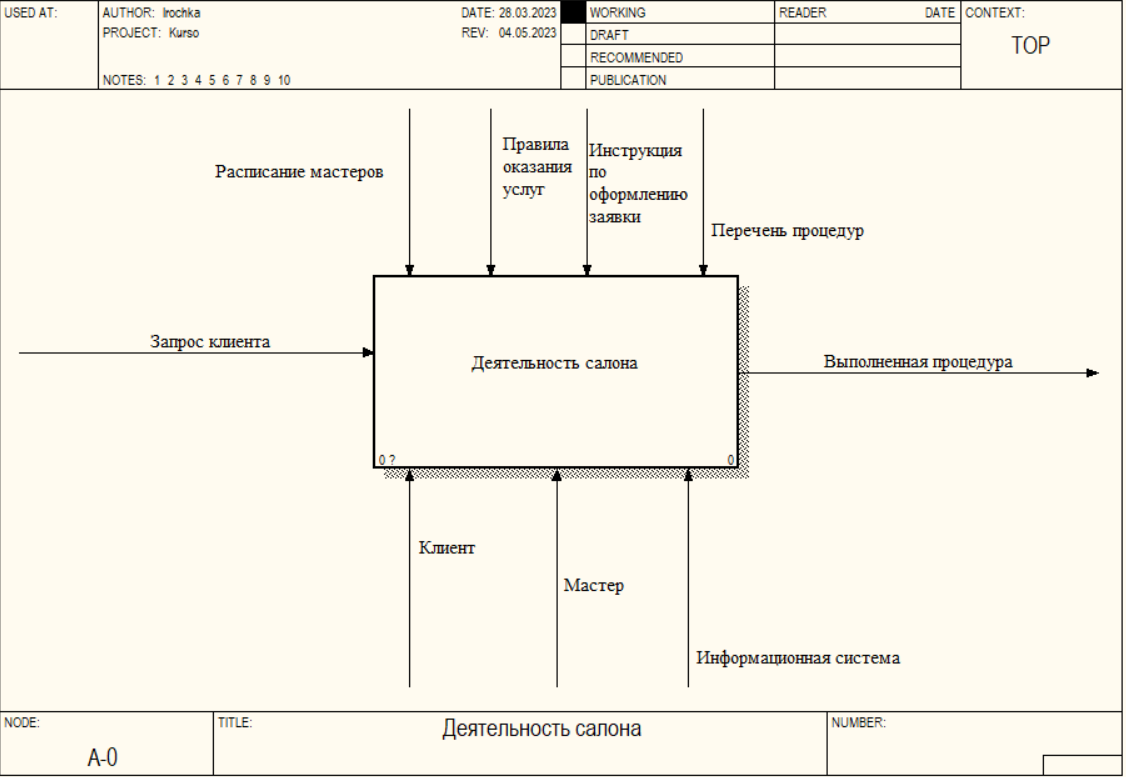


Рисунок 1.3 – Контекстная диаграмма «Деятельность салона»

Были определены входные и выходные данные, механизмы и управление.

Входными данными является запрос клиента; механизмами являются информационная система, клиент и мастер; управлением являются расписание мастеров, перечень процедур. Главной целью бизнес-процесса является выполненная процедура.

Для контекстной диаграммы можно выделить следующие подпроцессы (рисунок 1.4):

* создать заказ;
* принять заказ;
* выполнить услугу;
* подтвердить выполнение услуги.

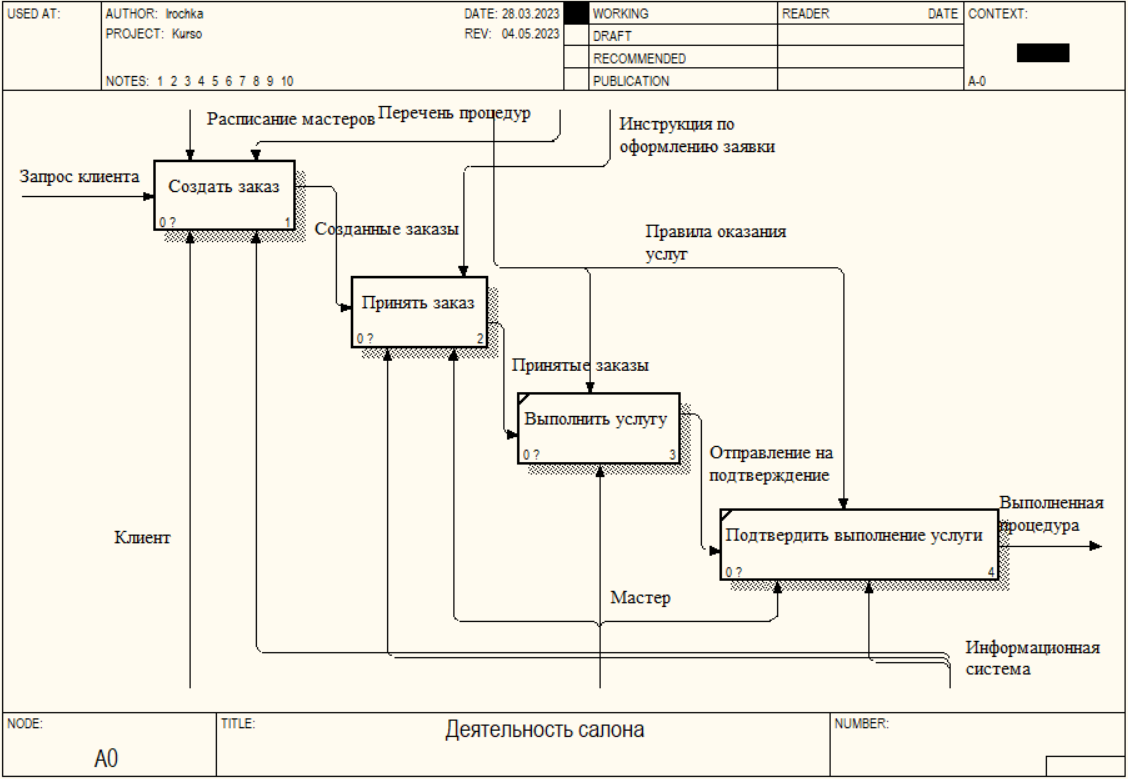


Рисунок 1.4 – Декомпозиция контекстной диаграммы «Деятельность салона»

Процесс начинается с создания заказа клиентом. Здесь мастер узнает, что необходимо клиенту согласно перечню предлагаемых услуг, и принимает заказ. Далее мастеру необходимо выполнить услугу и подтвердить ее выполнение, на этом процесс заканчивается.

Декомпозиция блока создания заказа представлена на рисунке 1.5.

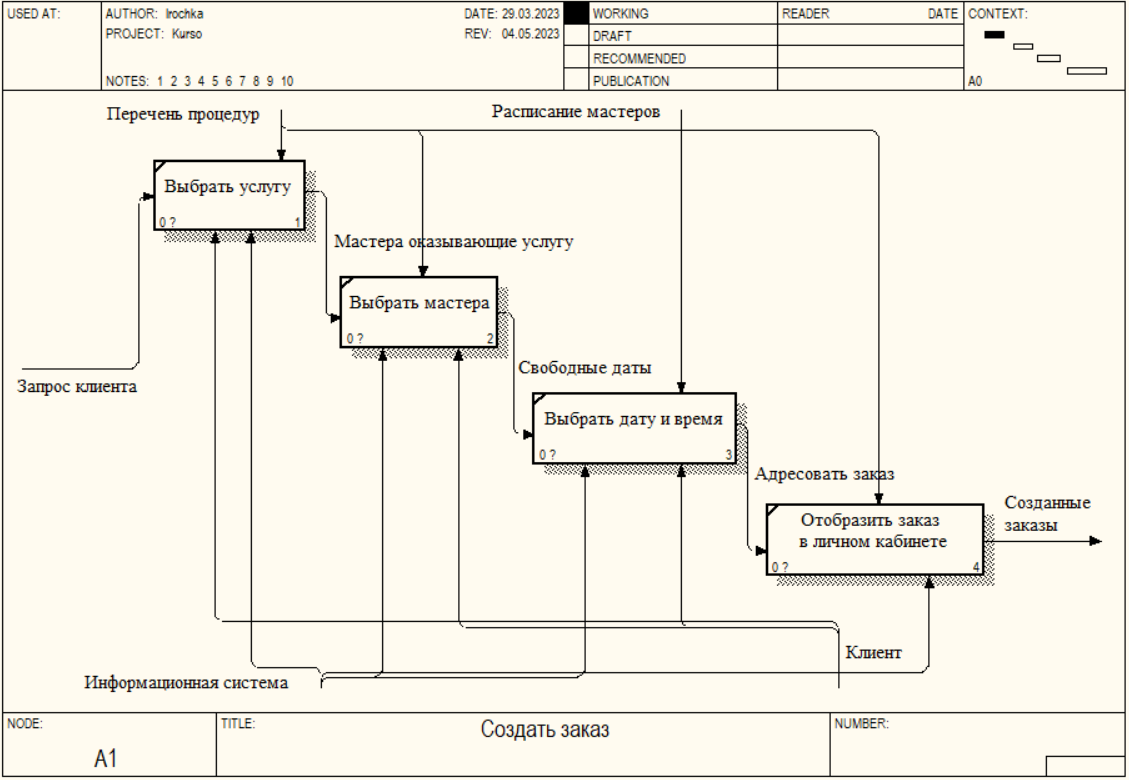


Рисунок 1.5 – Декомпозиция блока «Создать заказ»

Данная декомпозиция состоит из следующих этапов: «Выбрать услугу», «Выбрать мастера», «Выбрать дату и время», «Отобразить заказ в личном кабинете».

Анализируя запросы клиентов, составляется база запросов, которые необходимо будет принять мастерам для удовлетворения своих клиентов.

Вторым блоком декомпозиции главного процесса является блок «Принять заказ» (рисунок 1.6).

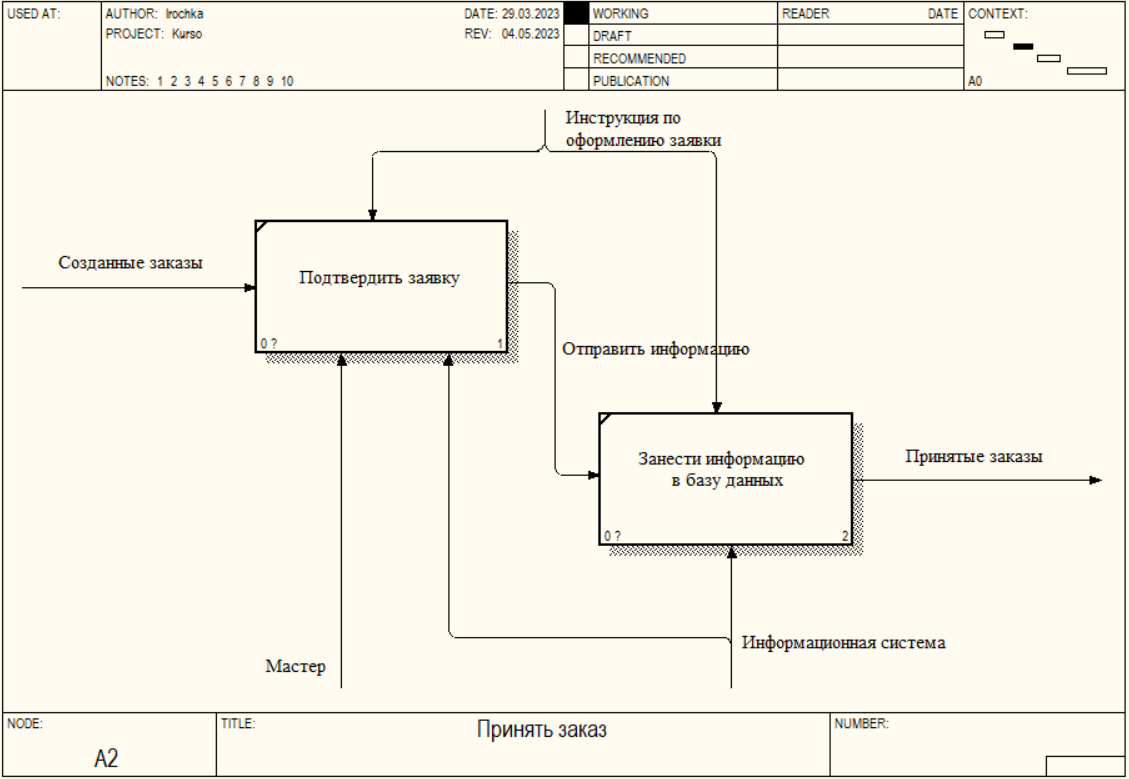


Рисунок 1.6 – Декомпозиция блока «Принять заказ»

Данная декомпозиция состоит из следующих блоков: «Подтвердить заявку», «Занести информацию в базу данных».

В результате мастер выполняет процедуру и отмечает ее как выполненную, что является подтверждением того, что процесс работы с клиентом является завершенным.

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

С целью анализа требований к разрабатываемому программному средству будет представлена диаграмма вариантов использования и ее описание.

Диаграмма вариантов использования (Use Case Diagram) – графическая модель, которая показывает отношения между актерами (пользователями) и функциональностью системы в виде взаимодействий. Она часто используется в процессе проектирования программного обеспечения и позволяет лучше понимать, как система будет использоваться пользователем.

Диаграмма вариантов использования состоит из актеров, вариантов использования и связей между ними. Актеры – внешние сущности, которые взаимодействуют с системой. Варианты использования – конкретные действия или задачи, которые пользователь может выполнить с помощью системы [2].

Связи между актерами и вариантами отображаются с использованием отношений четырех видов:

* ассоциаций – служит для обозначения взаимодействия актера с вариантом использования;
* обобщения – служит для указания того факта, что некоторая сущность А может быть обобщена до сущности В;
* включения (зависимость со стереотипом «include») – указывает, что некоторое заданное поведение одного варианта использования обязательно включается в качестве составного компонента в последовательность поведения другого варианта использования;
* расширения (зависимость со стереотипом «extend») – определяет потенциальную возможность включения поведения одного варианта использования в состав другого [3].

В курсовой работе есть 3 актера: мастер (грумер), клиент и администратор (рисунок 1.7)

Клиент сайта салона может:

* просматривать услуги, выполнять поиск необходимой;
* осуществлять фильтрацию услуг, для удобства поиска необходимой;
* оставлять заявку на услугу с возможностью выбрать мастера (грумера), дату и время;
* просматривать свои записи;
* просматривать профили мастеров (грумеров).

Мастер салона может:

* просматривать услуги;
* изменять свой профиль;
* добавлять услуги;
* подтверждать/отклонять записи;
* просматривать свои записи;
* просматривать профили мастеров (грумеров).

Администратор сайта может:

* выдавать доступ сотрудникам к системе;
* просматривать услуги;
* просматривать статистику;
* просматривать профили мастеров (грумеров);
* просматривать пользователей;
* добавлять категории услуг.

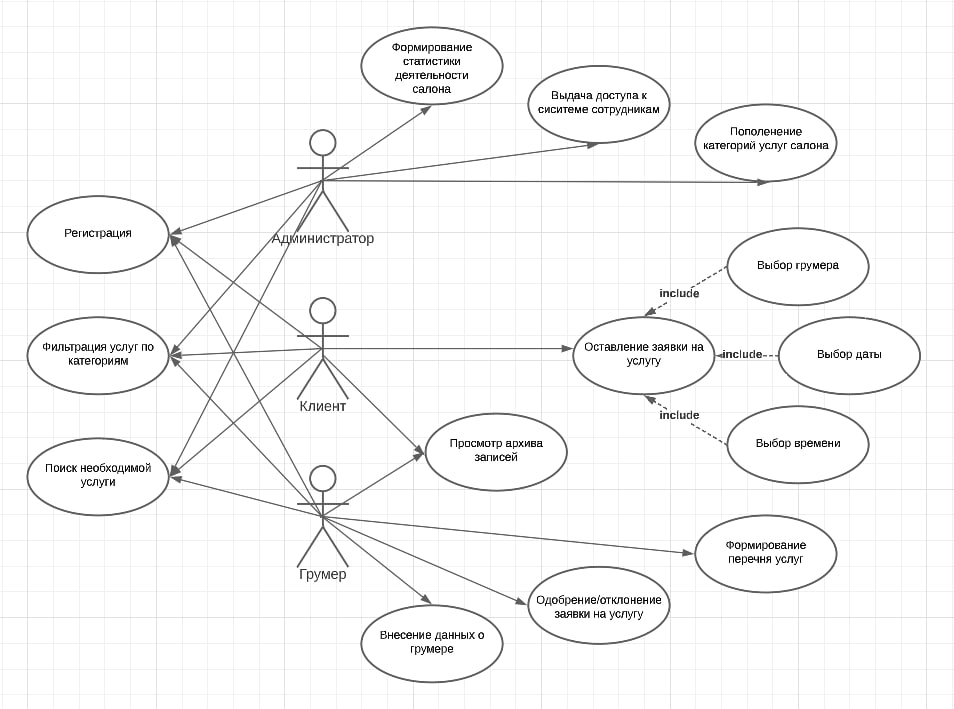


Рисунок 1.7 – Диаграмма вариантов использования

## **Разработка информационной модели предметной области**

Информационная модель предметной области – формализованное описание предметной области, которое отражает основные объекты, свойства и взаимосвязи между ними, а также операции, которые могут выполняться над этими объектами.

