СОДЕРЖАНИЕ

[Перечень условных обозначений, символов и терминов 6](#_Toc134557843)

[ВВЕДЕНИЕ 8](#_Toc134557844)

[1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПАРИКМАХЕРСКОЙ 10](#_Toc134557845)

[1.1 Описание парикмахерской 10](#_Toc134557846)

[1.2 Разработка функциональной модели предметной области 13](#_Toc134557847)

[1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований. 17](#_Toc134557848)

[1.4 Разработка информационной модели предметной области 18](#_Toc134557849)

[1.5 UML-модели представления программного средства и их описание. 20](#_Toc134557850)

[2 ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 24](#_Toc134557851)

[2.1 Постановка задачи 24](#_Toc134557852)

[2.2 Обоснование выбора компонентов и технологий для реализации программного средства 24](#_Toc134557853)

[2.3 Архитектурные решения 25](#_Toc134557854)

[2.4 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства 27](#_Toc134557855)

[2.5 Проектирование пользовательского интерфейса 30](#_Toc134557856)

[3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА 32](#_Toc134557857)

[4 РУКОВОДСТВО ПО РАЗВЕРТЫВНИЮ И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА 35](#_Toc134557858)

[4.1 Руководство по установке (развертыванию) и использованию программного средства 35](#_Toc134557859)

[4.2 Руководство пользователя 35](#_Toc134557860)

[ЗАКЛЮЧЕНИЕ 46](#_Toc134557861)

[ПРИЛОЖЕНИЕ А 48](#_Toc134557862)

[ПРИЛОЖЕНИЕ Б 49](#_Toc134557863)

[ПРИЛОЖЕНИЕ В 54](#_Toc134557864)

[ВЕДОМОСТЬ ДОКУМЕНТОВ КУРСОВОГО ПРОЕКТА 56](#_Toc134557865)

# Перечень условных обозначений, символов и терминов

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

# ВВЕДЕНИЕ

Посещение парикмахерской является одной из необходимых процедур по уходу за собой. Ведь парикмахерская – это то место, где можно освежиться, привести свой внешний вид в порядок, а также получить новый образ.

Парикмахерское искусство является одним их древнейших искусств в мире, так как его история уходит в далекий Древний Египет, примерно во второе тысячелетии до нашей эры. Египтяне владели техникой окрашивания волос хной, умели обесцвечивать их, а также выполняли завивку на специальном железном стержне. Мужчины из высших сословий полностью брили голову, а женщины делали короткие стрижки. На протяжении всего времени в моде были самые разнообразные стили и прически, однако, ближе к концу ХХ века модными стали более естественные прически, например распушенные волосы или локоны, собранные в конский хвост, и эта тенденция сохраняется по сегодняшний день.

Современная парикмахерская­­ –­­­­ это предприятие, занимающееся предоставлением услуг для населения по уходу за волосами в оборудованном специально для этого помещении.  С каждым годом число разнообразных услуг увеличивается, и поэтому довольно часто возникают проблемы с учетом деятельности.

В курсовом проекте актуальна проблема со временем записи клиентов, так как в программе намного быстрее вводить данные, чем вручную, компьютер сразу же осуществляет подсчет всех необходимых значений, в то время как любой из менеджеров может совершить ошибку в расчетах. Поэтому данная программа поможет оптимизировать работу парикмахерской.

Таким образом, целью данного курсового проекта является повышение эффективности и качества управления парикмахерской путем автоматизации процессов учета оказанных услуг внутри системы и управления информацией о клиентах и парикмахерах.

В данном курсовом проекте поставленная цель достигается автоматизации работы парикмахерской, то есть снижением времени работы с одним клиентом, а следовательно, повышением производительности работы парикмахерской в целом, сокращение ошибок при работе с клиентами, повышение прибыльности.

Поставленная цель потребовала решения следующих задач:

– провести анализ и моделирование учета выплат стипендий студентам, описать систему работы университета, разработать функциональную, информационную и UML модели;

– спроектировать программное средство, описать используемые технологии, рассмотреть уже имеющиеся аналоги, описать основные алгоритмы, обосновать выбор компонентов и технологий для реализации программного средства;

– протестировать и проверить работоспособность программного средства;

– описать инструкцию по развертыванию программного средства для администратора и клиента.

Объектом исследования является процесс учета деятельности парикмахерской.

Ключевые слова: АЛГОРИТМ, ФУНКЦИЯ, БЛОК-СХЕМА, ПАРИКМАХЕРСКАЯ, СТРИЖКА, КЛАСС, ПРОГРАММА, ПОЛЬЗОВАТЕЛЬ, АДМИНИСТРАТОР, ФАЙЛ, КЛИЕНТЫ, ПРИЛОЖЕНИЕ.

# АНАЛИЗ И МОДЕЛИРОВАНИЕ ПАРИКМАХЕРСКОЙ

## Описание парикмахерской

Парикмахерские услуги относятся к сфере бытовых услуг и принадлежат к числу наиболее востребованных в повседневном быту людей, а, следовательно, являются одними из самых массовых. Этот сектор услуг занимает наибольший объем на рынке бытового обслуживания населения и на данный момент представлен парикмахерскими, салонами, косметическими кабинетами и другими негосударственными предприятиями, составляющими 1/6 от общего числа предприятий сервиса. На рисунке 1.1 представлена структура бытовых услуг на данный момент.

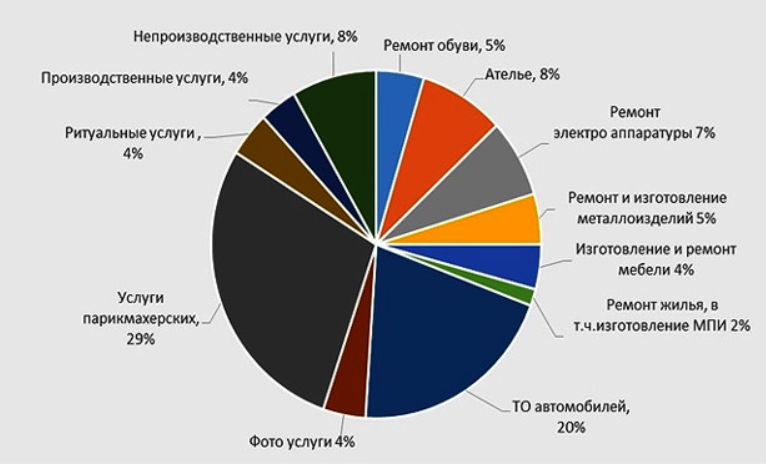


Рисунок 1.1 – Структура бытовых услуг

На сегодняшний день почти каждый пользуется услугами парикмахерских. Мужчины иначе смотрят на парикмахеров. Для них это больше, чем место, где можно стричься и бриться. Это место, где можно узнать новости сообщества, быть увиденным и определить, что означает мужественность на личном уровне. Во многих парикмахерских есть мужчины, которые приходят каждые 2–4 недели с постоянным приемом на обслуживание. Женщины это делают реже, но факт в том, что все хотят выглядеть опрятно, стильно, подобающе своему статусу. Ведь о человеке судят вначале по внешнему виду, а затем по его способностям: при встрече с незнакомцем прежде всего в глаза бросается его внешний вид, одежда, ухоженность волос, а после знакомства с ним, отдают должное его уму, знаниям.

В эпоху информационных технологий компьютеры и электронные гаджеты прочно вошли в нашу жизнь и изменили привычный уклад жизни человека прошлого столетия – человек ежедневно использует их для получения информации, все больше людей узнает об услугах парикмахерских, в следствии чего поток людей не утихает, а все увеличивается.

Помимо того, что парикмахерские предлагают свои профессиональные услуги и традиционный сопутствующий набор (журналы мод, музыка, новости), они идут дальше! Некоторым удаетсясоздать целый мир развлечений и окружить своих клиентов вниманием. Поскольку поход в парикмахерскую или салон красоты – это время отдыха, клиентам предлагаются закуски и напитки. Промоакции часто включают стрижку и кофе, например. Или какую-нибудь дополнительную процедуру по уходу за волосами.

Некоторые услуги, которые предоставляют парикмахерские:

* cтрижки мужские, женские;
* прически вечерние, торжественные, свадебные;
* укладки повседневные, для особых случаев;
* окраска волос;
* лечение волос;
* консультация по прическам и уходу за волосами.

Число парикмахерских с каждым годом все растет. В прямой пропорциональности росту парикмахерских растет и конкуренция между ними. И в выигрыше будут именно те, чья деятельность автоматизирована.

На рисунке 1.2 представлена таблица по числу парикмахерских по областям.

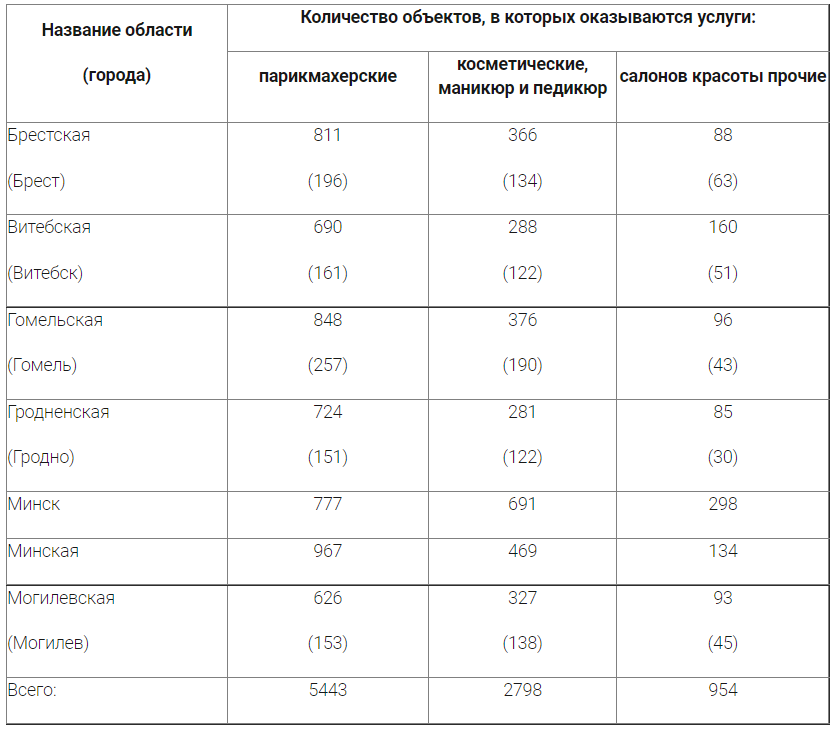


Рисунок 1.2 – Количество парикмахерских

Также парикмахерские делятся по сегментам:

* эконом;
* бизнес;
* премиум.