Информационная модель может быть представлена в виде графической схемы, таблицы или другого формата. В процессе разработки информационной модели проводится анализ предметной области, определяются основные элементы и структуры, выявляются свойства объектов и взаимосвязи между ними.

Основной целью создания информационной модели является обеспечение полного и точного понимания предметной области и создание основы для создания базы данных. Информационная модель позволяет описать структуру данных, которые будут храниться в базе данных, а также определить, как эти данные будут использоваться и обрабатываться в рамках различных бизнес-процессов [4].

В процессе разработки системы была сформирована информационная модель системы (см. рисунок 1.8). В ходе моделирования были выделены следующие сущности:

* category;
* notes;
* ordering;
* users;
* masters.

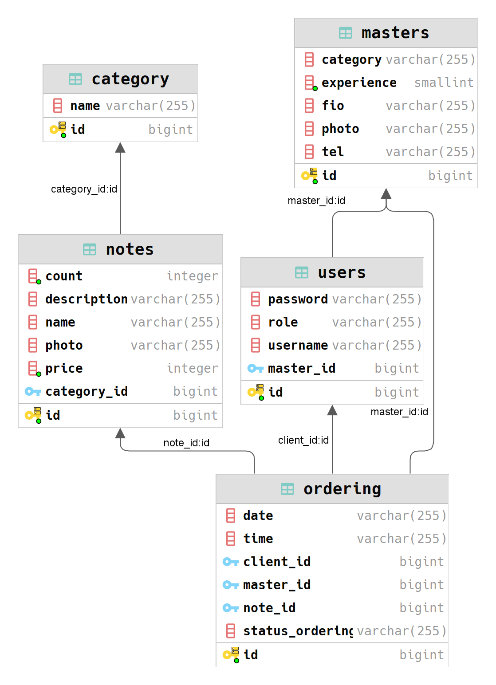


Рисунок 1.8 – Информационная модель системы

Сущность «category» включает в себя следующие атрибуты:

* «name» – название категории (направления);
* «id» – первичный ключ таблицы.

Сущность «notes» включает в себя следующие атрибуты:

* «discription» – описание услуги;
* «name» – название услуги;
* «photo» – имя картинки;
* «price» – стоимость услуги;
* «count» – сумма проданных процедур;
* «category\_id» – внешний ключ связанной таблицы;
* «id» – первичный ключ таблицы.

Сущность «ordering» включает в себя следующие атрибуты:

* «date» – дата записи;
* «time» – время записи;
* «client\_id» – внешний ключ связанной таблицы;
* «master\_id» – внешний ключ связанной таблицы;
* «note\_id» – внешний ключ связанной таблицы;
* «status\_ordering» – статус записи;
* «id» – первичный ключ таблицы.

Сущность «users» включает в себя следующие атрибуты:

* «password» – пароли пользователей;
* «role» – роль пользователя;
* «username» – логины пользователей;
* «master\_id» – внешний ключ связанной таблицы;
* «id» – первичный ключ таблицы.

Сущность «masters» включает в себя следующие атрибуты:

* «category» – категория (направление) мастера;
* «experience» – стаж;
* «fio» – фио мастера;
* «photo» – имя картинки;
* «tel» – телефон мастера;
* «id» – первичный ключ таблицы.

## **UML-модели представления программного средства и их описания**

Диаграмма последовательности (Sequence diagram) – графическое представление взаимодействия объектов в рамках определенного сценария. Это один из самых распространенных типов диаграмм UML (Unified Modeling Language) и используется для моделирования динамических аспектов системы.

Диаграмма последовательности показывает последовательность сообщений, отправляемых между объектами, и временные интервалы, в течение которых эти сообщения передаются. Каждый объект на диаграмме представлен вертикальной линией, называемой "жизненным циклом объекта". Сообщения представляются стрелками, направленными от одного объекта к другому, и сопровождаются названием операции и аргументами [5].

Диаграмма последовательности курсовой работы представлена на рисунке 1.9.

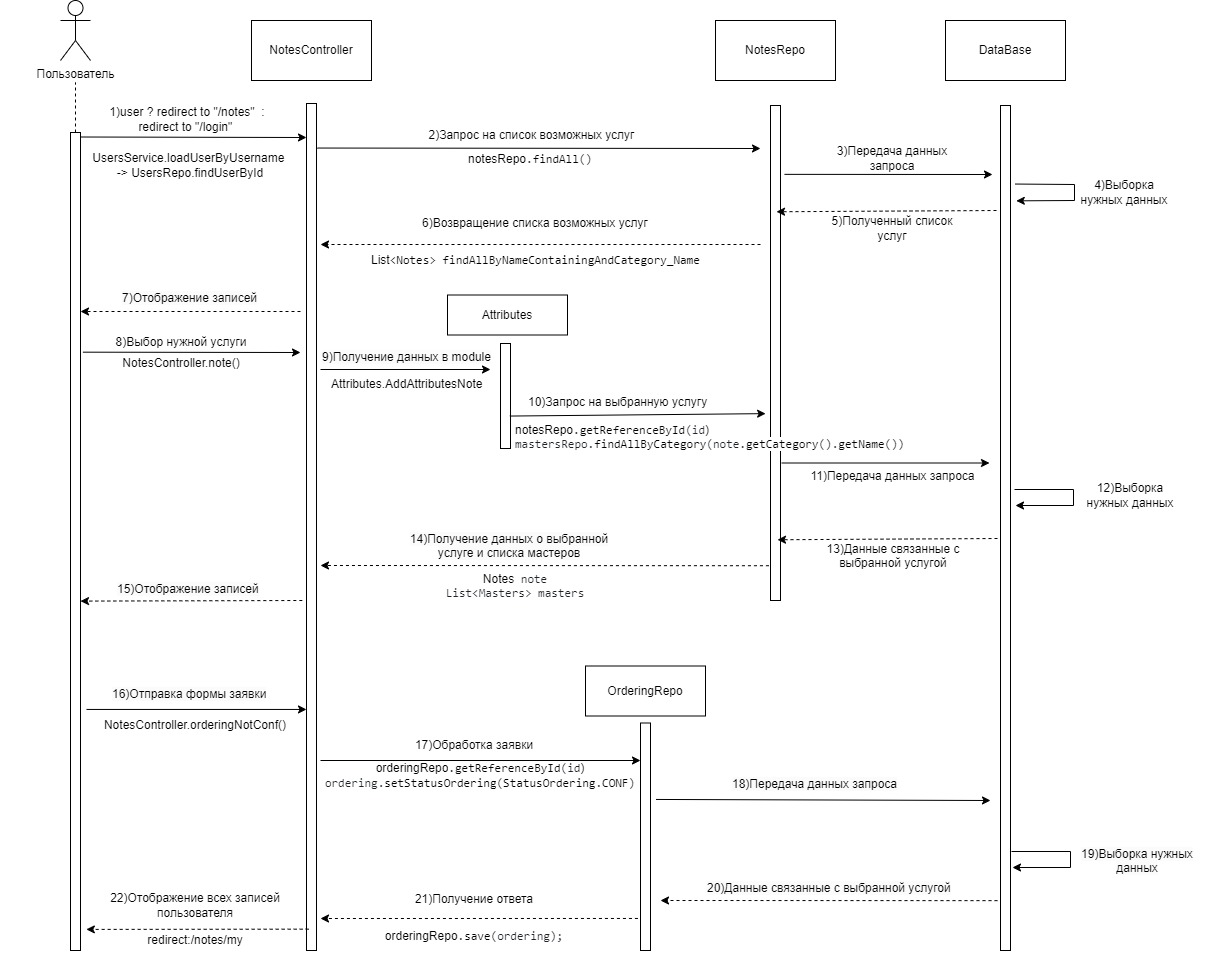


Рисунок 1.9 – Диаграмма последовательности

Диаграмма состояний (state diagram) – это графическая модель, которая используется для описания состояний, переходов между ними и событий, которые приводят к этим переходам. Она широко применяется в различных областях, включая программное обеспечение, автоматизацию производственных процессов и телекоммуникации. Диаграмма состояний представлена на рисунке 1.10. [6].

Начальным состоянием объекта является посетитель сайта салона груминга. После совершения пользователем определённых действий по регистрации в системе объект переходит в состояние «Зарегистрированный пользователь». После осуществления пользователем ввода определённых параметров по записи на приём, объект переходит в новое состояние: «Пользователь, осуществляющий запись на услугу». После формирования запроса на запись на услугу пользователь получает новый статус – «Клиент с запросом на услугу». Далее ему необходимо получить подтверждение его записи от мастера груминг салона и получить статус «Клиент с одобренной записью на услугу», в результате чего получит оказание услуги и перейдет в конечное состояние «Обслуженный клиент груминг салона». Однако мастер также может отклонить запись ввиду занятости в это время, в таком случае клиент переходит в состояние «Клиент с отклонённой записью» и его конечное состояние будет «Необслуженный клиент груминг салона».

Таким образом, клиент может принимать два конечных состояния: «Обслуженный клиент груминг салона» и «Необслуженный клиент груминг салона».

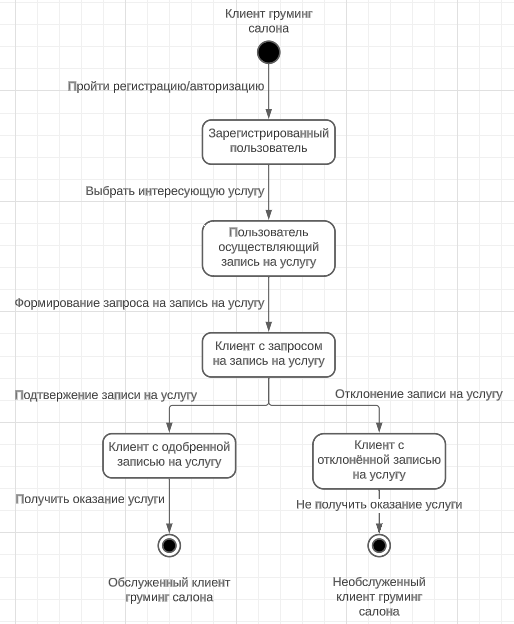


Рисунок 1.10 – Диаграмма состояний клиента в процессе осуществления записи на услугу

# **ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

## **Постановка задачи**

Задачей курсовой работы является разработка веб-приложения для автоматизации рабочего процесса салона красоты для животных, которое позволит клиентам записываться на услуги, а администратору отслеживать аналитику записей.

Первостепенной целью разработки данного программного продукта является упрощение и ускорение рабочего процесса салона, в частности, регистрации клиентов, их записи на услуги, составление графиков работы мастеров.

Программное средства должно обладать следующими пользовательскими функциями:

* просмотр информации об услугах и мастерах на сайте;
* возможностью регистрации и записи на процедуры.

Программное средства должно обладать следующими функциями для мастера:

* просмотр информации об услугах и мастерах на сайте;
* возможностью регистрации и изменения информации о себе;
* возможность добавлять и изменять услуги.

Программное средства должно обладать следующими функциями для администратора:

* просмотр информации об услугах и мастерах на сайте;
* возможностью добавлять категории услуг и изменять их;
* возможностью изменять роль и удалять пользователей;
* возможностью просматривать аналитику записей.

В ходе реализации необходимо разделить роли администратора, клиента и мастера.

Данное приложение должно облегчить работу администратора, а также позволить клиентам записывать своих питомцев на процедуры в любое удобное для них время.

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

PostgreSQL (или просто Postgres) – мощная объектно-реляционная система управления базами данных (СУБД), которая позволяет эффективно хранить, организовывать и манипулировать большими объемами структурированных данных. PostgreSQL является свободной и открытой программой с открытым исходным кодом и поддерживает широкий спектр операционных систем, включая Linux, Unix и Windows.

PostgreSQL предоставляет множество функций и возможностей, таких как поддержка транзакций, проверка целостности данных, гибкий язык запросов SQL, процедурное расширение на языке PL/pgSQL и многие другие. Благодаря своей надежности и масштабируемости PostgreSQL широко используется в качестве базы данных для крупных приложений, включая социальные сети, электронную коммерцию и финансовые системы [7].

Некоторые из основных возможностей Postgres:

* масштабируемость: PostgreSQL может управлять большим объемом данных и поддерживать множество соединений с базой данных одновременно;
* гибкость: PostgreSQL позволяет определять пользовательские типы данных, функции и операторы;
* транзакционность: PostgreSQL поддерживает транзакционность, что позволяет гарантировать целостность данных при выполнении нескольких операций в одной транзакции;
* безопасность: PostgreSQL предоставляет механизмы авторизации и аутентификации, а также возможность управления доступом к базе данных;
* индексация: PostgreSQL поддерживает различные типы индексов, которые могут использоваться для ускорения выполнения запросов к базе данных;
* репликация: PostgreSQL поддерживает репликацию данных, что позволяет создавать резервные копии базы данных и обеспечивать отказоустойчивость системы;
* расширяемость: PostgreSQL поддерживает плагины и расширения, которые позволяют добавлять дополнительную функциональность к базе данных [8].

Java Spring – фреймворк для разработки приложений на языке Java. Он предоставляет набор инструментов и библиотек, которые помогают упростить процесс разработки и улучшить производительность. Spring является одним из самых популярных и широко используемых фреймворков для разработки приложений на Java.

Spring был создан для упрощения разработки приложений на Java и предоставления более эффективного способа интеграции компонентов и библиотек. Он состоит из различных модулей, которые могут быть использованы в сочетании или отдельно для создания приложения.

Одним из наиболее известных модулей Spring является Spring MVC, который предоставляет архитектуру Model-View-Controller для построения веб-приложений. Вместе с Spring Boot, который предоставляет упрощенный способ создания приложений на основе Spring, Spring является одним из наиболее распространенных фреймворков для разработки приложений на Java [9].

MVC (Model-View-Controller) – это архитектурный паттерн, который используется для разработки веб-приложений. Он разделяет приложение на три основных компонента: Model, View и Controller. Каждый из этих компонентов выполняет свою специфическую функцию, которая позволяет разработчикам разделять логику и данные, обрабатывать пользовательский ввод и представлять информацию пользователю.

Model – это компонент, который отвечает за хранение данных и выполнение бизнес-логики. Этот компонент может включать базу данных, модели данных и другие компоненты, которые работают с данными приложения.

View – это компонент, который отвечает за представление данных пользователю. Этот компонент может включать HTML-шаблоны, CSS-стили и JavaScript-скрипты, которые отображают данные пользователю.

Controller – это компонент, который отвечает за управление потоком данных и управление пользовательским вводом. Этот компонент может включать маршрутизаторы, контроллеры и другие компоненты, которые управляют потоком данных между моделью и представлением.

Преимущества использования паттерна MVC включают:

Разделение логики приложения на компоненты, которые могут быть легко изменены и заменены.

Улучшение тестируемости приложения, поскольку каждый компонент может быть протестирован независимо от других компонентов.

Улучшение масштабируемости приложения, поскольку каждый компонент может быть масштабирован независимо от других компонентов.

Однако использование паттерна MVC также может привести к более сложному коду и более длительному процессу разработки, поскольку разработчики должны уделять внимание разделению логики на компоненты и управлению потоком данных между ними [10].

Spring Security – это мощный и гибкий фреймворк для аутентификации и авторизации в приложениях, построенных на Spring Framework. Он предоставляет ряд функций, которые позволяют разработчикам создавать безопасные приложения, защищая их от несанкционированного доступа.

Spring Security включает в себя ряд стандартных функций, таких как аутентификация пользователя, авторизация доступа, управление сессией и управление ролями пользователей. Он также обеспечивает механизмы защиты от атак CSRF (межсайтовая подделка запросов) и XSS (межсайтовые скрипты).

Spring Security обеспечивает разделение обязанностей, чтобы упростить процесс создания безопасных приложений. Он позволяет разработчикам определить правила безопасности в конфигурационном файле, а не в коде приложения. Это упрощает поддержку приложения и делает его более легким для изменения.

Spring Security также поддерживает множество способов аутентификации, включая базовую аутентификацию, аутентификацию на основе формы, аутентификацию на основе SSO (одно учетная запись для всех приложений), аутентификацию на основе токенов и многое другое.