Эконом ­– это недорогие парикмахерские, расположенные в полуподвальных помещениях, либо в квартирах на первых этажах жилых домов, переоборудованных для предпринимательских целей. Организатор заведения обычно обходится минимальными расходами на косметический ремонт и рекламу. Несмотря на это, парикмахерские эконом-класса обеспечивают персоналу максимальную загруженность, примерно 60-65 % рабочего времени. Из выше сказанного вытекает то, что парикмахерским на сегодняшний день необходима автоматизация. Она повысит уровень эффективности, облегчит труд работников. Благодаря этим изменениям возрастет уровень конкурентоспособности на рынке.

Бизнес – это парикмахерские расположенные в центре города, в многолюдных местах, там работают мастера квалификации не ниже парикмахера-стилиста.  Нагрузка на персонал обычно достигает показателя 50 % от общего рабочего времени, а стоимость услуг в три-четыре раза дороже, чем в парикмахерских категории «эконом».

Премиум – это заведения, в которых предоставляются эксклюзивные услуги, встречаются не в каждом городе. Обычно их основателями или управляющими являются профессионалы с авторским подходом, которые достигли известности на российских и международных конкурсах и чемпионатах. Персонал здесь загружен максимум на 30 % от общего рабочего времени, но посетители постоянные и лояльные. Расценки обычно не разглашаются.

На рисунке 1.3 предоставлено общее число парикмахерских, распределенное по сегментам.

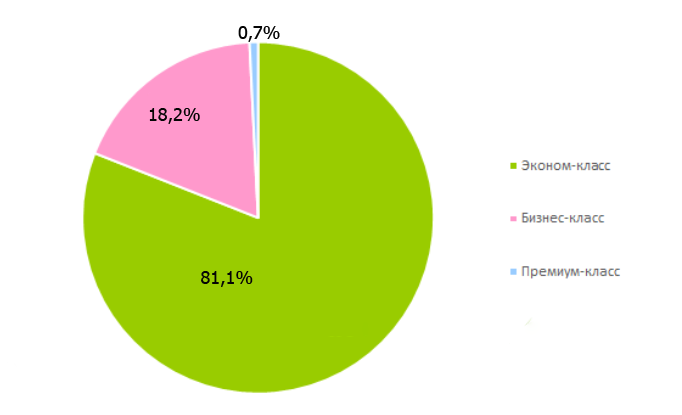


Рисунок 1.3 – Распределение парикмахерских по сегментам

## Разработка функциональной модели предметной области

Для наиболее наглядного представления об автоматизации управления парикмахерской была разработана функциональная модель.

Для создания функциональной модели был выбран стандарт IDEF0. IDEF0 — методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность.[1]

На контекстной диаграмме верхнего уровня (рисунок 1.4) представлена функциональная модель «Парикмахерская», а также определены потоки входных и выходных данных, механизмы ограничения и управления данными.

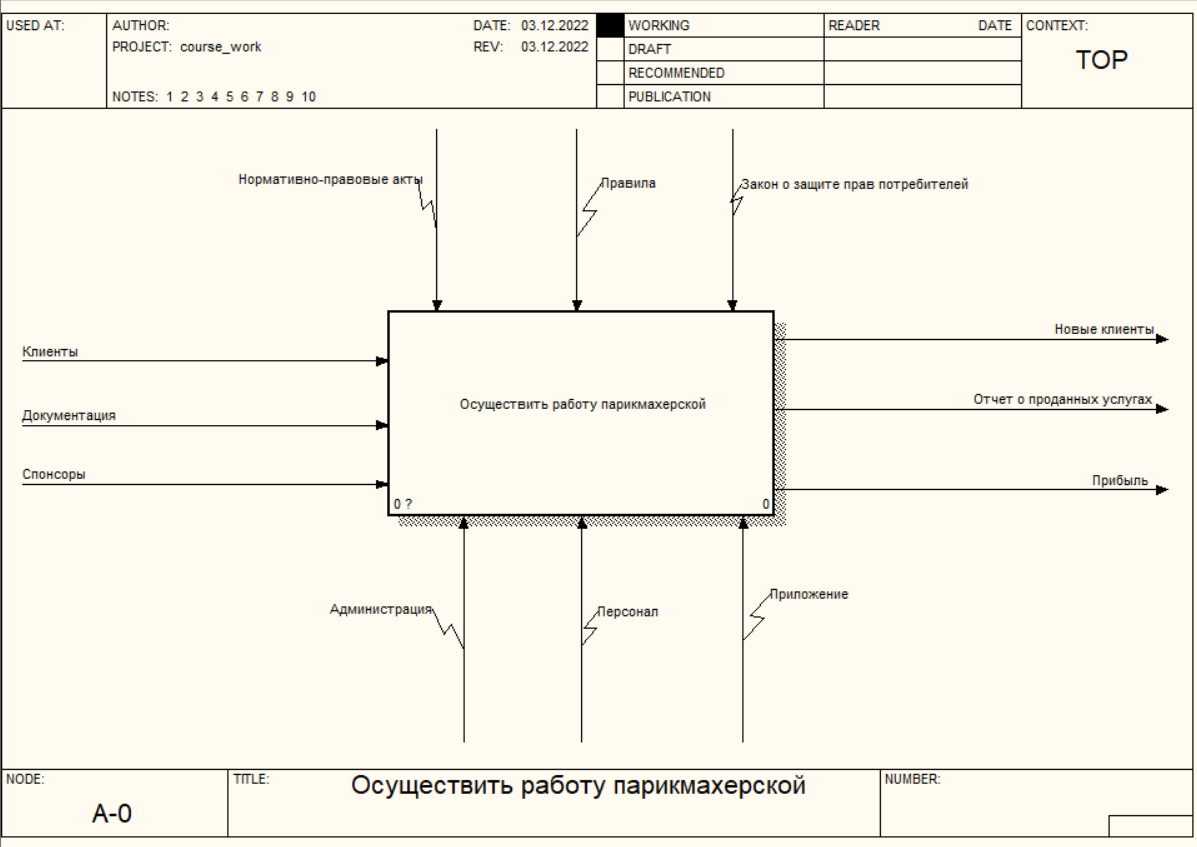


Рисунок 1.4 – Деятельность парикмахерской

Декомпозиция верхнего уровня представлена на рисунке 1.5 и состоит из 4 блоков: «Финансировать деятельность», «Разработать стратегию развития», «Обучить персонал», «Оказать услуги».

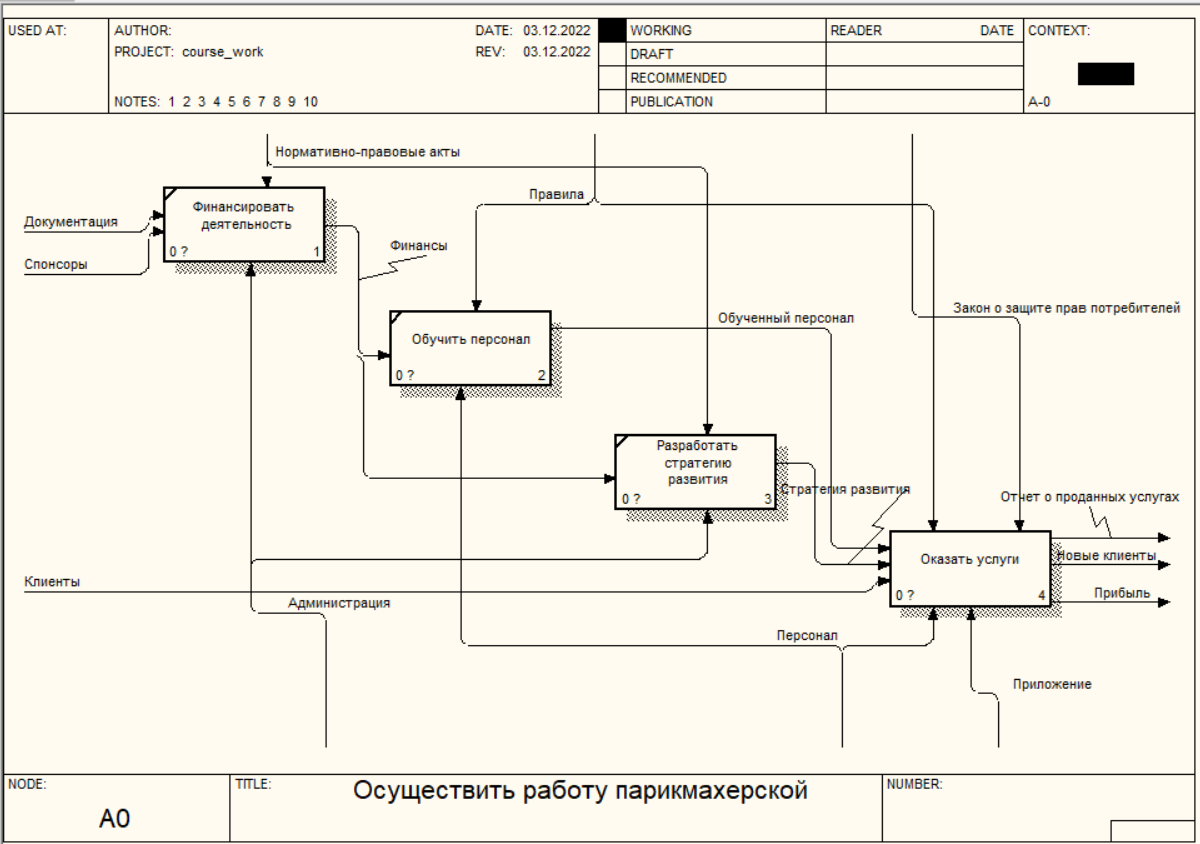


Рисунок 1.5 – Декомпозиция контекстной диаграммы верхнего уровня

На рисунке 1.6 изображена декомпозиция блока «Оказать услуги». Она включает в себя следующие блоки: «Проконсультировать клиента», «Выбрать услуги», «Записаться на услугу», «Выполнить услугу», «Осуществить оплату».