В целом, Spring Security предоставляет мощный и гибкий набор инструментов для обеспечения безопасности приложений на Spring Framework. Он упрощает создание безопасных приложений, уменьшает вероятность ошибок и увеличивает общую безопасность системы [11].

Hibernate Framework – это фреймворк для языка [Java](https://blog.skillfactory.ru/java-komu-i-dlya-chego-nuzhen/), предназначенный для работы с [базами данных](https://blog.skillfactory.ru/glossary/baza-dannyh/). Он реализует [объектно-реляционную модель](https://blog.skillfactory.ru/glossary/oop-obektno-orientirovannoe-programmirovanie/) – технологию, которая «соединяет» программные сущности и соответствующие записи в базе.

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

Какие же преимущества даёт использование Hibernate:

* обеспечивает простой API для записи и получения Java-объектов в/из БД;
* минимизирует доступ к БД, используя стратегии fetching;
* не требует сервера приложения;
* позволяет нам не работать с типами данных языка SQL, а иметь дело с привычными нам типами данных Java;
* заботится о создании связей между Java-классами и таблицами БД с помощью XML-файлов не внося изменения в программный код;
* если нам необходимо изменить БД, то достаточно лишь внести изменения в XML-файлы.

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

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

Рассмотрим отдельно каждый из элементов Hibernate:

* transaction: этот объект представляет собой рабочую единицу работы с БД. В Hibernate транзакции обрабатываются менеджером транзакций;
* sessionFactory: самый важный и самый тяжёлый объект (обычно создаётся в единственном экземпляре, при запуске приложения). Нам необходима как минимум одна SessionFactory для каждой БД, каждый из которых конфигурируется отдельным конфигурационным файлом;
* session: сессия используется для получения физического соединения с БД. Обычно, сессия создаётся при необходимости, а после этого закрывается. Это связано с тем, что эти объекты крайне легковесны. Чтобы понять, что это такое, модно сказать, что создание, чтение, изменение и удаление объектов происходит через объект Session;
* query: этот объект использует HQL или SQL для чтения/записи данных из/в БД. Экземпляр запроса используется для связывания параметров запроса, ограничения количества результатов, которые будут возвращены и для выполнения запроса;
* configuration: этот объект используется для создания объекта SessionFactory и конфигурирует сам Hibernate с помощью конфигурационного XML-файла, который объясняет, как обрабатывать объект Session;
* criteria: используется для создания и выполнения объектно-ориентированных запроса для получения объектов [12].

## **Архитектурные решения**

Диаграмма компонентов позволяет определить состав программных компонентов, а также установить зависимости между ними.

Данная диаграмма обеспечивает согласованный переход от логического к физическому представлению системы в виде программных компонентов. Одни компоненты могут существовать только на этапе компиляции программного кода, другие – на этапе его исполнения. Основными элементами диаграммы являются компоненты, интерфейсы и зависимости между ними. Кроме этого, на ней могут отображаться ключевые классы, входящие в компоненты.

Компонент – это физическая часть системы. Компоненты, представляющие собой файлы с исходным кодом классов, библиотеки, исполняемые модули и т.п., которые должны обладать согласованным набором интерфейсов.

Внутри компонента, как и класса, могут быть выделены дополнительные секции, в которых указываются предоставляемые или необходимые для работы интерфейсы и классы, методы, наименование файла-компонента и т.п.

Отношение ассоциации отображается между компонентами и их интерфейсами. Отношение зависимости означает зависимость реализации одних компонентов от реализации других.

В курсовой работе между собой связаны 3 компонента: страница сайта в браузере для клиента, сервер и БД. Связь компонентов представлена на рисунке 2.1.

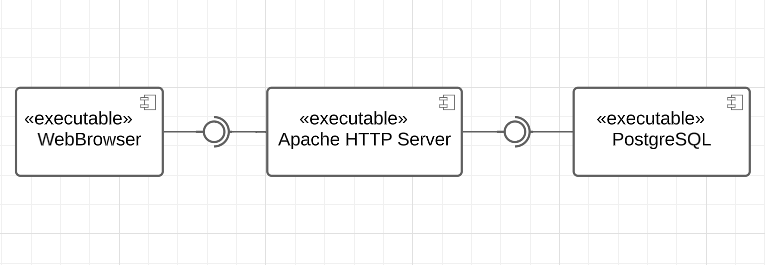


Рисунок 2.1 – Диаграмма компонентов

Второй формой физического представления программной системы является диаграмма развертывания. Она применяется для представления общей конфигурации и топологии распределенной информационной системы, содержит сведения о распределении компонентов по отдельным узлам системы и каналом связи между аппаратными средствами.

Таким образом, диаграмма развертывания предназначена для визуализации элементов и компонентов системы, существующих лишь на этапе ее исполнения, к которым относятся исполнимые файлы, динамические библиотеки, таблицы БД и т. д. Те компоненты, которые не используются на этапе исполнения, на диаграмме не показываются.

Основные цели, преследуемые при разработке диаграммы развертывания:

* распределение компонентов системы по ее физическим узлам;
* отображение физических связей между узлами системы на этапе исполнения;
* выявление узких мест системы и реконфигурация ее топологии для достижения требуемой производительности.

Элементами диаграммы развертывания являются узлы, компоненты и связи между ними [13].

Диаграмма развертывания программного средства представлена на рисунке 2.2. В состав диаграммы входят веб-сервер, сервер базы данных и компьютер, на котором пользователь просматривает сайт.

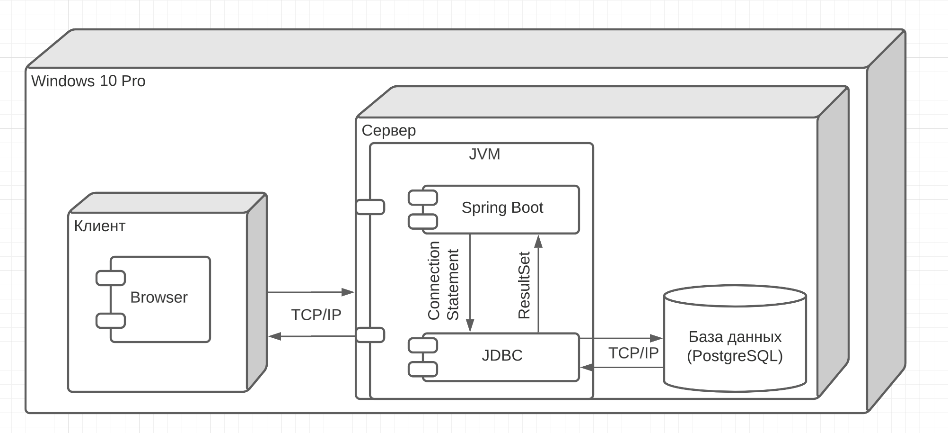


Рисунок 2.2 – Диаграмма развертывания

При составлении диаграммы классов необходимо учитывать особенности задания атрибутов, методов и отношений между классами с учетом специфики используемого языка программирования, в данном случае Java.

Графически класс отображается в виде прямоугольника, который может быть разделен горизонтальными линиями на секции. В этих секциях указывается имя, атрибуты и методы.

С точки зрения структурного подхода, атрибуты – это переменные, а методы – это функции, описанные в теле класса. Они могут быть доступны или не доступны для изменения или выполнения внешними объектами.

Обязательным элементом обозначения класса на диаграмме является его имя. Оно должно быть уникальным в пределах пакета. Если класс является абстрактным, то его имя пишется курсивом [14].

Отношения, которые можно устанавливать между классами:

* отношение ассоциации: означает наличие атрибута, в котором будет храниться ссылка на объект класса, в сторону которого направлена стрелка ассоциации;
* отношения агрегации и композиции являются частными случаями ассоциации;
* отношение обобщения в тексте программы на языке Java показывается ключевым словом «extends» в дочернем классе;
* отношение зависимости не приводит к автоматической генерации кода программы, но свидетельствует об обращении из объекта зависимого класса к атрибутам, методам или непосредственно к объектам независимого класса;
* отношение реализации - отображается только между классами и интерфейсами. В тексте на языке Java данное отношение обозначается ключевым словом «implements» [15].

Диаграмма классов представлена на рисунке 2.3.



Рисунок 2.3 – Диаграмма классов

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

Бизнес-логика программного средства (ПС) представляет собой набор правил и процессов, которые определяют, как ПС должно функционировать, чтобы соответствовать потребностям бизнеса и достигать поставленных целей. Бизнес-логика является основным элементом любого ПС и включает в себя описание бизнес-процессов, правил и требований, которые необходимы для решения задач бизнеса.

Бизнес-логика может быть представлена в виде диаграмм, документации, кода или других форматов. В современном ПС, бизнес-логика часто разделяется на модули, чтобы обеспечить более эффективное управление и обслуживание.

Основные функции бизнес-логики ПС включают:

* управление бизнес-процессами – определяет последовательность действий и операций, необходимых для выполнения определенной задачи.
* управление данными – определяет, как данные будут храниться, обрабатываться и анализироваться в ПС.
* управление правами доступа – определяет, кто имеет доступ к каким данным и функциям ПС.
* управление логикой приложения – определяет, как ПС будет взаимодействовать с пользователем и какие действия будут выполняться в ответ на конкретные действия пользователя [16].

Главная задача приложения салона красоты для животных позволить клиенту быстро и самостоятельно записываться на услуги. Вначале клиент регистрируется на сайте для записи на услугу, затем выбирает услугу, на которую ему необходимо записаться, далее выбирает мастера, дату и время, которые ему необходимы. Мастер же в свою очередь после создания записи клиентом видит в личном кабинете новую запись и принимает решение одобрить либо же отклонить запись. После одобрения записи и выполнения процедуры мастер отмечает услугу как выполненную, тогда она появляется в аналитике, которая доступна к просмотру администратору. Также администратор, контролирует учетные записи клиента и мастера, может назначить им роли и удалять.

Для того чтобы наглядно отобразить алгоритм добавления данных о пользователе в базу данных, рассмотрим блок-схему соответствующей функции (рисунок 2.4).

Для успешной авторизации пользователю нужно ввести правильные логин и пароль для входа в систему. Если логин и пароль, введённые пользователем, будут совпадать с таковыми в базе данных, то ему будет разрешен вход в систему. В случае несовпадения или наличия блокировки пользовательского аккаунта будет выдано сообщение об ошибке.

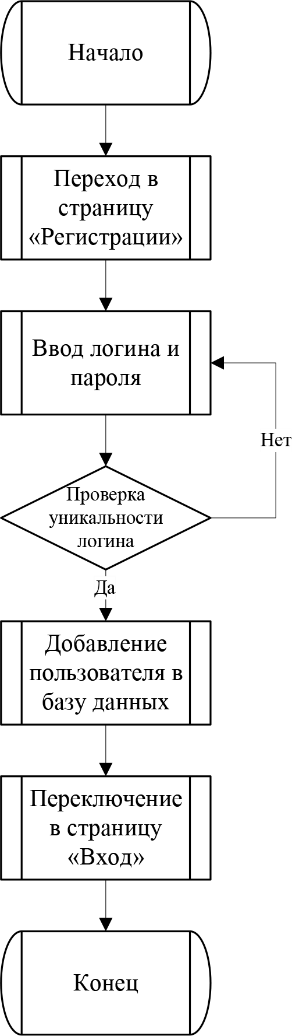


Рисунок 2.4 – Блок-схема добавления информации о пользователе

На рисунке 2.5 представлен алгоритм авторизации и аутентификации в виде блок-схемы.

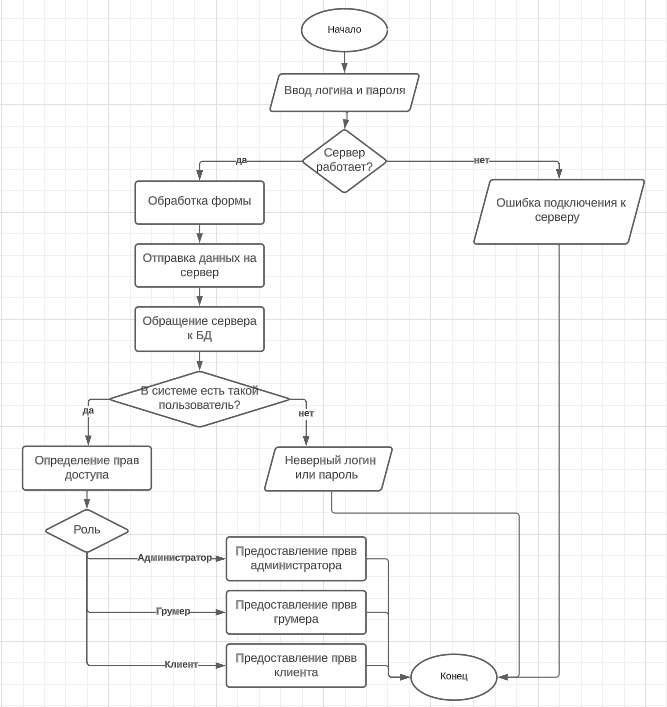


Рисунок 2.5 – Алгоритм авторизации пользователя

## **2.5 Проектирование пользовательского интерфейса**

Проектирование пользовательского интерфейса является важной частью разработки программного обеспечения, сайтов и мобильных приложений. Цель проектирования пользовательского интерфейса заключается в разработке ясного и понятного представления о том, каким должен быть интерфейс системы. Этот инструмент помогает достижению бизнес-целей и созданию конкурентных преимуществ, а также предоставляет разработчикам ясные указания по созданию системы.

В процессе проектирования пользовательского интерфейса необходимо учитывать следующие факторы:

* целевая аудитория: дизайн должен быть нацелен на определенную аудиторию пользователей. Например, дизайн интерфейса для старших людей будет отличаться от дизайна интерфейса для молодых;
* цели и задачи пользователя: дизайн должен помочь пользователям быстро и легко достигать своих целей. Например, если сайт продает товары, то интерфейс должен быть удобным для поиска и покупки товаров;
* информационная архитектура: важно определить структуру и организацию информации на странице или экране, чтобы пользователи могли быстро и легко найти нужную им информацию;
* навигация: дизайн должен обеспечивать удобную и понятную навигацию для пользователя, чтобы он мог легко переходить между различными разделами и функциями;
* визуальный дизайн: дизайн должен быть привлекательным и легко воспринимаемым пользователем. Это включает выбор цветовой палитры, типографики, иконок и других элементов дизайна;
* размер и расположение элементов: дизайн должен учитывать размер и расположение элементов интерфейса, чтобы пользователь мог легко их обнаружить и использовать;
* контекст использования: дизайн должен учитывать контекст использования приложения или сайта, чтобы пользователи могли использовать его в различных ситуациях и на различных устройствах.

В целом, проектирование пользовательского интерфейса – это искусство сбалансировать между несколькими факторами и создать удобный, привлекательный и эффективный интерфейс для пользователей [17].

# **ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА**

Тестирование сайта является неотъемлемым и заключительным этапом в технической разработке сайта, который имеет важнейшее значение в процессе создания ресурса. Качество тестирования напрямую влияет на будущую работу сайта. Однако, к сожалению, разработчики не всегда уделяют достаточное внимание этому этапу, полагаясь на свой опыт. Это может привести к огромным затратам времени и денег на исправление существующих ошибок. Наличие ошибок на сайте негативно влияет на посетителей и может привести к потере аудитории. Кроме того, владелец сайта вынужден платить за доработку или даже повторную разработку ресурса.

Для использования программного средства первоначально необходимо зарегистрироваться. В случае если происходит попытка регистрации пользователя с уже имеющимся в базе логином, будет выведено предупреждение (см. рис. 3.1), а также если поле «Логин» или «Пароль» оставлено пустым (см. рис. 3.2).

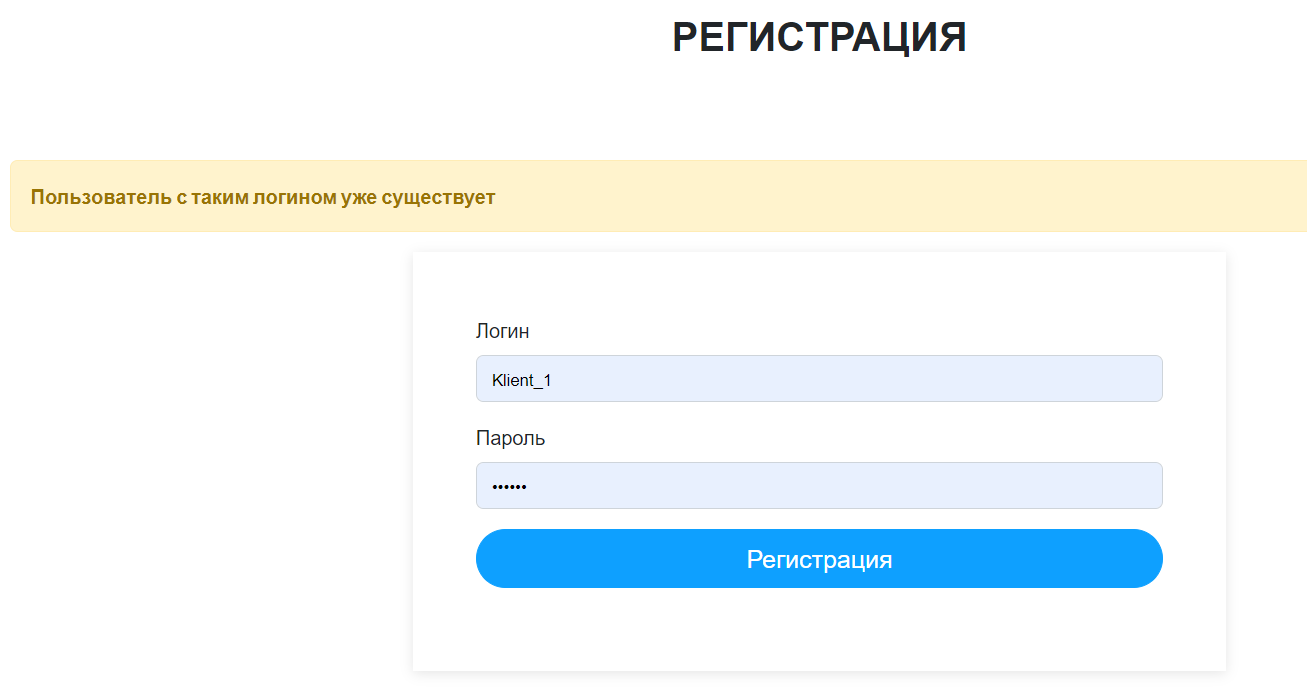


Рисунок 3.1 – Тестирование регистрации

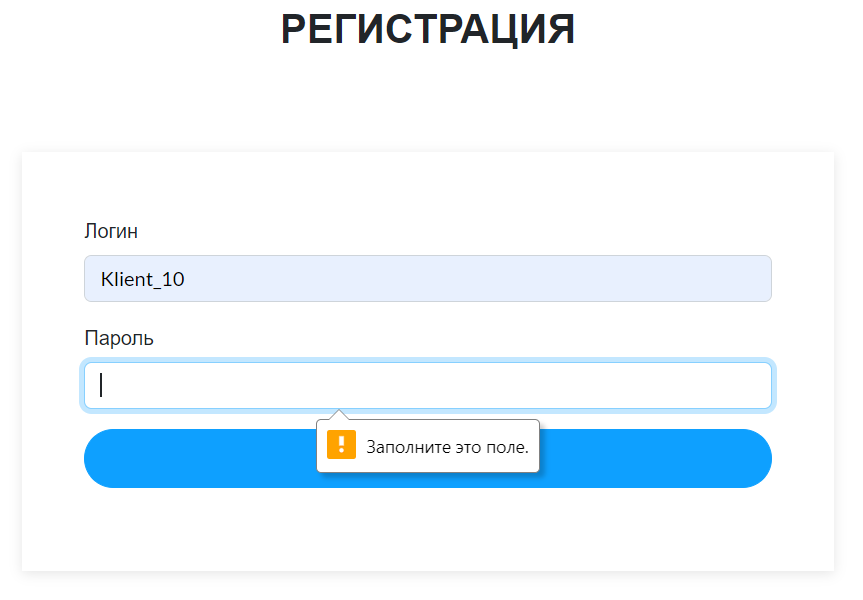


Рисунок 3.2 – Тестирование регистрации

Для входа в систему, необходимо ввести логин и пароль уже зарегистрированного пользователя. Если данные введены неверно или поля оставлены пустыми, выводится соответствующее предупреждение (рисунок 3.3-3.4).

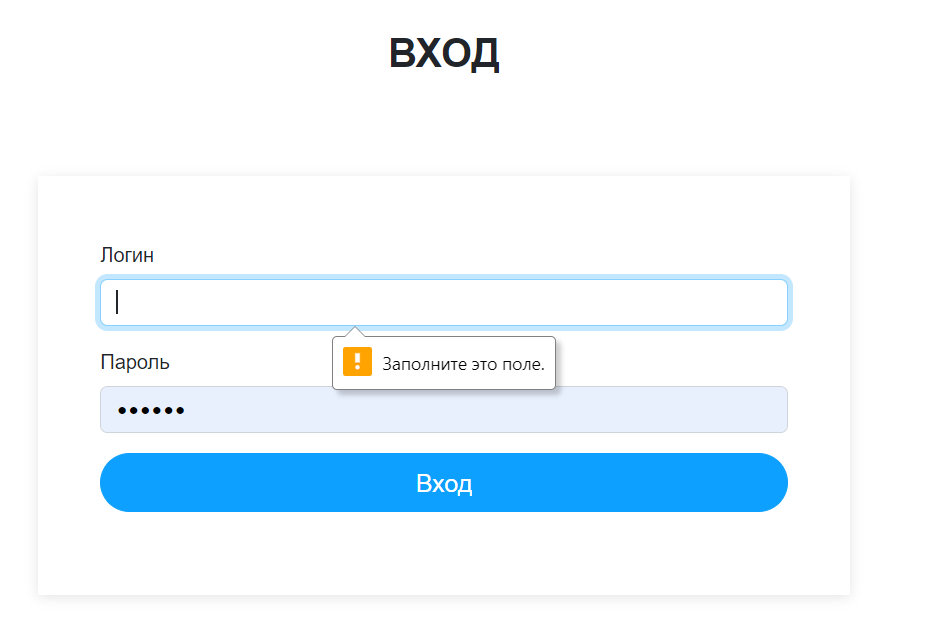


Рисунок 3.3 – Тестирование входа

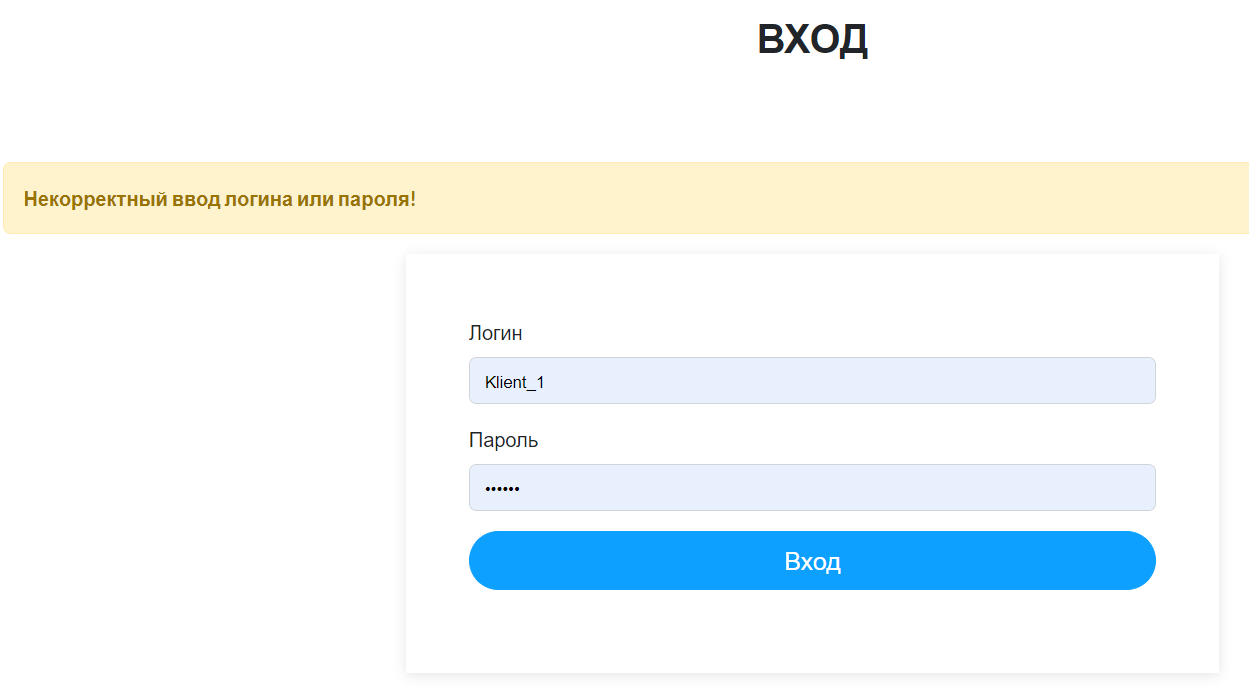


Рисунок 3.4 – Тестирование входа

При записи на услугу необходимо выбрать грумера, дату и время, при оставлении полей пустыми выводится предупреждение (см. рис. 3.5), это же происходит и при оставлении полей пустыми при оформлении профиля грумером (см. рис. 3.6), при создании новых категорий услуг (см. рис. 3.7), при добавлении услуг (см. рис. 3.8-3.9).