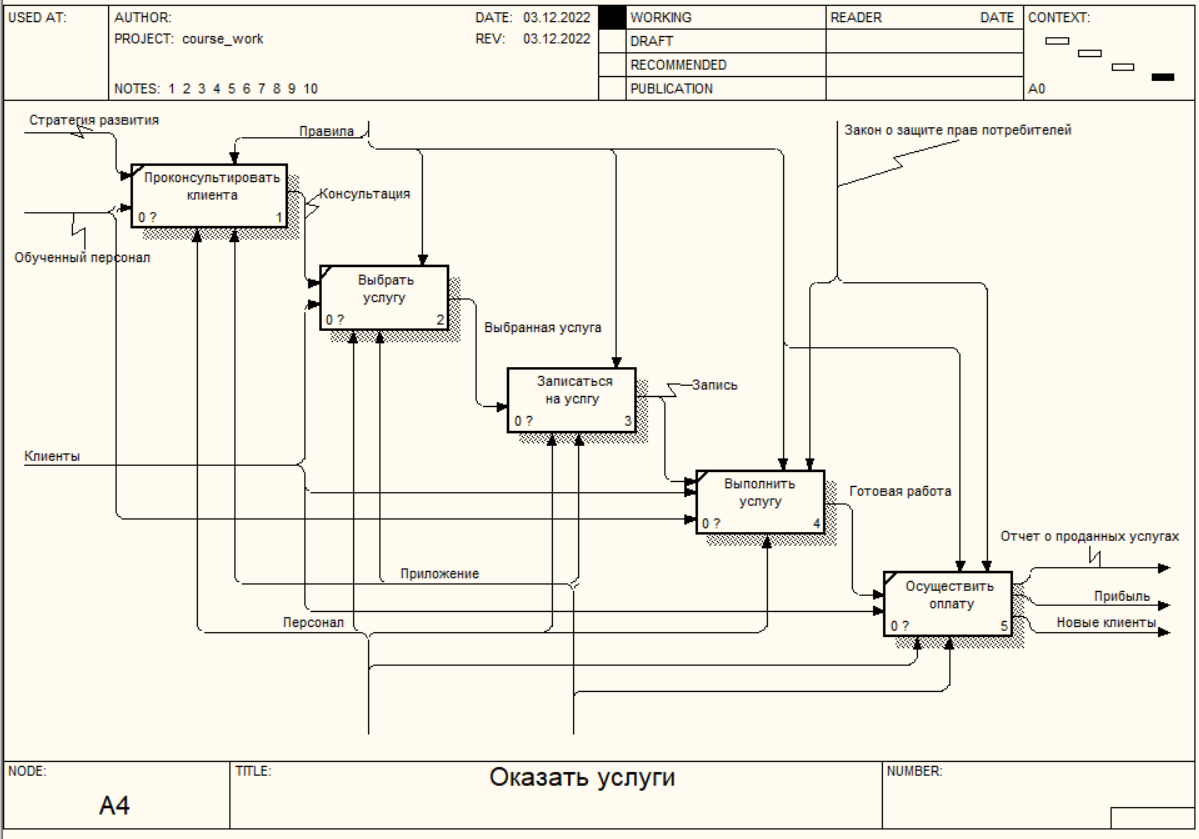


Рисунок 1.6 – Декомпозиция блока «Оказать услуги»

Декомпозиция блока «Записаться на услугу» представлена на рисунке 1.7 Она состоит из 3 блоков: «Выбрать парикмахера», «Выбрать дату» и «Занести изменения в расписание».

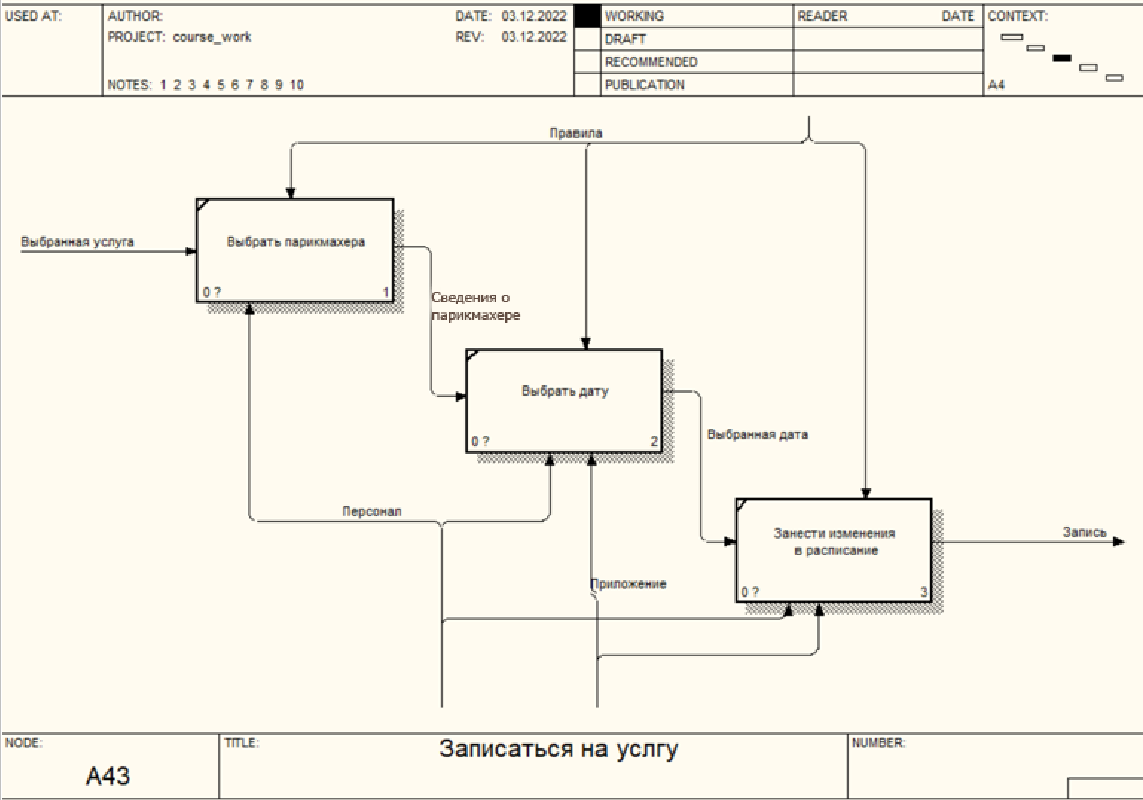


Рисунок 1.7 – Декомпозиция блока «Записаться на услугу»

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

Для автоматизации работы парикмахерской надо понять принцип работы любой парикмахерской.

Так у любого клиента есть свои предпочтения по прическе, мастеру и времени для стрижки. Необходимо спроектировать базу данных, которая хранила бы всю необходимую информацию о клиенте. Это ускорит поиск данных по определенному клиенту, что в свою очередь повысит эффективность работы менеджера и всего персонала в целом.

Для хранения информации будет использованa СУБД MySql.

Одной из главных задач программного средства является обеспечение возможности формирования расписания мастеров. Необходимо учесть занятость каждого мастера, так, если у одного мастера полностью занят день, на который хочет записаться клиент, необходимо предложить ему другого мастера, либо отменить запись.

Диаграмма вариантов использования (Use Case Diagram) определяет функциональное назначение моделируемой системы или предметной области.

Данная диаграмма отображает множество актеров, взаимодействующих с проектируемой системой (программным средством) с помощью вариантов использования. Таким образом, основными элементами диаграммы вариантов использования являются актер и вариант использования.[2]

В данном курсовом проекте актерами являются клиент, администратор и парикмахер (рисунок 1.8). Пользователь и администратор имеют такой вариант использования, как авторизация, которая предусматривает ввод логина и пароля (в скрытом виде). Кроме перечисленных вариантов использования, администратор может работать с учётными записями (редактировать, добавлять, удалять, выводить данные клиентов в табличной форме), пароли учётных записей хранятся в зашифрованном виде.

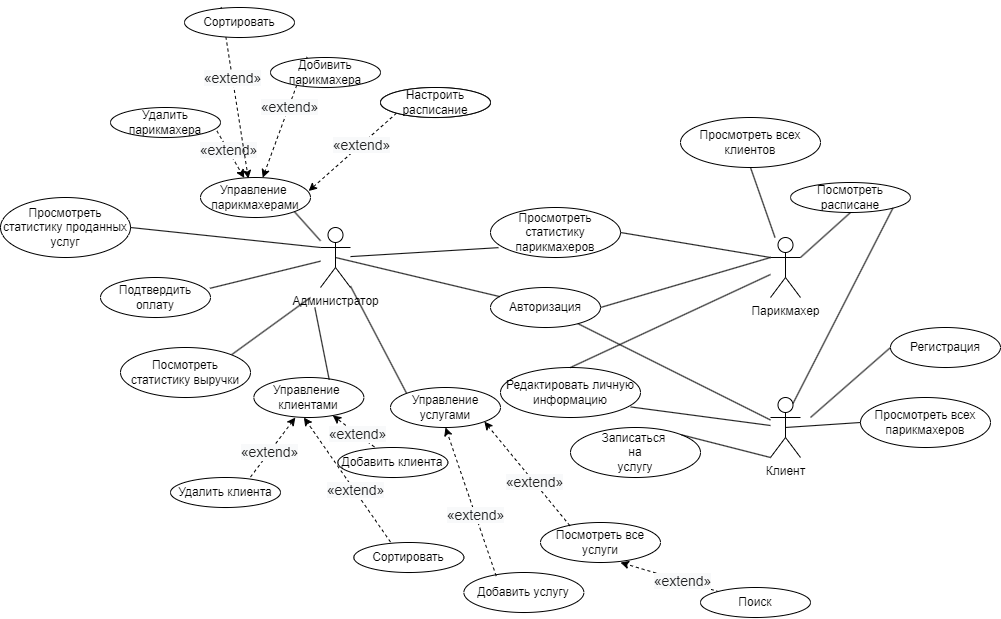


Рисунок 1.8 – Диаграмма вариантов использования

## Разработка информационной модели предметной области

При проектировании системы использовались следующие сущности:

* client;
* user;
* barber;
* admin;
* appointment;
* specialization;
* timetable.

Отображение модели приведено на рисунке 1.9.