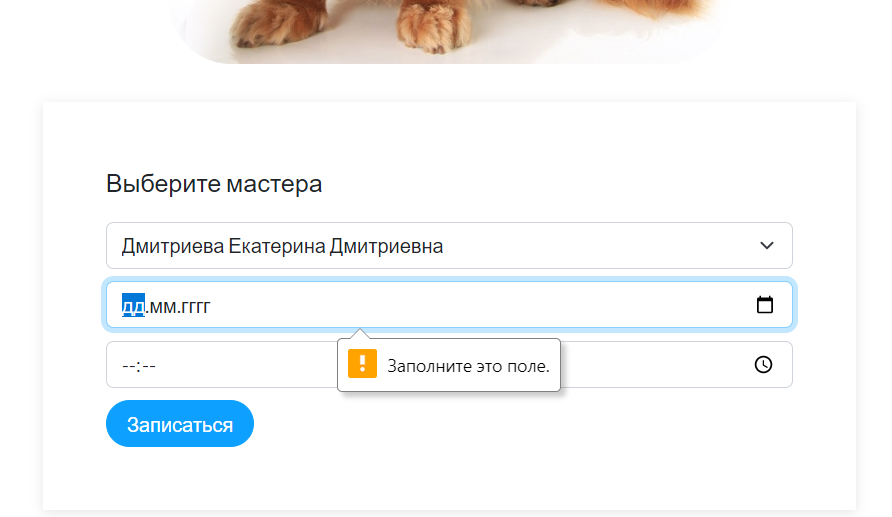


Рисунок 3.5 – Тестирование формы записи

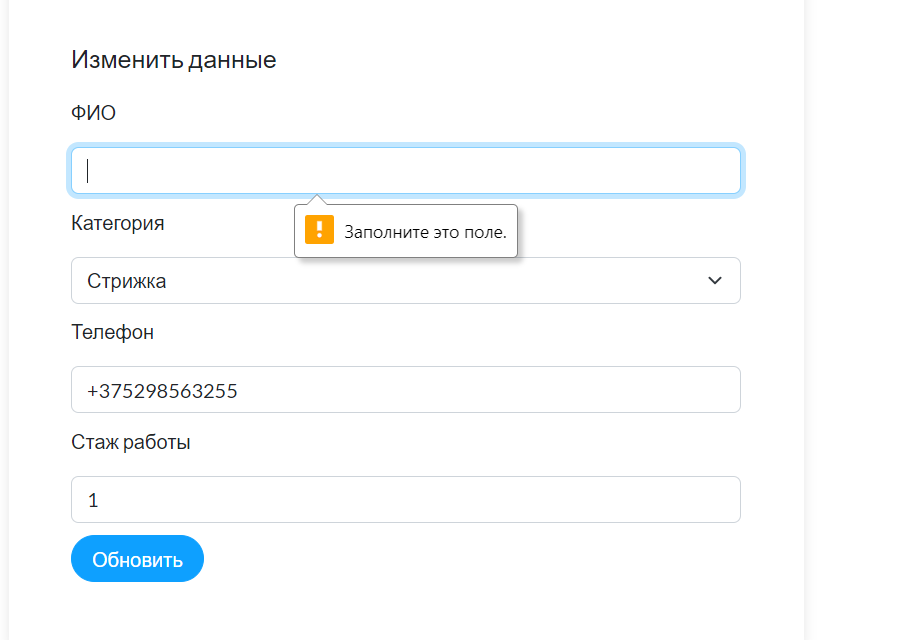


Рисунок 3.6 – Тестирование формы профиля

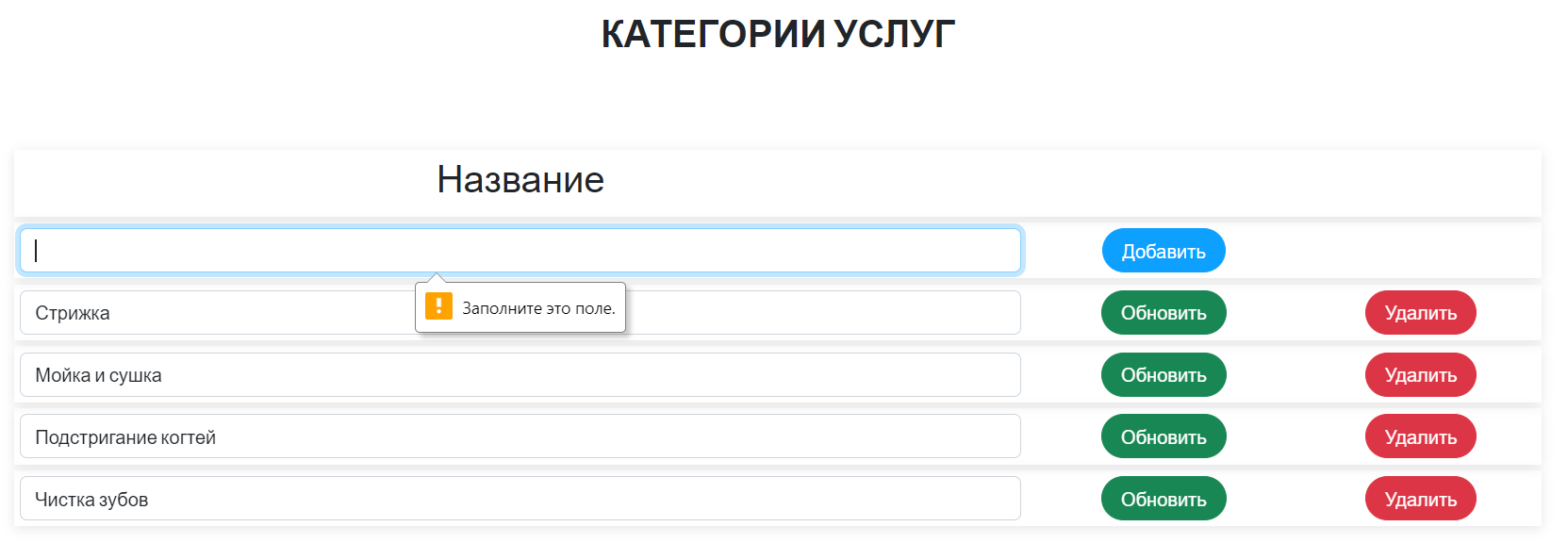


Рисунок 3.7 – Тестирование формы добавления категорий услуг

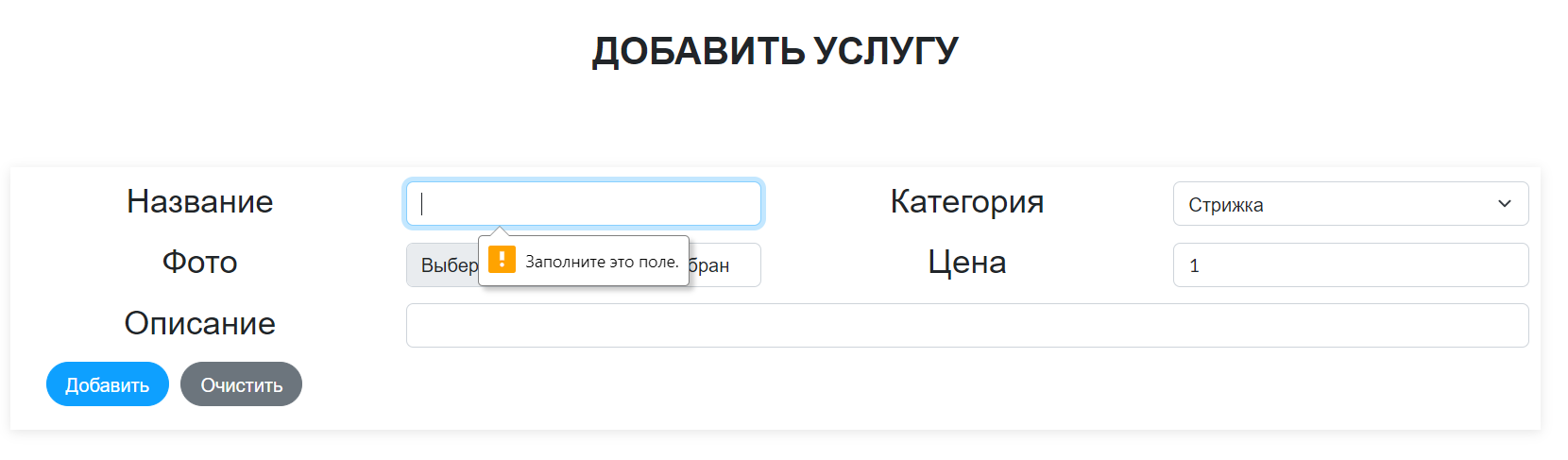


Рисунок 3.8 – Тестирование формы добавления услуг

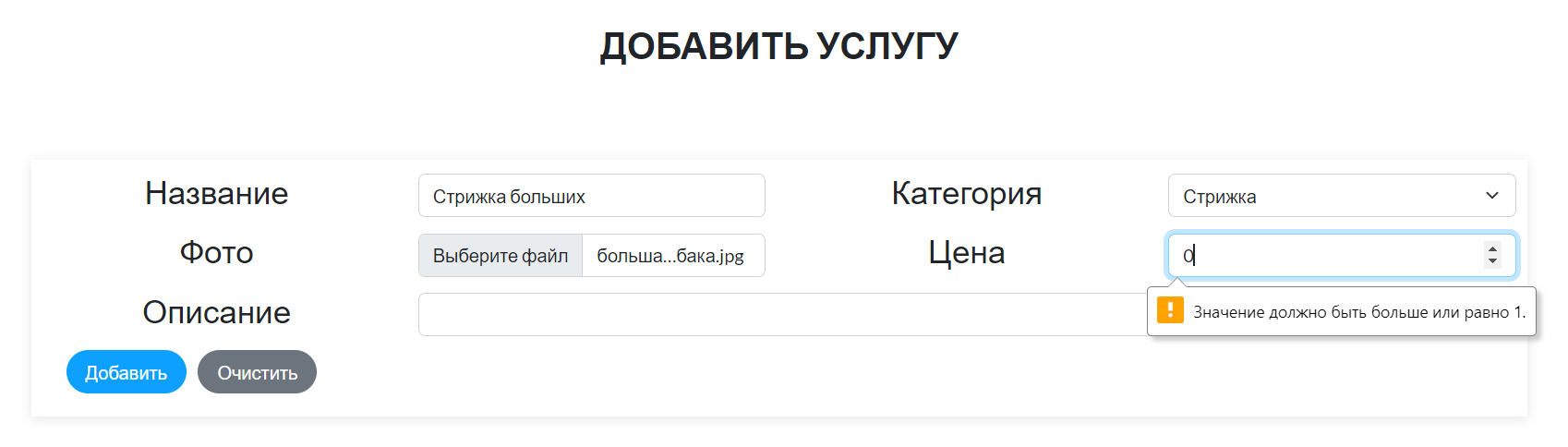


Рисунок 3.9 – Тестирование формы добавления услуг

# **РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА**

## **Руководство по установке (развертыванию) программного средства**

Для развертывания автоматизированной системы работы груминг салона подойдет любая операционная система, которая поддерживает необходимые версии приложений для работы программного средства.

Docker - программное обеспечение, которое автоматизирует процесс развертывания приложений, обеспечивая возможность запуска приложения на любом устройстве, независимо от операционной системы. Для развертывания приложения необходимо перейти в директорию приложения, где находится файл docker-compose.yml, и выполнить команду docker compose up в консоли. Docker создаст контейнер на основе файла docker-compose.yml, установит все необходимые зависимости для работы приложения и создаст базу данных. Затем будут выполнены скрипты миграции базы данных для генерации необходимых таблиц, и приложение будет готово к использованию. После запуска системы пользователь увидит главную страницу сайта.

## **Руководство пользователя**

Для использования программного средства первоначально необходимо зарегистрироваться в системе (см. рис. 4.3). а далее перейти на страницу «Вход» и ввести логин и пароль. При добавлении нового пользователя ему по умолчанию присваивается роль клиента, а администратор в свою очередь на странице «Пользователи» может присвоить необходимую роль для дальнейшего использования.

Важно отметить, что в случае успешной регистрации логин не должен совпадать с уже зарегистрированными пользователями, пользователь будет перенаправлен в страницу входа в случае успешной регистрации.

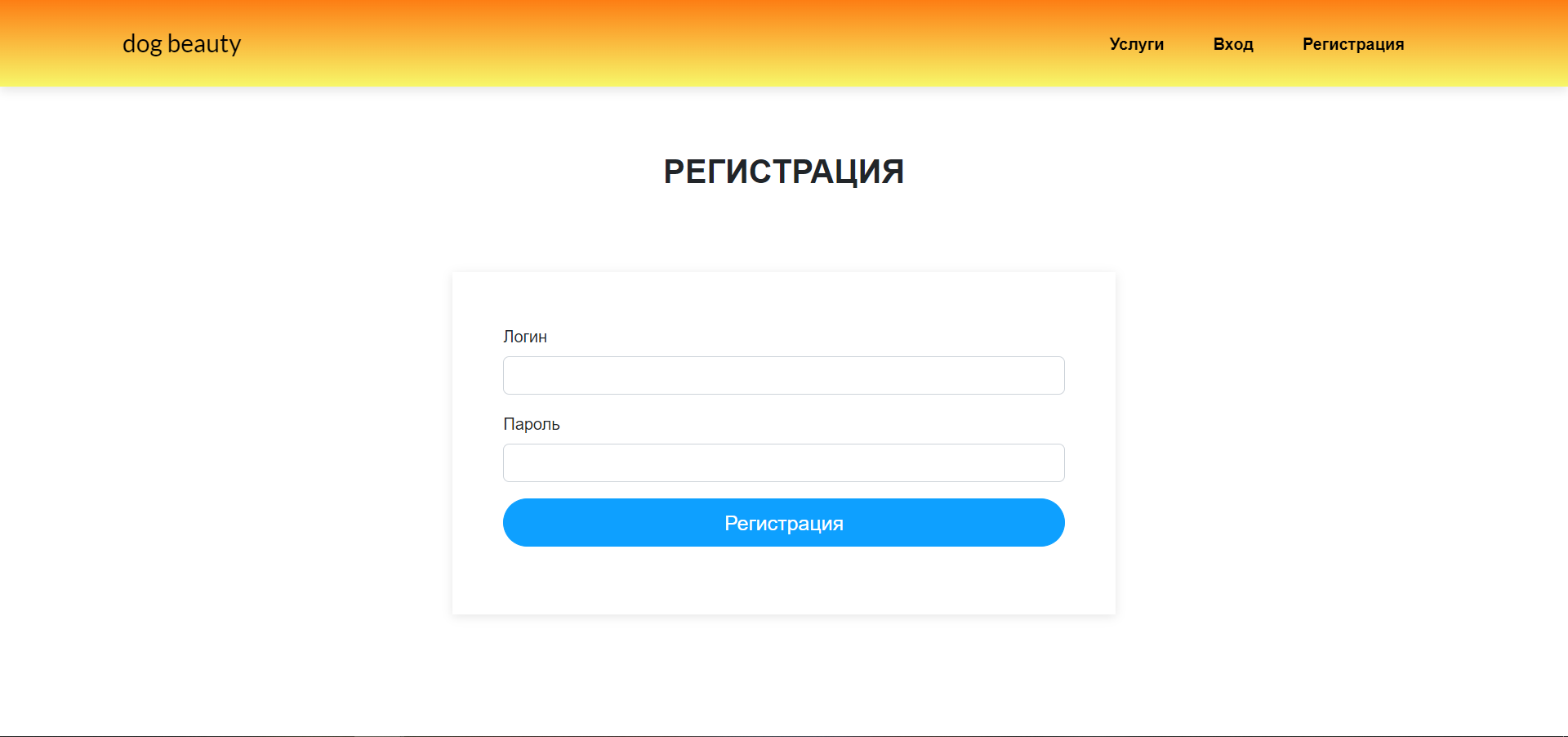


Рисунок 4.3 – Главная страница без входа в систему

После входа в уже зарегистрированный аккаунт под ролью администратора, будет доступен функционал из панели сверху страницы, это страница просмотра услуг, страница для создания и редакции категорий услуг, страница просмотра грумеров, статистики и управления ролями зарегистрированных пользователей (см. рис. 4.4).

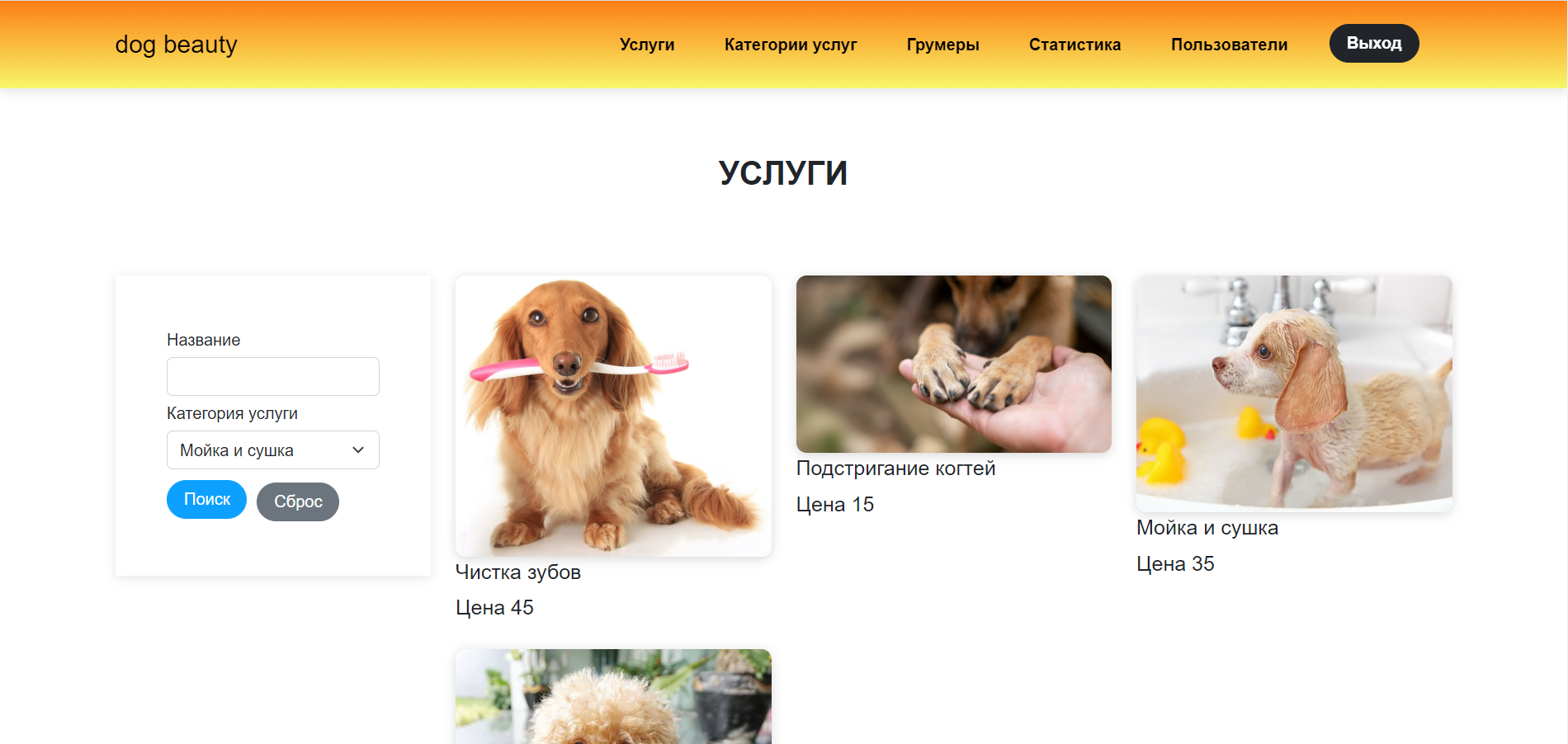


Рисунок 4.4 – Пункты меню для администратора

На странице «Услуги» можно просмотреть все услуги предоставляемые груминг салоном, их стоимость и описание к ним, а также при необходимости осуществить запись на услугу (см. рис. 4.5).

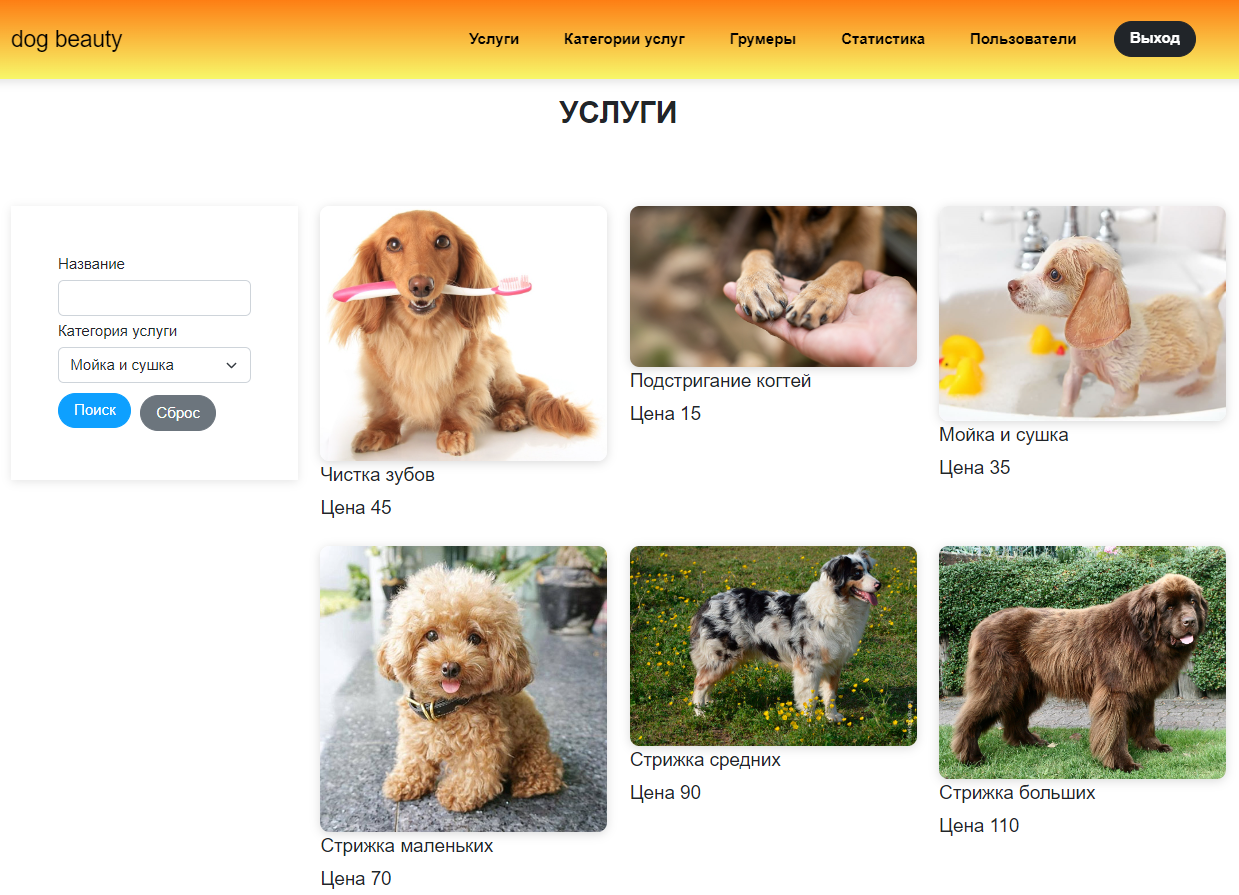


Рисунок 4.5 – Страница «Услуги»

На странице «Категории услуг» администратор может добавлять новые категории услуг, изменять названия уже имеющихся, а таже удалять их (см. рис. 4.6).