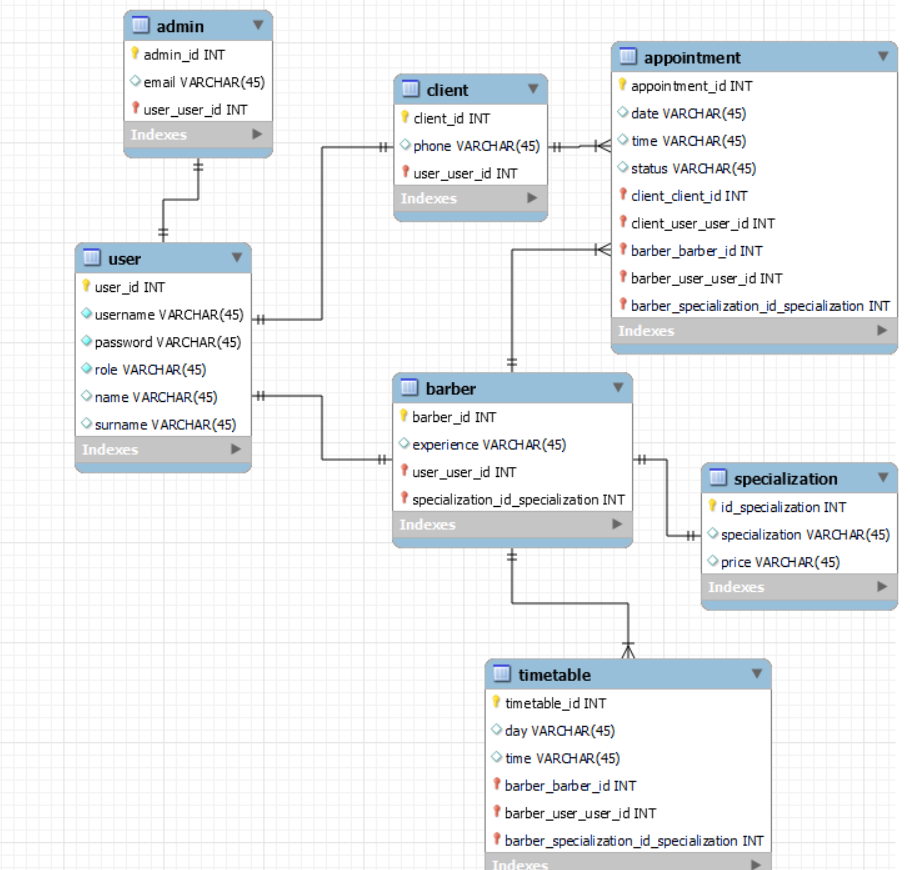


Рисунок 1.9 – Информационная модель системы

Ниже представлено подробное описание некоторых таблиц, входящих в модель.

Таблица User являющаяся родителем для таблиц Client, Barber и Admin имеет следующие поля:

* user\_id – уникальный идентификатор пользователя;
* username – логин пользователя;
* password – пароль пользователя;
* role – роль пользователя в системе;
* name – имя пользователя;
* surname – фамилия пользователя;

Таблица Client имеет следующие поля:

* client\_id – уникальный идентификатор клиента;
* phone – номер телефона клиента;
* user\_user\_id – мигрирующий ключ;

Таблица Barber имеет следующие поля:

* barber\_id – уникальный идентификатор парикмахера.
* experience – опыт работы парикмахера;
* user\_user\_id, specialization\_id\_specialization – мигрирующие ключи.

Таблица Admin имеет следующие поля:

* admin\_id – уникальный идентификатор администратора;
* email – почта администратора;
* user\_user\_id – мигрирующий ключ;

Таблица Timetable имеет следующие поля:

timetable\_id – уникальный идентификатор таблицы.

day – день;

time – время;

barber\_barber\_id, barber\_user\_user\_id, barber\_specialization\_id\_specialization – мигрирующие ключи.

*SQL*-скрипт для генерации базы данных приведен в приложении В.

## UML-модели представления программного средства и их описание.

**1.5.1** Диаграмма состояний показывает путь записи клиента на услугу. При нехватке информации система просит добавить недостающие поля формы. Она изображена на рисунке 1.10.

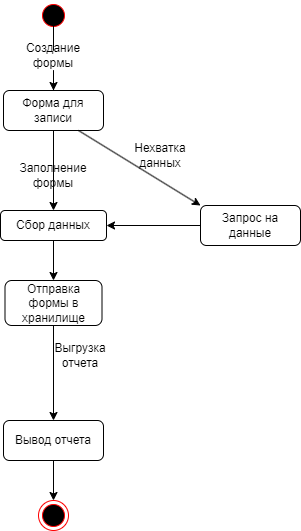


Рисунок 1.10 – Диаграмма состояний записи клиента

**1.5.2** Диаграмма последовательности (sequence diagram).

На рисунке 1.11 представлена диаграмма последовательности, на которой отображен процесс взаимодействия пользователя и сервера.

Основными элементами диаграммы последовательности являются: обозначения объектов (прямоугольники с названиями объектов); вертикальные «линии жизни», отображающие течение времени; прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»); стрелки, показывающие обмен сигналами или сообщениями между объектами [3].

Сначала пользователь отправляет логин и пароль, введенные данные обрабатываются на серверной стороне, где данные отправляются в базу данных и проверяются там, в случае если произошел сбой в проверке данных, ошибка отправляется сначала на сервер, затем клиенту на экран.

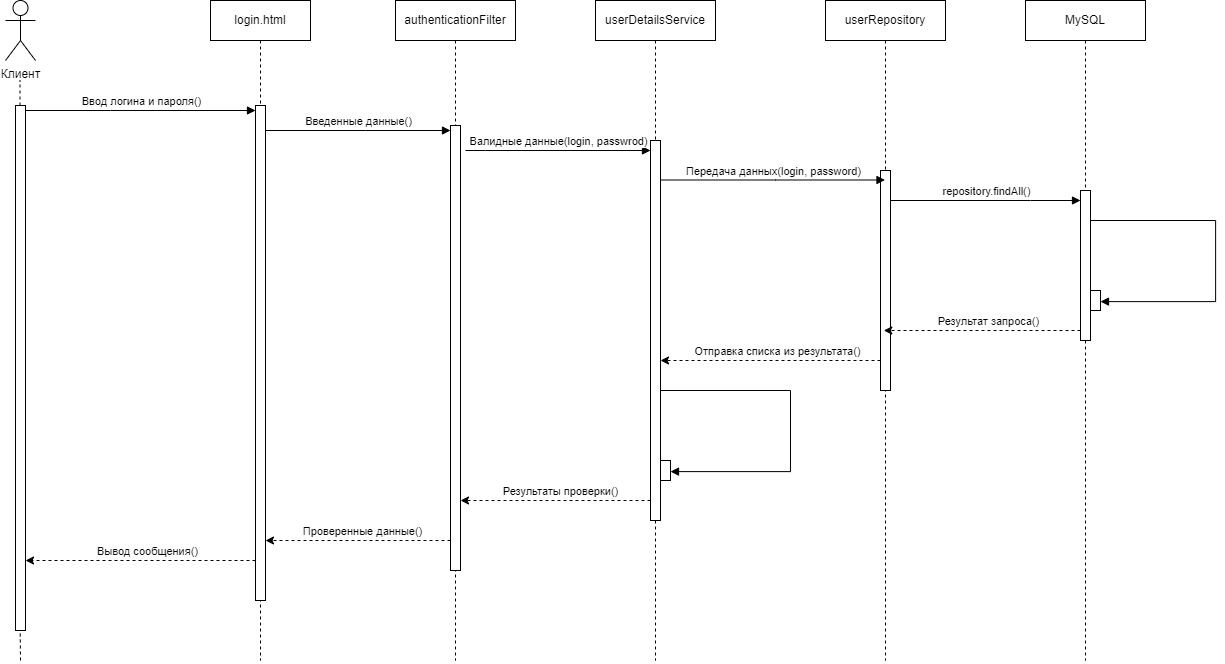


Рисунок 1.11 – Диаграмма последовательности

**1.5.3** Диаграмма развёртывания. С помощью диаграммы развертывания отобразим элементы и компоненты системы, существующие на этапе ее исполнения.

Диаграмму развертывания системы приведена на рисунке 1.12.

На данной диаграмме в качестве устройств выступают ПК клиента, сервер и база данных. В качестве компонентов выступают исполняемые файлы courseProject.war и браузер (Edge).

Клиент взаимодействует с сервером через HTTP запросы, сервер связывается с базой данные через JDBC драйвер. Файл courseProject.war использует для работы библиотеки, а сервер использует базу данных, для хранения данных.