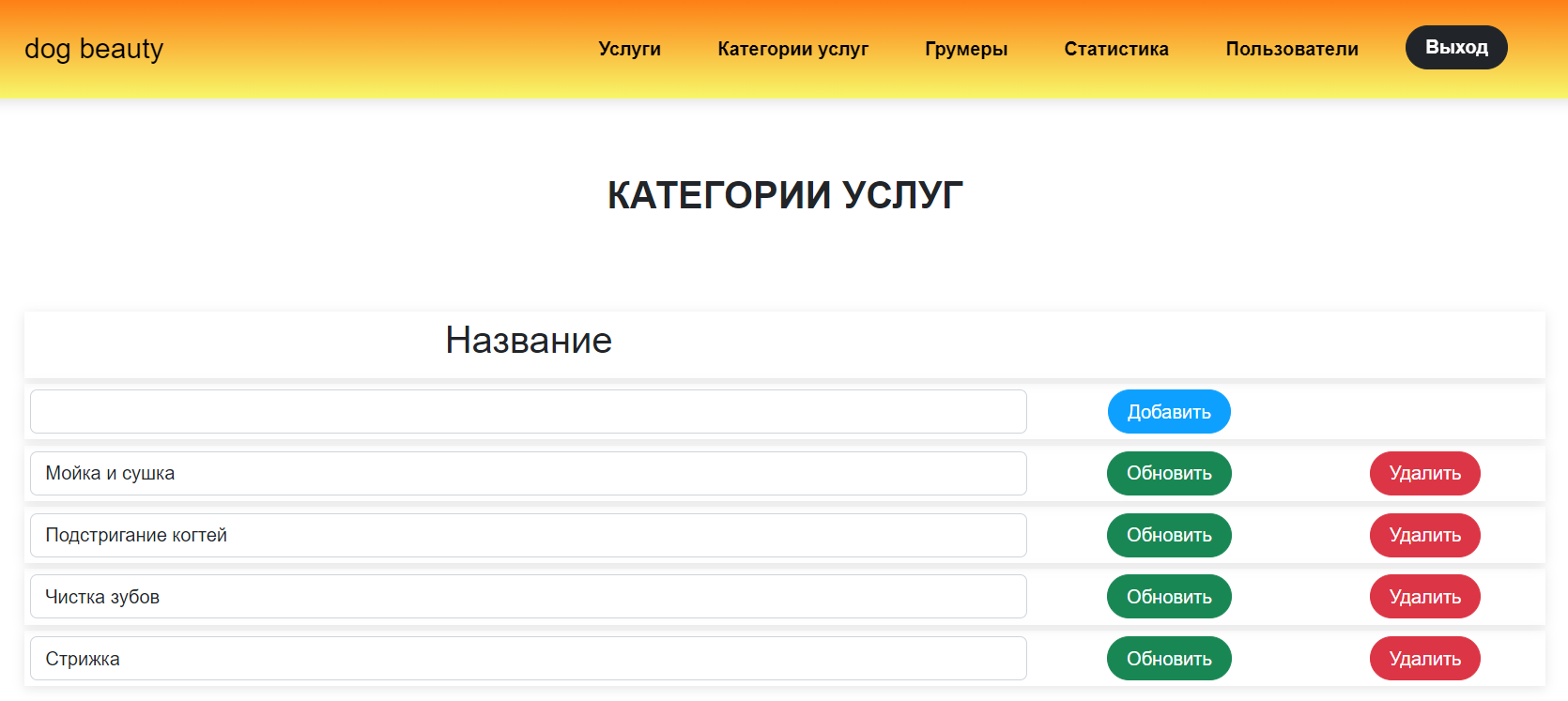


Рисунок 4.6 – Страница «Категории услуг»

На странице «Грумеры» можно увидеть профили грумеров рисунок 4.7

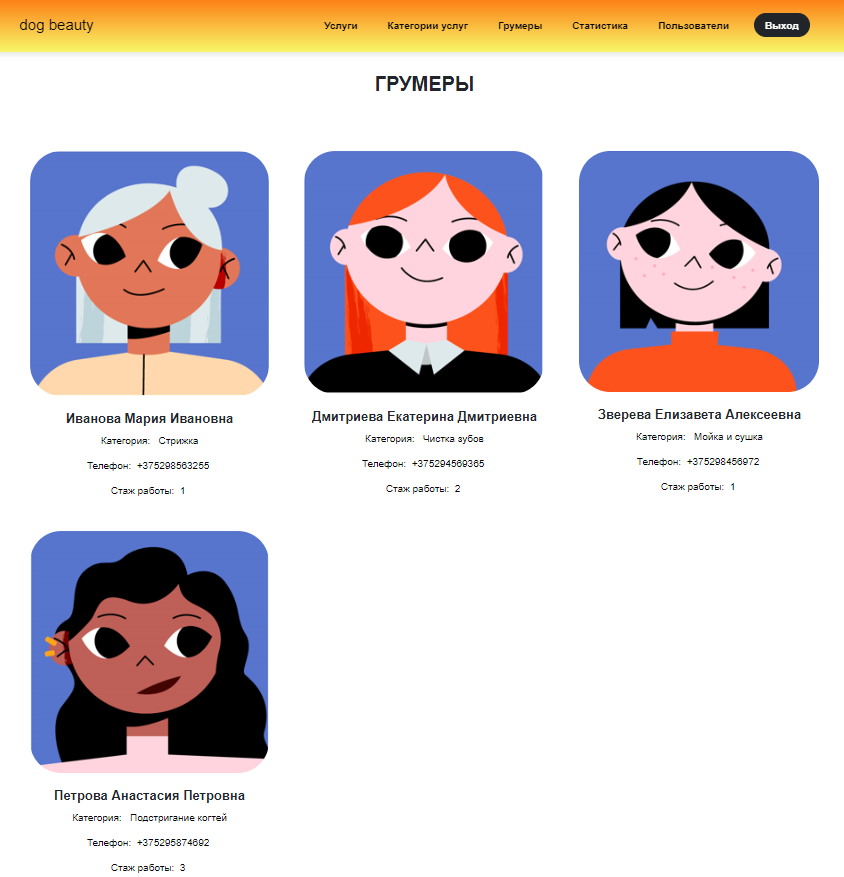


Рисунок 4.7 – Страница «Грумеры»

На странице «Статистика» администратор может просматривать аналитику груминг салона, где показана прибыль по каждой услуге (см. рис. 4.8).



Рисунок 4.8 – Страница «Статистика»

На странице «Пользователи» администратор может, как уже было описано выше, присваивать роли пользователям, а также удалять их (см. рис. 4.9).

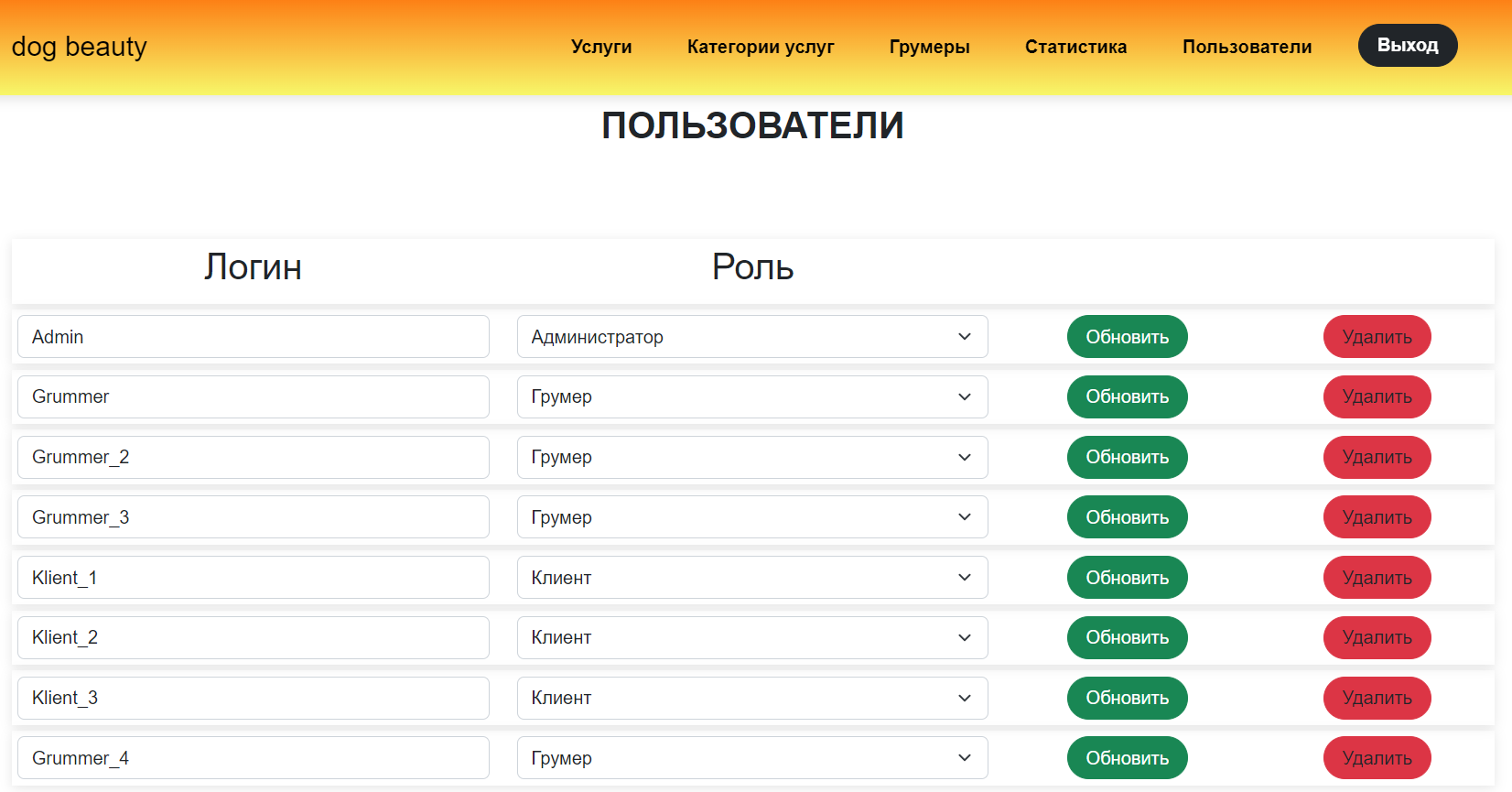


Рисунок 4.9 – Страница «Пользователи»

После входа в уже зарегистрированный аккаунт под ролью грумера, будет доступен функционал из панели сверху страницы, это страница просмотра услуг, оформления и изменения собственного профиля, страница просмотра грумеров, страница просмотра архива записей, страница для создания услуг (см. рис. 4.10).

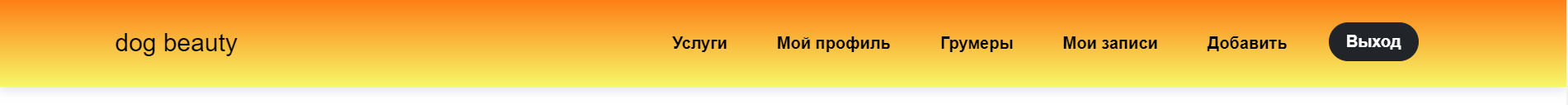


Рисунок 4.10 – Пункты меню для грумера

На странице «Услуги» можно просмотреть все услуги предоставляемые груминг салоном, их стоимость и описание к ним (см. рис. 4.11).

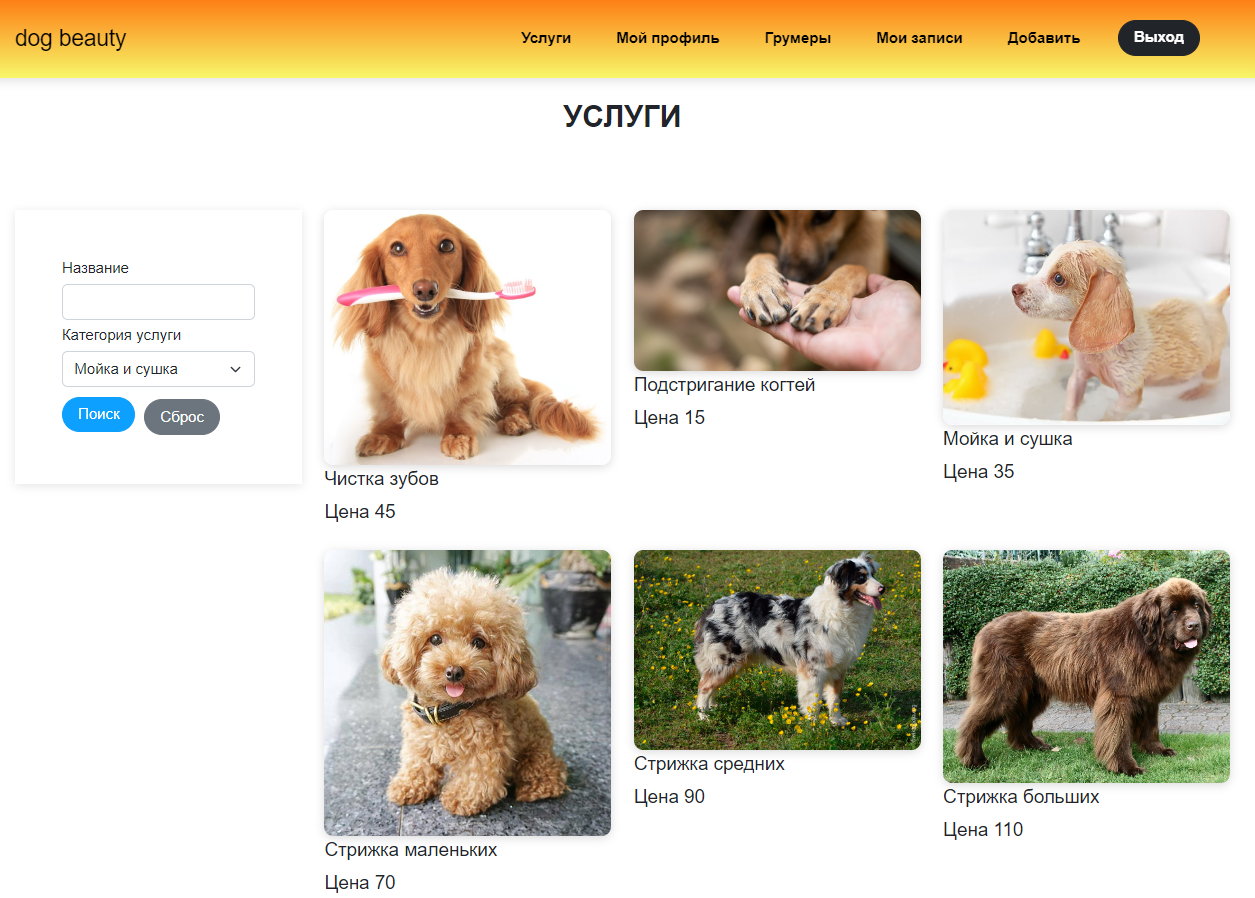


Рисунок 4.11 – Страница «Услуги»

На странице «Мой профиль» грумер заполняет данные о себе, а это ФИО, категорию услуг в которой он работает, телефон, а также стаж (см. рис. 4.12).

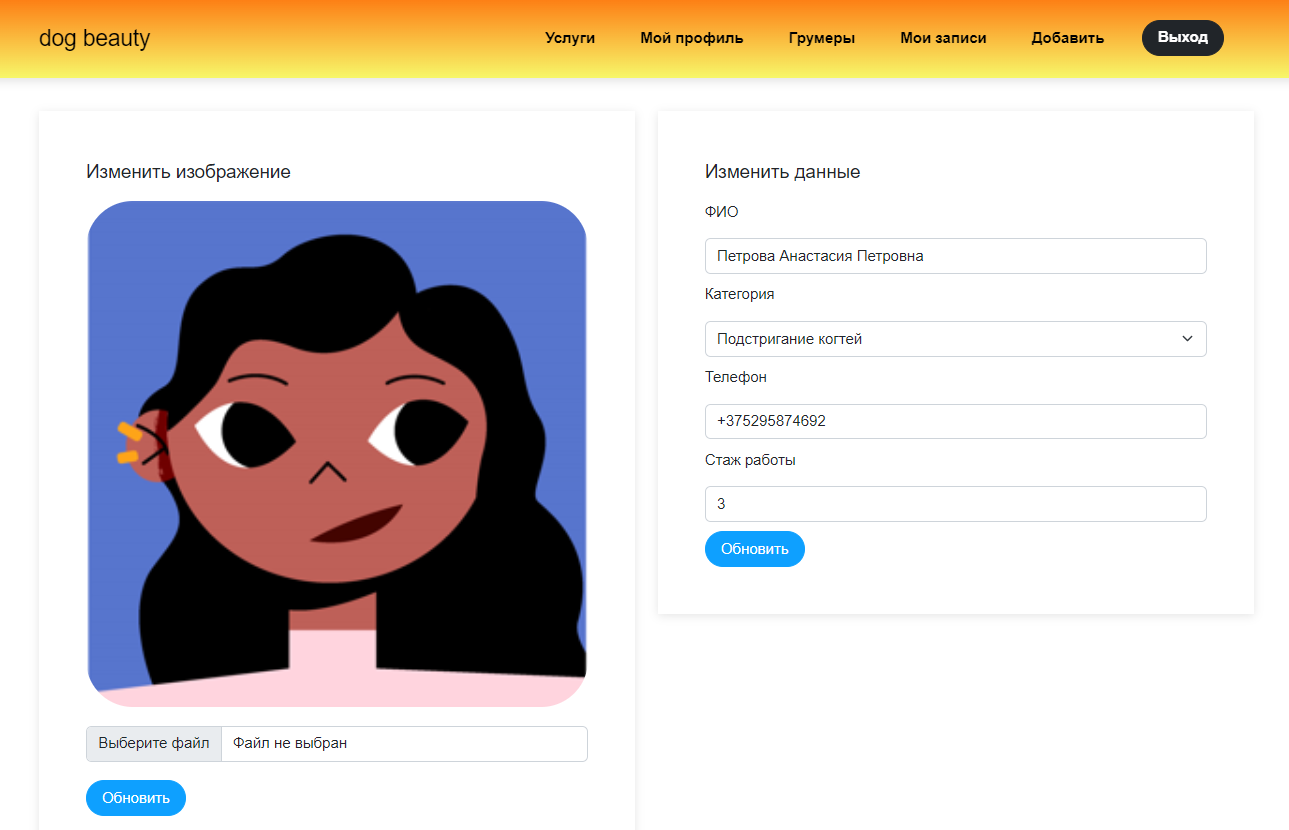


Рисунок 4.12 – Страница «Мой профиль»

На странице «Грумеры» можно просмотреть профили грумеров (см. рис. 4.13).