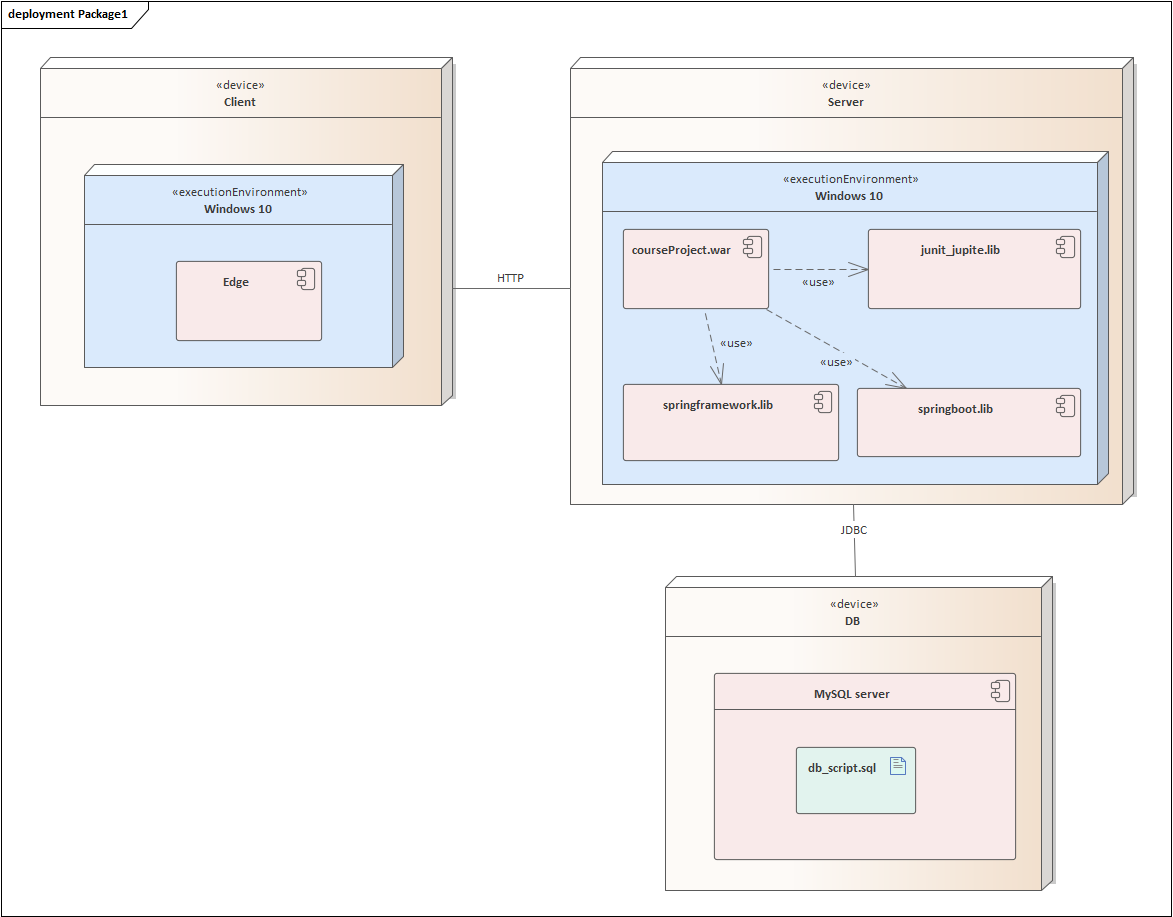


Рисунок 1.12 – Диаграмма развертывания

Таким образом, диаграмма развертывания предназначена для визуализации элементов и компонентов системы, существующих лишь на этапе ее исполнения (*runtime*), к которым относятся исполнимые файлы, динамические библиотеки, таблицы базы данных. Те компоненты, которые не используются на этапе исполнения (например, исходные тексты программ), на диаграмме не показываются.

Таким образом, в данной главе курсового проекта представлены моделирование предметной области и разработка требований к программному средству, анализ и формализация бизнес-процессов службы управления персоналом, разработаны спецификация функциональных требований, информационная модель программного средства, а также модели представления программного средства на основе языка *UML*.

# ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## Постановка задачи

Для разработки программного средства надо понять принцип работы любой парикмахерской.

Так, у любого клиента есть свои предпочтения по прическе, мастеру и времени для стрижки. Необходимо спроектировать базу данных, которая хранила бы всю необходимую информацию о клиенте, услугах, парикмахерах. Это ускорит поиск данных по определенному клиенту. Облегчит клиенту процесс записи к парикмахеру.

Для хранения информации будет использованa СУБД MySql.

Необходимо предоставить администратору функции просмотра статистики: у какого парикмахера побывало наибольшее число клиентов, какая услуга пользуется наибольшей популярностью. Для дальнейшего улучшения качества обслуживания.

Одной из главных задач программного средства является обеспечение возможности формирования расписания мастеров. Необходимо учесть занятость каждого мастера, так, если у одного мастера полностью занят день, на который хочет записаться клиент, необходимо предложить ему другого мастера, либо отменить запись. Чтобы это реализовать необходимо предоставить возможность оставления клиентом заявки на стрижку и последующую обработку заявки менеджером.

## Обоснование выбора компонентов и технологий для реализации программного средства

Для написания курсового проекта использовался Spring framework, Bootstrap framework.

Spring framework был выбран так как он обеспечивает решения многих задач, с которыми сталкиваются Java-разработчики и организации, которые хотят создать информационную систему, основанную на платформе Java. Из-за широкой функциональности трудно определить наиболее значимые структурные элементы, из которых он состоит. Spring не всецело связан с платформой Java Enterprise, несмотря на свою масштабную интеграцию с ней, что является важной причиной его популярности.

Spring, вероятно, наиболее известен как источник расширений (features), нужных для эффективной разработки сложных бизнес-приложений вне тяжеловесных программных моделей, которые исторически были доминирующими в промышленности. Ещё одно его достоинство в том, что он ввел ранее неиспользуемые функциональные возможности в сегодняшние господствующие методы разработки, даже вне платформы Java.

Этот фреймворк предлагает последовательную модель и делает её применимой к большинству типов приложений, которые уже созданы на основе платформы Java. Считается, что Spring реализует модель разработки, основанную на лучших стандартах индустрии, и делает её доступной во многих областях Java [10].

Bootstrap был выбран, потому что это открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для быстрой вёрстки адаптивных дизайнов сайтов и веб-приложений.

Основная область его применения – это фронтенд разработка сайтов и интерфейсов администрирующих панелей. Среди аналогичных систем фреймворк Bootstrap является самым популярным и простым в использовании.

## Архитектурные решения

Диаграммы классов используются при моделировании программных средств наиболее часто. На диаграммах классов показываются классы, интерфейсы и отношения между ними. В данном курсовом проекте были реализованы классы: *Appointment*, *Barber*, Specialization*, Timetable, User*. Все они используются для хранения информации о сущностях, присутствующих в системе. Класс *User* хранит в себе поле *email*, которое необходимо для связи с конкретным пользователем. Класс *Barber* хранит в себе поля *experience, specialization*. Поле *experience* хранит информацию об опыте работы конкретного парикмахера для того, чтобы клиент примерно понял, с кем он будет иметь дело. Поле *specialization* содержит в себе специализацию (услугу, которую парикмахер может предоставить).

Класс *EmailService* отвечает за коммуникацию с клиентом, а именно за отправку сообщений ему на почту. Он также является «Одиночкой», это необходимо для того, чтобы в программе был только один экземпляр класса, и чтобы можно было обращаться к нему из любого места программы.

Далее, на рисунке 2.1 будет показана диаграмма классов, пакета *entities.*

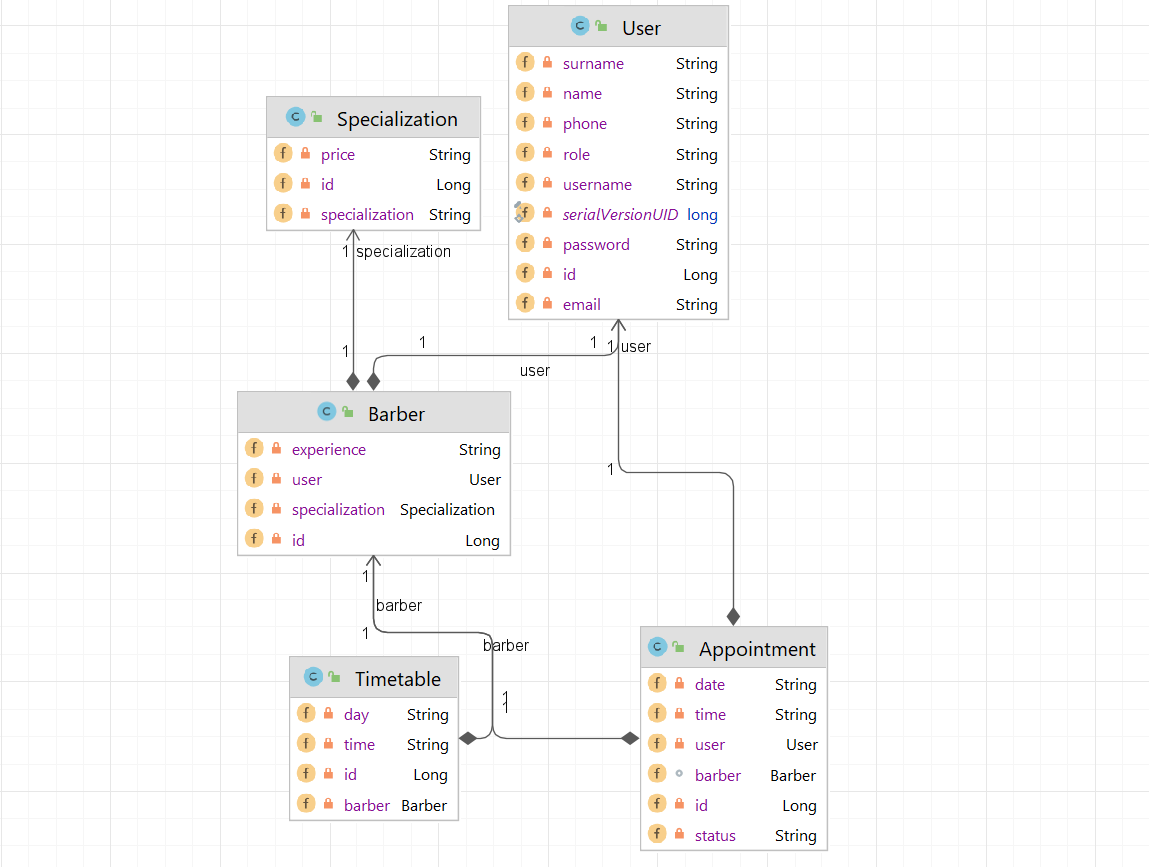


Рисунок 2.1 – Диаграмма классов пакета *entities*

Одиночка – это порождающий паттерн проектирования, который гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа.[4] В качестве класса, реализующего данный паттерн проектирования, выступает класс *EmailService*, схему которого можно увидеть на рисунке 2.2. У данного класса есть статическое поле instance, представляющее собой глобальную точку доступа к единственному экземпляру класса *EmailService*.

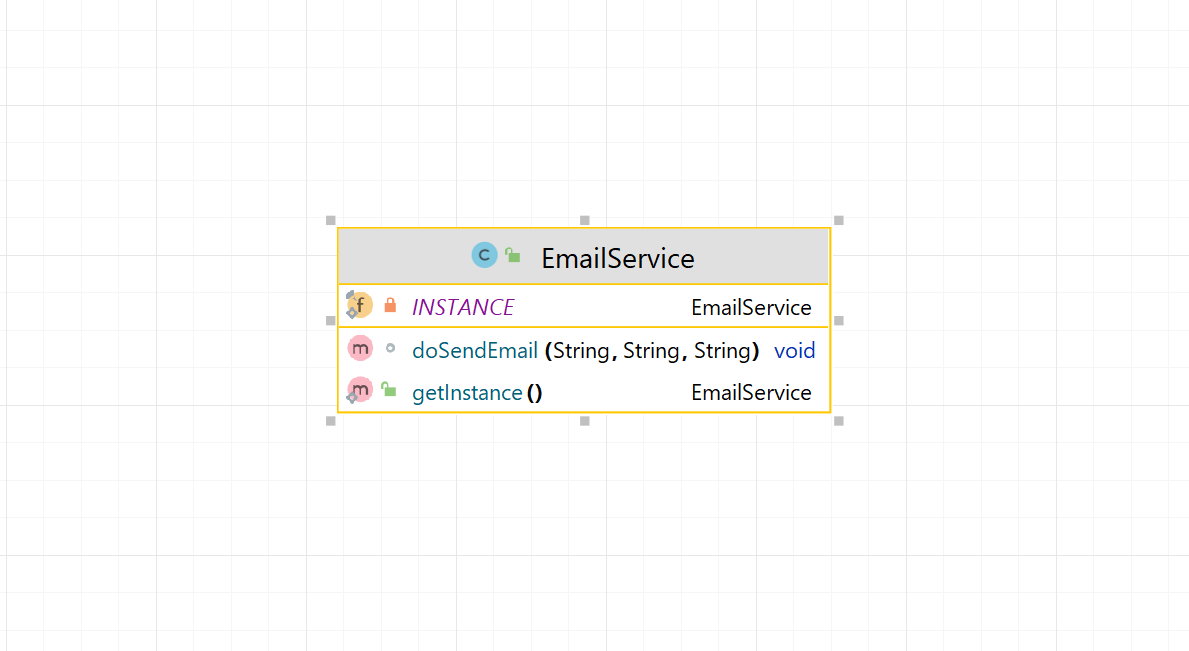


Рисунок 2.2 – Класс-одиночка *EmailService*

Также реализован паттерн «Строитель» посредством аннотации @*Builder* проекта *Lombok.* Аннотируя класс при помощи *@Builder*, *Lombok* выдает класс, реализующий вышеупомянутый паттерн «строитель»*.*

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

Рассмотрим схему алгоритма работы программного средства.

При запуске программы пользователю предлагается пройти авторизации, после успешного прохождения которой, в зависимости от уровня доступа пользователю предоставляется разный набор функций.

Если авторизация не прошла успешно, то на экран выводится соответствующее сообщение и можно попытаться авторизоваться снова.

Так, если был выполнен вход под парикмахером, то ему предоставляется возможность редактирования личных данных, просмотр своего расписания, просмотр клиентов, просмотр статистики парикмахеров.

У администратора функционал более широкий. Помимо стандартных операций он может добавлять, удалять клиентов и парикмахеров, просматривать статистику. Добавлять новые услуги и удалять те, которые парикмахерская уже не предоставляет.

Клиент помимо просмотра своего расписания, может записаться на услугу, просмотреть доступные услуги, редактировать личную информацию.

На рисунке 2.4 предоставлена схема алгоритма обобщенной работы программы. На рисунке 2.5 предоставлен алгоритм записи клиента к парикмахеру.

После просмотра доступных услуг клиент может записаться на понравившуюся ему услугу. Для этого ему нужно будет заполнить форму, в которой он должен выбрать парикмахера, услугу, дату и время ее проведения.

Введенные данные отправляются на сервер, который уже обрабатывает полученную информацию, формирует запись, обозначает выбранное клиентом время как занятое и уже сформированную запись добавляет в таблицу встреч базы данных.

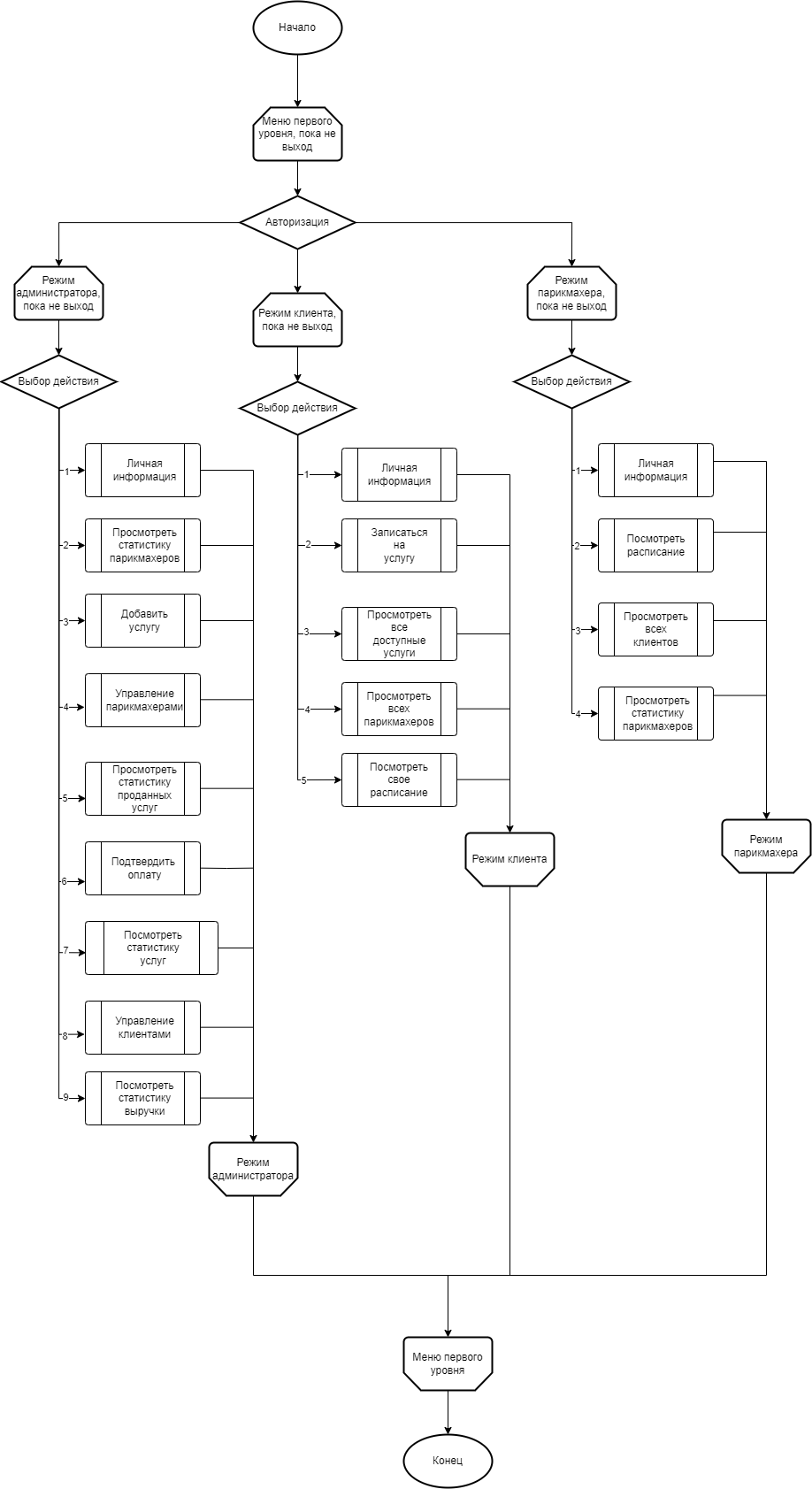


Рисунок 2.4 – Схема алгоритма

На рисунке 2.5 предоставлен алгоритм записи клиента к парикмахеру.

После просмотра доступных услуг клиент может записаться на понравившуюся ему услугу. Для этого ему нужно будет заполнить форму, в которой он должен выбрать парикмахера, услугу, дату и время ее проведения.

Введенные данные отправляются на сервер, который уже обрабатывает полученную информацию, формирует запись, обозначает выбранное клиентом время как занятое и уже сформированную запись добавляет в таблицу встреч базы данных.

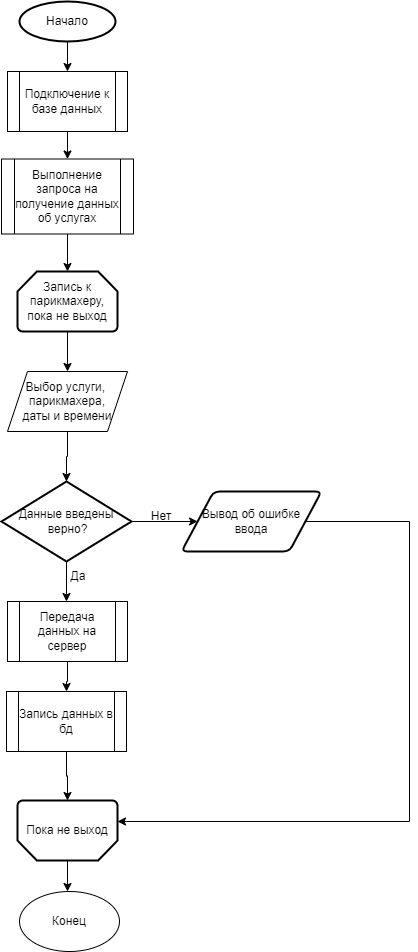


Рисунок 2.5 – Схема-алгоритма записи к парикмахеру

На рисунке 2.6 предоставлен алгоритм регистрации нового клиента. После ввода логина и пароля логин проверяется, находится ли он в пользовании после чего клиенту поступает информация: прошла ли регистрация успешно, либо регистрация закончилась ошибкой.

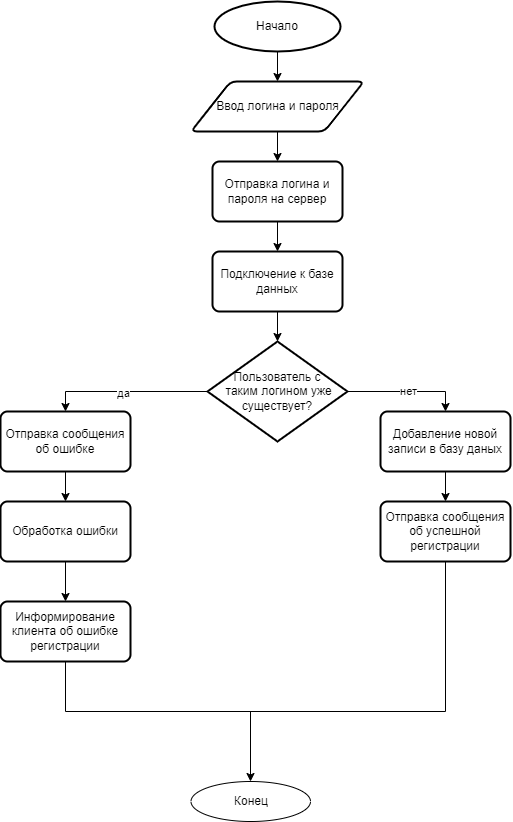


Рисунок 2.6 – Схема-алгоритма регистрации

## Проектирование пользовательского интерфейса

Интерфейс пользователя –  [интерфейс](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81), обеспечивающий передачу информации между [пользователем](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C)-человеком и [программно](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)-[аппаратными](https://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) компонентами [компьютерной системы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) [5].