На странице «Мои записи» грумер может видеть все свои записи, а также подтверждать или отклонять их и устанавливать им статус «Выполнено» после оказании услуги (см. рис. 4.14).

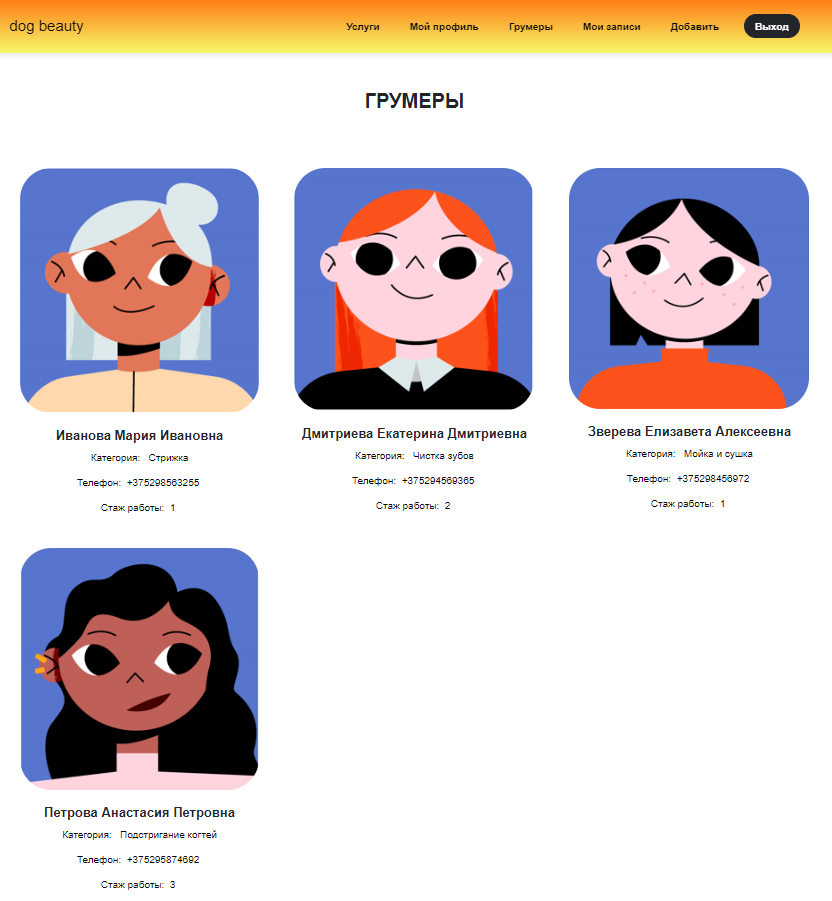


Рисунок 4.13 – Страница «Грумеры»

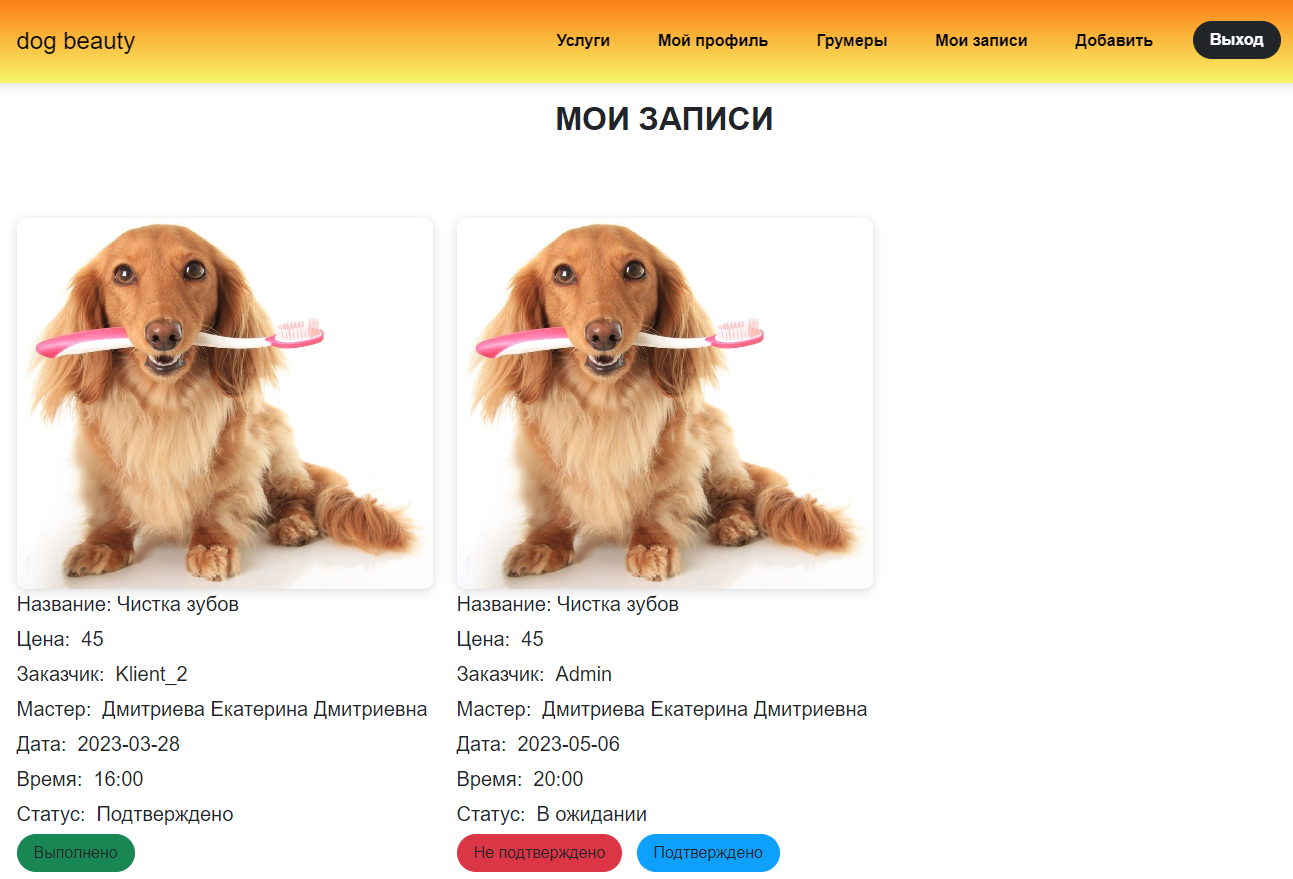


Рисунок 4.14 – Страница «Мои записи»

На странице «Добавить услугу» грумер добавляет выполняемые им услуги и описание к ним (название, категория, фото, цена и описание) (см. рис. 4.15).

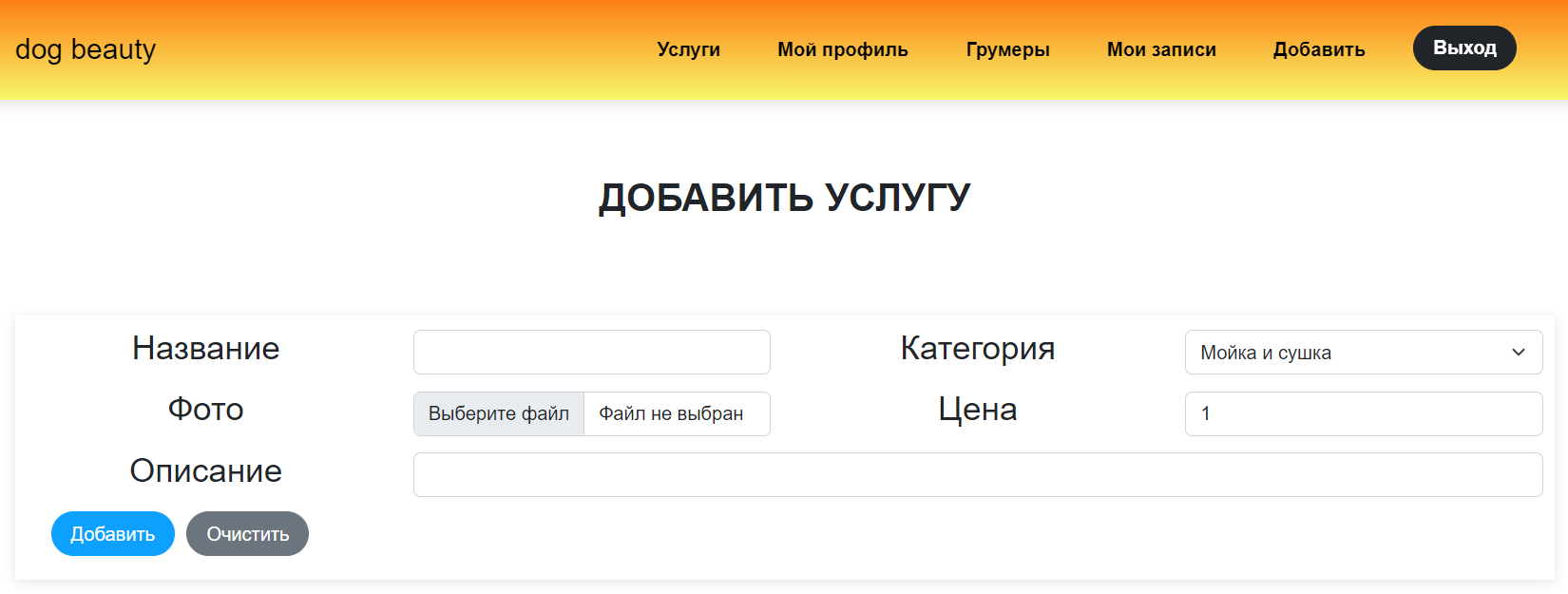


Рисунок 4.15 – Страница «Добавить услугу»

После входа в уже зарегистрированный аккаунт под ролью клиента, будет доступен функционал из панели сверху страницы, это страница просмотра услуг и оформления записи на услугу, страница просмотра грумеров, страница просмотра архива записей (см. рис. 4.16).



Рисунок 4.16 – Пункты меню для клиента

На странице «Услуги» можно просмотреть все услуги предоставляемые груминг салоном, их стоимость и описание к ним, а также записаться (см. рис. 4.17).



Рисунок 4.17 – Страница «Услуги»

Для записи необходимо нажать на интересующую услугу (их можно фильтровать по категориям и названию) и откроется страница услуги с ее полным описанием и формой для выбора даты и времени (см. рис. 4.18).

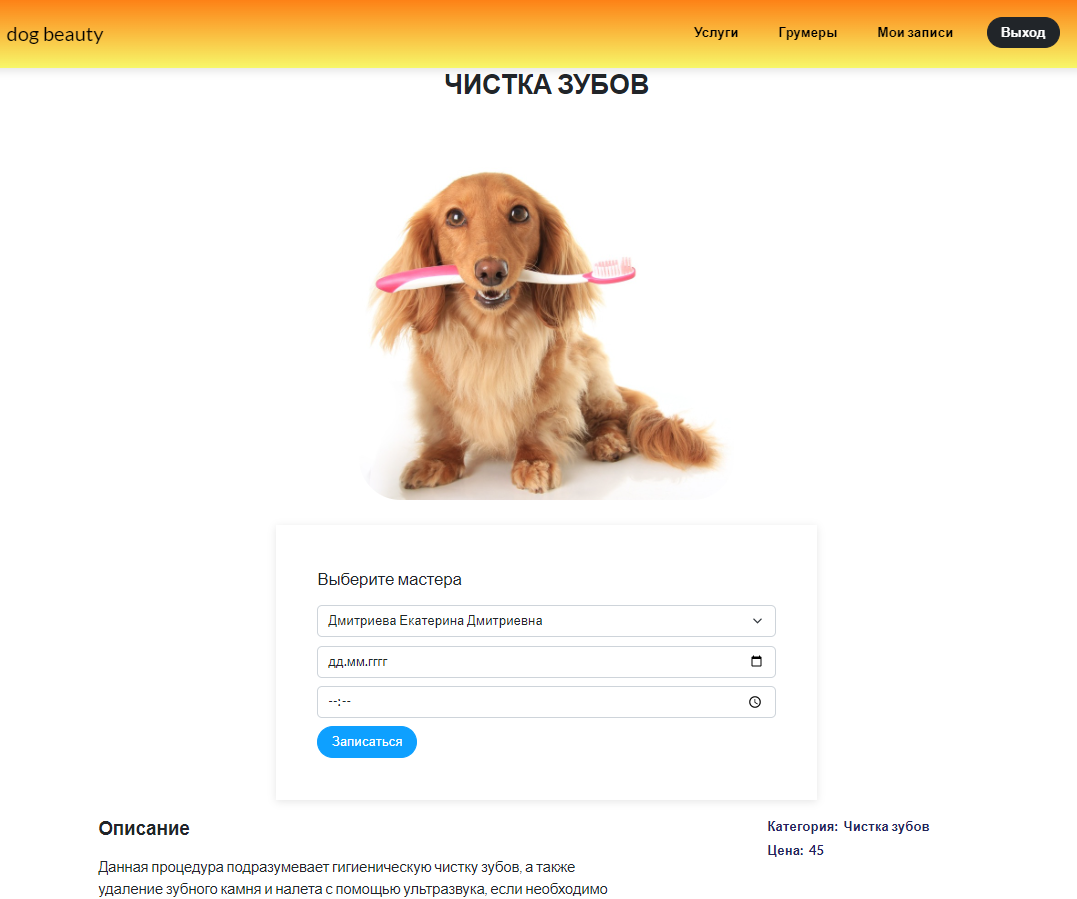


Рисунок 4.18 – Страница записи на услугу

На странице «Грумеры» можно просмотреть профили грумеров (см. рис. 4.19).

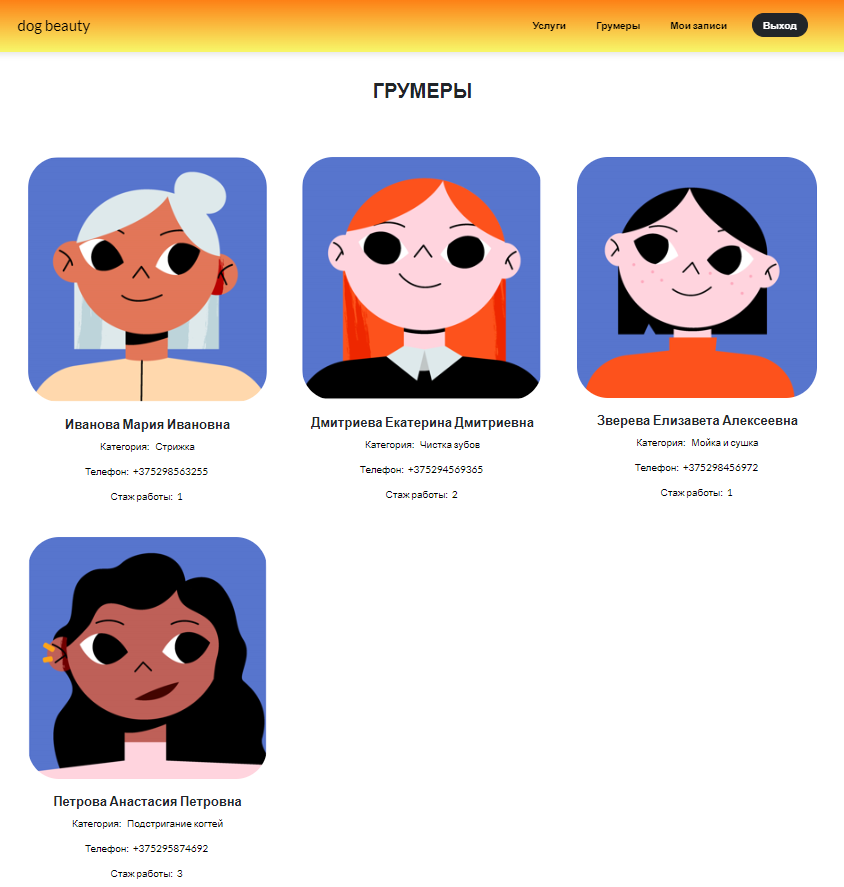


Рисунок 4.19 – Страница «Грумеры»

На странице «Мои записи» клиент может видеть все свои записи, а также их статус (см. рис. 4.20).



Рисунок 4.20 – Страница «Мои записи»

# **ЗАКЛЮЧЕНИЕ**

В результате проделанной работы, целью которой являлось создание веб-приложения, автоматизирующего процесс работы администрации груминг салона, в частности была произведена автоматизация процесса записи на услуги.

Программное средство было написано при помощи языке программирования Java. Для веб-приложения характерны такие свойства, как модифицируемость и расширяемость.

С целью обеспечения безопасности данных были использованы современные технологии авторизации и аутентификации, а также ограничения доступа к ресурсам, доступ к которым имеют только определенные группы пользователей.

Данный проект имеет удобный и доступный пользовательский интерфейс (внешнее оформление), который делает веб-сайт простым для использования любыми посетителями. Помимо этого, реализовано множество дополнений и улучшений, упрощающих работу, что сделало пользовательский интерфейс гораздо более удобным и понятным.

По окончанию разработки программы были выполнены все поставленные задачи. Данная программа является законченной, но имеет возможность обновления, усовершенствования и масштабирования при необходимости, в будущем данная задача не составит большого труда благодаря применению объектно-ориентированного программирования.

Разработанный сайт достаточно успешно может автоматизировать работу груминг салона и его администратора, увеличить число клиентов для грумеров, улучшить взаимодействие салона с клиентами, сократить время на выполнение повседневных задач, повысить эффективность работы и улучшить качество обслуживания клиентов. Клиенты избавляются от рутинной поездки или звонков в салон, поскольку могут записывать своих любимцев на процедуру дистанционно. Это позволяет администратору максимально быстро обрабатывать огромное количество записей. Разработанный сайт облегчает жизнь администратору не только благодаря быстрой обработке поступающих заказов, а также благодаря централизованному хранению информации в базе данных. В результате, груминг-салон может получить более довольных клиентов и увеличить прибыль. Внедрение веб-приложения также помогает салону оставаться конкурентоспособным и современным в условиях быстро меняющегося рынка.

# **Список использованных источников**

[1] Научная электронная библиотека [электронный ресурс]: Методология моделирования предметной области // Научная электронная библиотека. – Режим доступа: <https://monographies.ru/ru/book/section?id=4627>

[2] Использование диаграммы вариантов проектирования UML при проектировании программного обеспечения / Хабр. [электронный ресурс]: Использование диаграммы вариантов проектирования UML при проектировании программного обеспечения // Все статьи подряд / Хабр. – Режим доступа: <https://habr.com/ru/articles/566218>

[3] НОУ ИНТУИТ | Лекция | Элементы графической нотации диаграммы вариантов использования [электронный ресурс]: Отношения на диаграмме вариантов использования // Национальный Открытый Университет "ИНТУИТ" | Бесплатное образование. – Режим доступа: <https://intuit.ru/studies/courses/32/32/lecture/1004?page=2>

[4] Информационные системы [электронный ресурс]: Лекция № 2 «Понятие предметной области» // Новости - РФ ПГУ. – Режим доступа: http://rfpgu.ru/booksystem/page20.html

[5] Диаграммы UML для моделирования процессов и архитектуры проекта [электронный ресурс]: UML для бизнес-моделирования: зачем нужны диаграммы процессов // Evergreen - web разработка и диджитализация бизнеса с помощью AI продуктов. – Режим доступа: <https://evergreens.com.ua/ru/articles/uml-diagrams.html>

[6] Теория и практика UML. Диаграмма состояний | Статьи [электронный ресурс]: Теория и практика UML. Диаграмма состояний // Библиотека международных, региональных и национальных стандартов оформления проектной документации (ГОСТы, СНИП, СТ РК, ИСО). – Режим доступа: <http://it-gost.ru/articles/view_articles/97>

[7] PostgreSQL — популярная свободная объектно-реляционная система управления базами данных [электронный ресурс]: PostgreSQL — объектно-реляционная система управления базами данных // Web Creator — разработка сложных сайтов и веб-приложений, комплексный интернет-маркетинг. – Режим доступа: <https://web-creator.ru/articles/postgresql>

[8] Сравнение PostgreSQL с MySQL: в чем разница? [электронный ресурс]: PostgreSQL vs MySQL: какая система подходит вашему бизнесу // Cloud — облачный провайдер IaaS/PaaS и ML сервисов. – Режим доступа: <https://cloud.ru/ru/warp/blog/mysql-vs-postgresql>

[9] Spring Framework - что это за фреймворк для Java [электронный ресурс]: Spring // Онлайн-школа Skillfactory — онлайн-обучение востребованным IT-профессиям. – Режим доступа: <https://blog.skillfactory.ru/glossary/spring/>

[10] Что такое MVC: рассказываем простыми словами [электронный ресурс]: Что такое MVC: рассказываем простыми словами // Хекслет — больше чем школа программирования. Онлайн курсы, сообщество программистов. – Режим доступа: <https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami>

[11] Краткий обзор Spring Security / Хабр [электронный ресурс]: Краткий обзор Spring Security // Все статьи подряд / Хабр. – Режим доступа: <https://habr.com/ru/articles/203318>

[12] Hibernate - что это за фреймворк для Java: ORM-модель [электронный ресурс]: Hibernate // Онлайн-школа Skillfactory — онлайн-обучение востребованным IT-профессиям. – Режим доступа: https://blog.skillfactory.ru/glossary/hibernate/

[13] Леоненков А. В. Самоучитель UML / А.В. Леоненков. — СПб.: БХВ-Петербург, 2007. — 576 с.

[14] Диаграммы классов - Унифицированный язык моделирования UML [электронный ресурс]: Диаграмма классов // Vuzlit. – Режим доступа: <https://vuzlit.com/1030025/diagrammy_klassov>

[15] Различия между абстрактными классами и интерфейсами в Java [электронный ресурс]: Различия между абстрактными классами и интерфейсами в Java // Курсы программирования на Java. – Режим доступа: <https://topjava.ru/blog/razlichiya-mezhdu-abstraktnymi-klassami-i-interfeysami-v-java>

[16] Что такое бизнес-процесс и как его создать? — Разработка на vc.ru [электронный ресурс]: Что такое бизнес-процесс и как его создать? // vc.ru — бизнес, технологии, идеи, модели роста, стартапы. - Режим доступа: <https://vc.ru/dev/378228-chto-takoe-biznes-process-i-kak-ego-sozdat>

[17] Проектирование интерфейсов пользователя : пособие для студентов специальности 1-47 01 02 «Дизайн электронных и веб-изданий» / Т. П. Брусенцова, Т. В. Кишкурно. – Минск : БГТУ, 2019. – 172 с.

# **ПРИЛОЖЕНИЕ А**

# **(обязательное)**

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

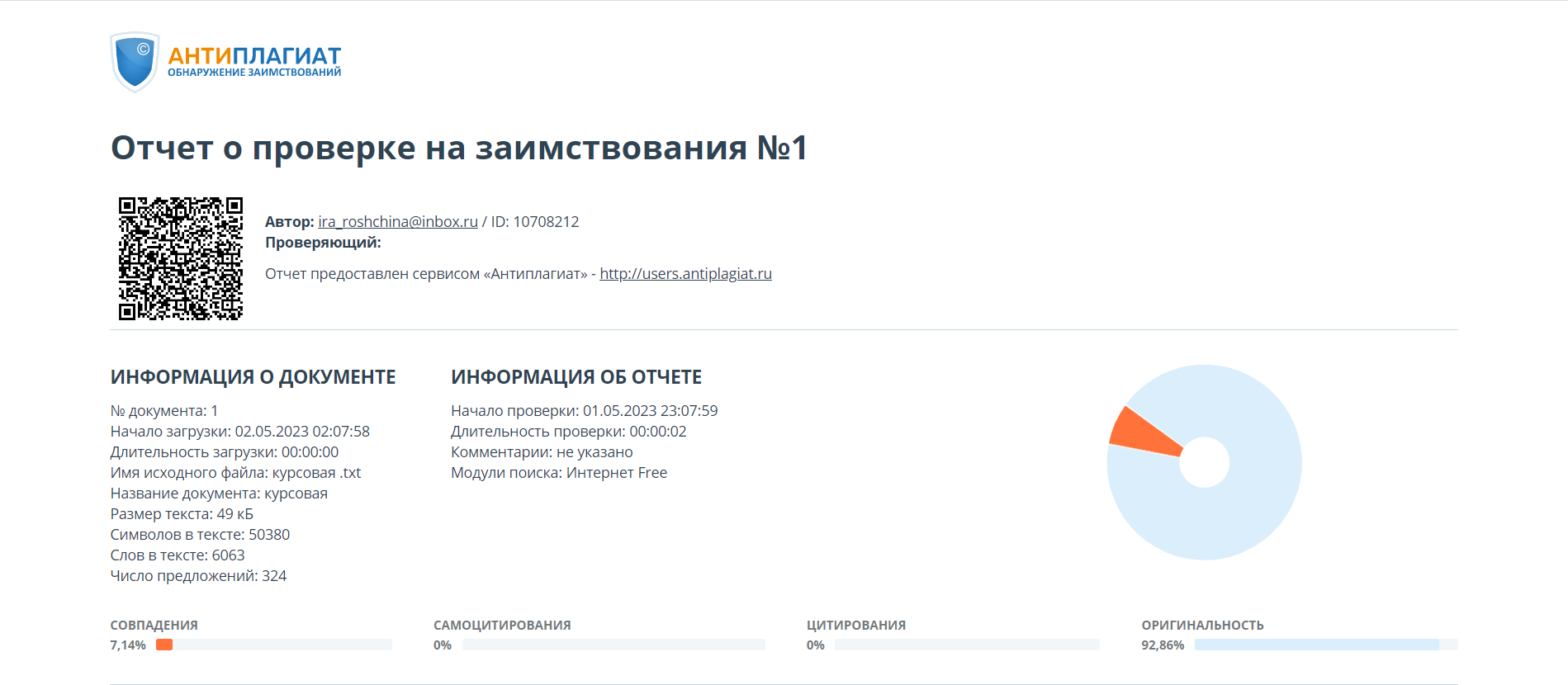


Рисунок А ­– Проверка на заимствования

# **ПРИЛОЖЕНИЕ Б**

# **(обязательное)**

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

package com.dog\_beauty.controller;

import com.dog\_beauty.controller.main.Attributes;

import com.dog\_beauty.model.\*;

import com.dog\_beauty.model.enums.StatusOrdering;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

import org.springframework.web.multipart.MultipartFile;

import java.io.File;

import java.util.List;

import java.util.Objects;

import java.util.UUID;

@Controller

@RequestMapping("/notes")

public class NotesCont extends Attributes {

@GetMapping("/add")

public String noteAdd(Model model) {

AddAttributesNoteAdd(model);

return "noteAdd";

}

@GetMapping("/my")

public String orderingsMy(Model model) {

AddAttributesNotesMy(model);

return "notesMy";

}

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

public String orderingNotConf(@PathVariable Long id) {

Ordering ordering = orderingRepo.getReferenceById(id);

ordering.setStatusOrdering(StatusOrdering.NOT\_CONF);

orderingRepo.save(ordering);

return "redirect:/notes/my";

}

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

public String orderingConf(@PathVariable Long id) {

Ordering ordering = orderingRepo.getReferenceById(id);

ordering.setStatusOrdering(StatusOrdering.CONF);

orderingRepo.save(ordering);

return "redirect:/notes/my";

}

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

public String orderingDone(@PathVariable Long id) {

Ordering ordering = orderingRepo.getReferenceById(id);

ordering.setStatusOrdering(StatusOrdering.DONE);

ordering.getNote().setCount(ordering.getNote().getCount() + 1);

orderingRepo.save(ordering);

return "redirect:/notes/my";

}

@GetMapping("/{id}")

public String note(Model model, @PathVariable Long id) {

AddAttributesNote(model, id);

return "note";

}

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

public String noteEdit(Model model, @PathVariable Long id) {

AddAttributesNoteEdit(model, id);

return "noteEdit";

}

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

public String noteDelete(@PathVariable Long id) {

List<Ordering> orderings = orderingRepo.findAllByNote\_Id(id);

for (Ordering i : orderings){

orderingRepo.deleteById(i.getId());

}

notesRepo.deleteById(id);

return "redirect:/";

}

@PostMapping("/ordering/{noteId}")

public String noteOrdering(@RequestParam Long masterId, @RequestParam String date, @RequestParam String time, @PathVariable Long noteId) {

Masters master = mastersRepo.getReferenceById(masterId);

Users client = getUser();

Notes note = notesRepo.getReferenceById(noteId);

Ordering ordering = new Ordering(master, client, note, date, time);

orderingRepo.save(ordering);

return "redirect:/notes/{noteId}";

}

@PostMapping("/add")

public String noteAddNew(Model model, @RequestParam String name, @RequestParam Long categoryId, @RequestParam MultipartFile photo, @RequestParam int price, @RequestParam String description) {

String res = "";

if (photo != null && !Objects.requireNonNull(photo.getOriginalFilename()).isEmpty()) {

String uuidFile = UUID.randomUUID().toString();

boolean createDir = true;

try {

File uploadDir = new File(uploadImg);

if (!uploadDir.exists()) createDir = uploadDir.mkdir();

if (createDir) {

res = "notes/" + uuidFile + "\_" + photo.getOriginalFilename();

photo.transferTo(new File(uploadImg + "/" + res));

}

} catch (Exception e) {

model.addAttribute("message", "Некорректный данные!");

AddAttributesNoteAdd(model);

return "noteAdd";

}

}

Notes note = new Notes(name, res, price, description);

Category category = categoryRepo.getReferenceById(categoryId);

category.addNote(note);

categoryRepo.save(category);

return "redirect:/notes/add";

}

@PostMapping("/edit/{id}")

public String noteEditOld(Model model, @RequestParam String name, @RequestParam Long categoryId, @RequestParam MultipartFile photo, @RequestParam int price, @RequestParam String description, @PathVariable Long id) {

Notes note = notesRepo.getReferenceById(id);

if (photo != null && !Objects.requireNonNull(photo.getOriginalFilename()).isEmpty()) {

String res = "";

String uuidFile = UUID.randomUUID().toString();

boolean createDir = true;

try {

File uploadDir = new File(uploadImg);

if (!uploadDir.exists()) createDir = uploadDir.mkdir();

if (createDir) {

res = "notes/" + uuidFile + "\_" + photo.getOriginalFilename();

photo.transferTo(new File(uploadImg + "/" + res));

}

} catch (Exception e) {

model.addAttribute("message", "Некорректный данные!");

AddAttributesNoteEdit(model, id);

return "noteEdit";

}

note.setPhoto(res);

}

note.setName(name);

note.setPrice(price);

note.setDescription(description);

Category categoryNew = categoryRepo.getReferenceById(categoryId);

if ((!categoryNew.getId().equals(note.getCategory().getId()))) {

Category categoryOld = note.getCategory();

categoryOld.removeNote(note);

categoryNew.addNote(note);

}

notesRepo.save(note);

return "redirect:/notes";

}

}

# **ПРИЛОЖЕНИе В**

# **(обязательное)**

# **Листинг скрипта генерации базы данных**

create sequence hibernate\_sequence;

alter sequence hibernate\_sequence owner to beauty;

create table category

(

id bigint not null

primary key,

name varchar(255)

);

alter table category

owner to beauty;

create table masters

(

id bigint not null

primary key,

category varchar(255),

experience smallint not null,

fio varchar(255),

photo varchar(255),

tel varchar(255)

);

alter table masters

owner to beauty;

create table users

(

id bigint not null

primary key,

password varchar(255),

role varchar(255),

username varchar(255),

master\_id bigint

constraint fkiq855vcnq82cvgvp7687svh4e

references masters

);

alter table users

owner to beauty;

create table notes

(

id bigint not null

primary key,

count integer not null,

description varchar(255),

name varchar(255),

photo varchar(255),

price integer not null,

category\_id bigint

constraint fkcj0t34ccxr2h8f1lit50od4qi

references category

);

alter table notes

owner to beauty;

create table ordering

(

id bigint not null

primary key,

date varchar(255),

time varchar(255),

client\_id bigint

constraint fkvqfjrn2uurn8ld0c6gsnm3da

references users,

master\_id bigint

constraint fkde68ciwjmipr8bxfsqoen7g6k

references masters,

note\_id bigint

constraint fkdhn0atvrhqm19yhkxk9dqh2uj

references notes,

status\_ordering varchar(255)

);

alter table ordering

owner to beauty;