Для проектирования пользовательского интерфейса использовались такие технологии как: шаблонизатор Thymeleaf, язык программирования Javascript, язык описания внешнего вида страницы CSS, Bootstrap framework.

Thymeleaf - это механизм шаблонов для рендеринга содержимого XML / XHTML / HTML5. Подобно JSP, FreeMaker и т. д., он также может быть легко интегрирован с веб-фреймворками, такими как Spring MVC, в качестве механизма шаблонов для веб-приложений.

CSS используется создателями веб-страниц для задания цветов, шрифтов, стилей, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS является ограждение и отделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

JavaScript — мультипарадигменный язык программирования. JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам (действия при клике на объект, наведении, также используется для динамического обновления данных на странице и т.п.).

# ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

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

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

Во-первых, была протестирована функция безопасности. А именно: был проведен тест, который убеждает в том, что мы не может зайти на какую-либо страницу нашего приложение, предварительно не авторизовавшись. Тест и результаты показаны на рисунке 3.1.

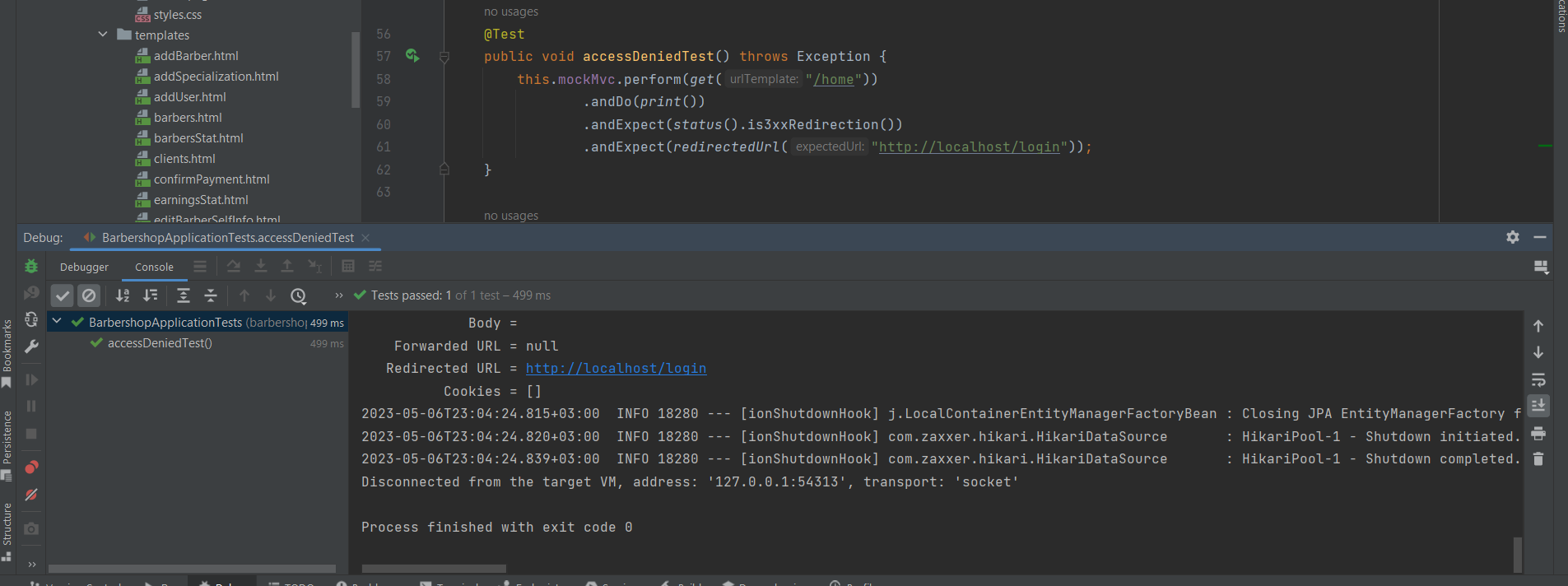


Рисунок 3.1 – Результаты тестирования авторизации

Во-вторых, была протестирована функция авторизации с вводом верных данных. Результат проведения теста показан на рисунке 3.2.

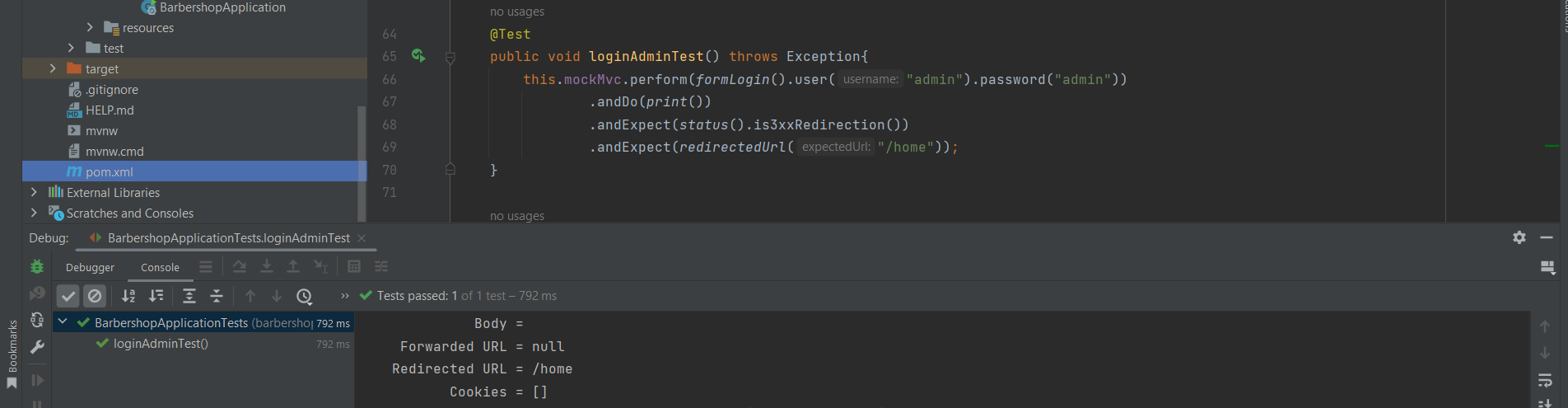


Рисунок 3.2 – Результаты тестирования авторизации

Далее была протестирована функция регистрации. На рисунке 3.3 показан результат тестирования регистрации с вводом имени пользователя, которое уже существует.

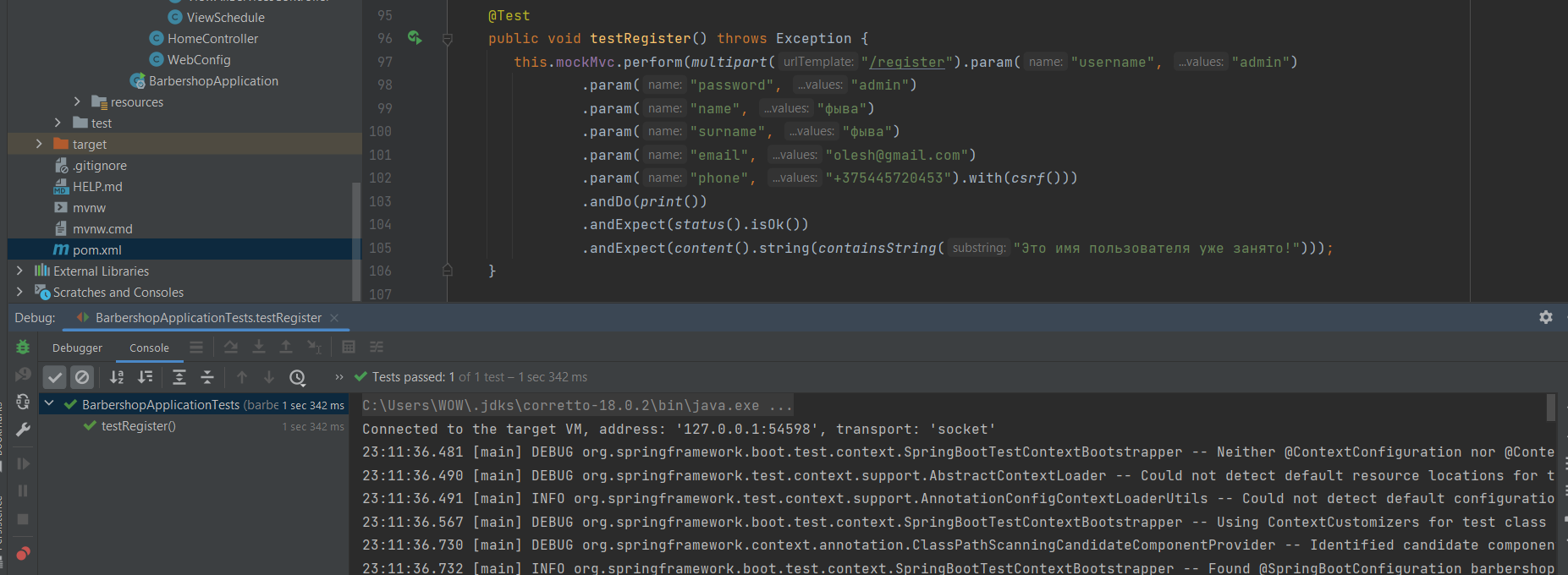


Рисунок 3.3 – Результаты тестирования регистрации

На рисунке 3.4 показан результат тестирования регистрации без ввода номера телефона.

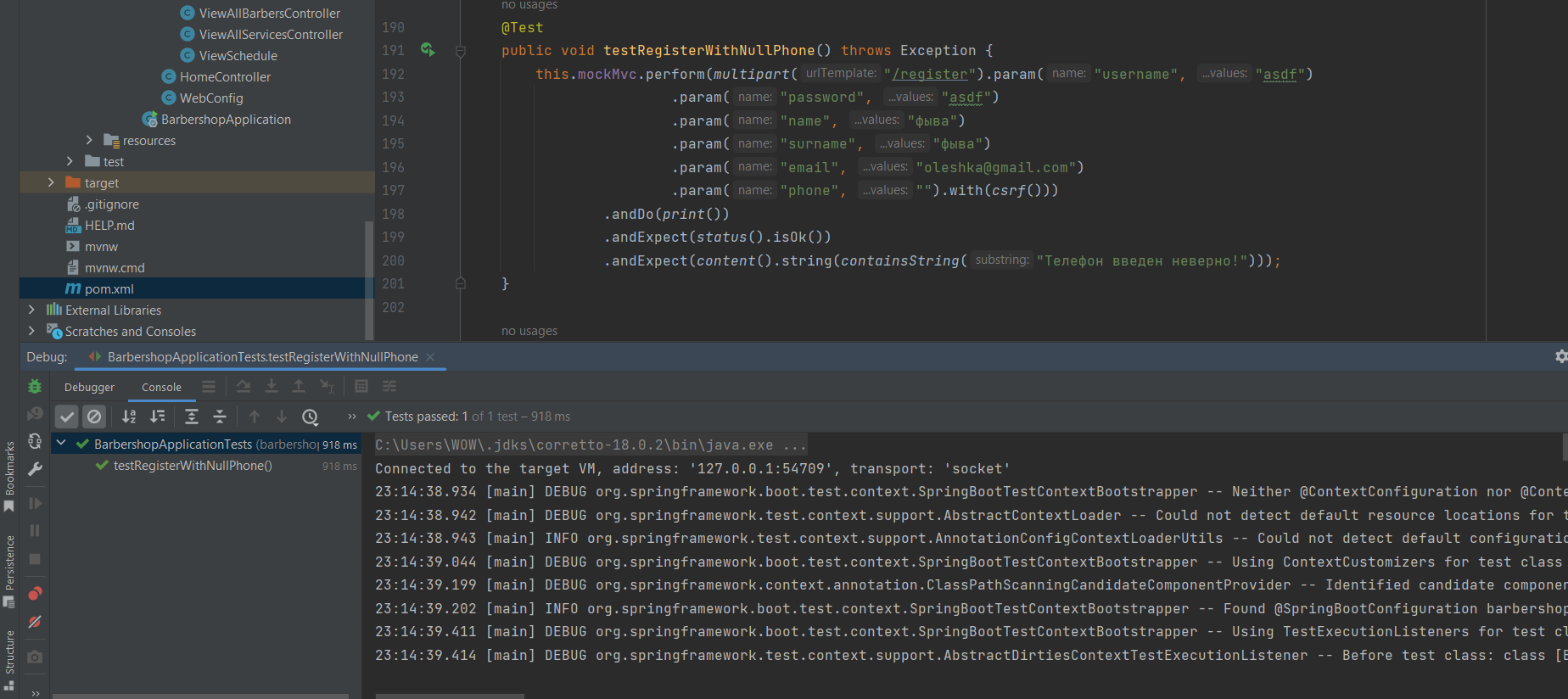


Рисунок 3.4 – Результаты тестирования регистрации без номера телефона

Далее был протестирован запрос на получение страницы по адресу «*localhost:8080»*. Результат продемонстрирован на рисунке 3.5.

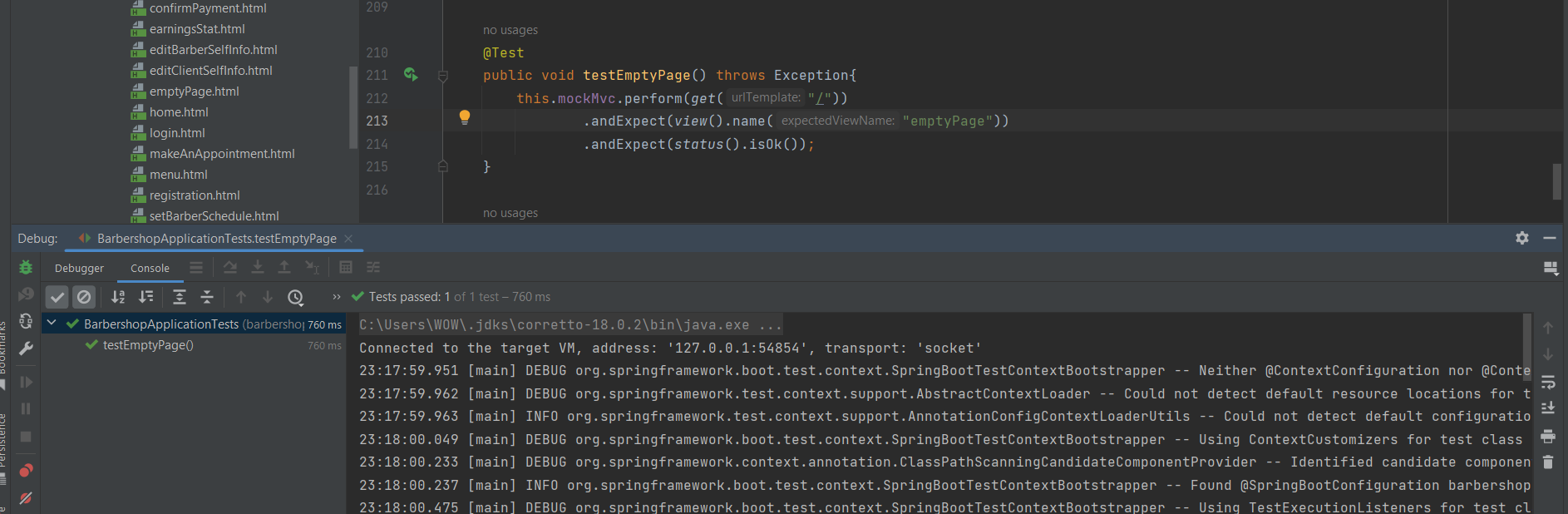


Рисунок 3.5 – Результаты тестирования получения страницы

Таким образом, по результатам тестирования приложения можно сказать, что оно ведет себя адекватно.

# РУКОВОДСТВО ПО РАЗВЕРТЫВНИЮ И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

## Руководство по установке (развертыванию) и использованию программного средства

Написанный курсовой проект был упакован в war-архив barbershop.war. Для запуска его необходимо разместить на сервере и выбрать в качестве запускаемого.

## Руководство пользователя

После запуска приложения появится страница авторизации, в котором будет предложено войти или зарегистрироваться. На рисунке 4.1 показана страница авторизации.

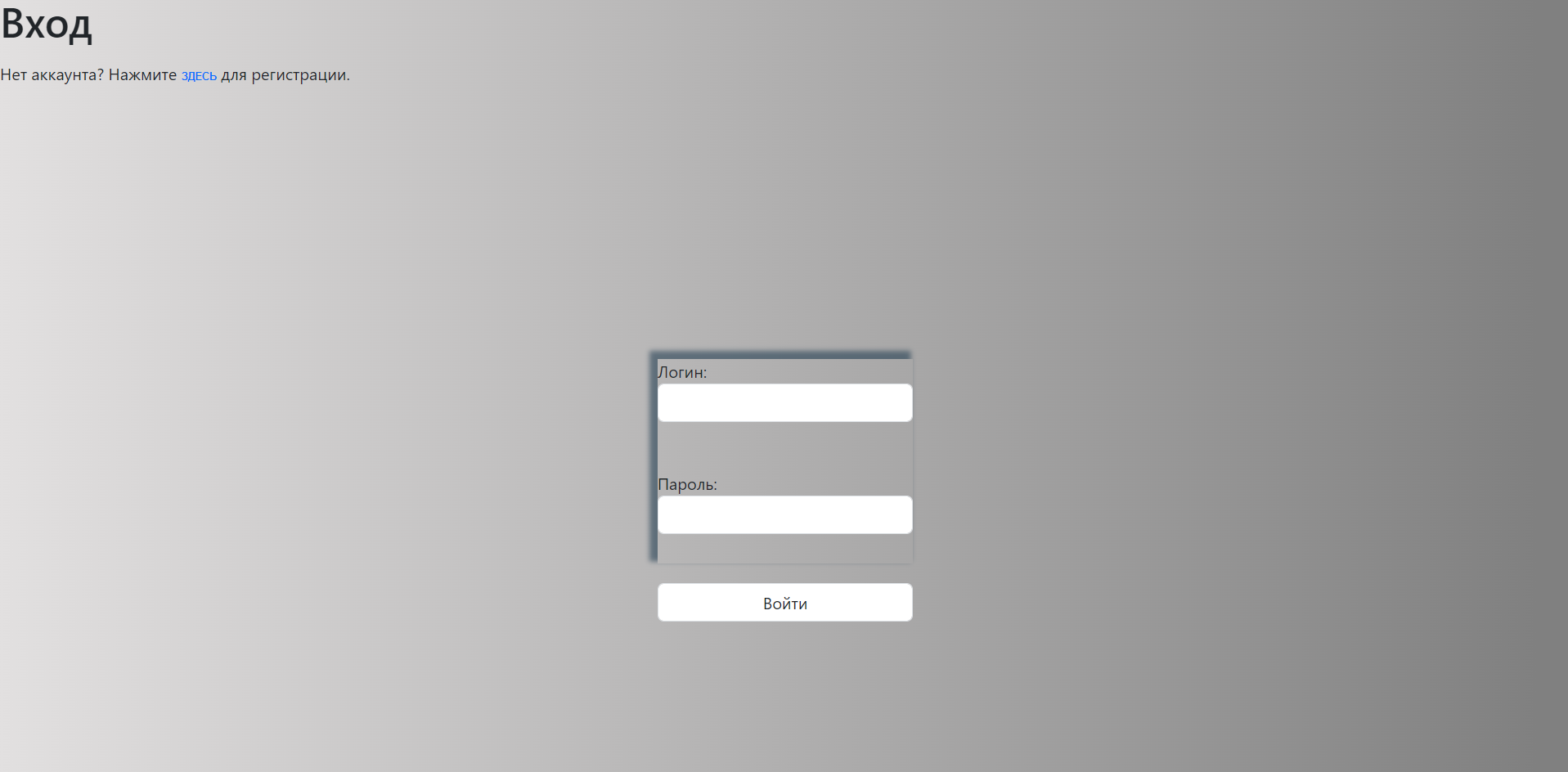


Рисунок 4.1 – Страница входа

В случае, если у пользователя нет аккаунта, то он может зарегистрироваться, пройдя по ссылке, расположенной на странице входа. На рисунке 4.2 показана страница регистрации.

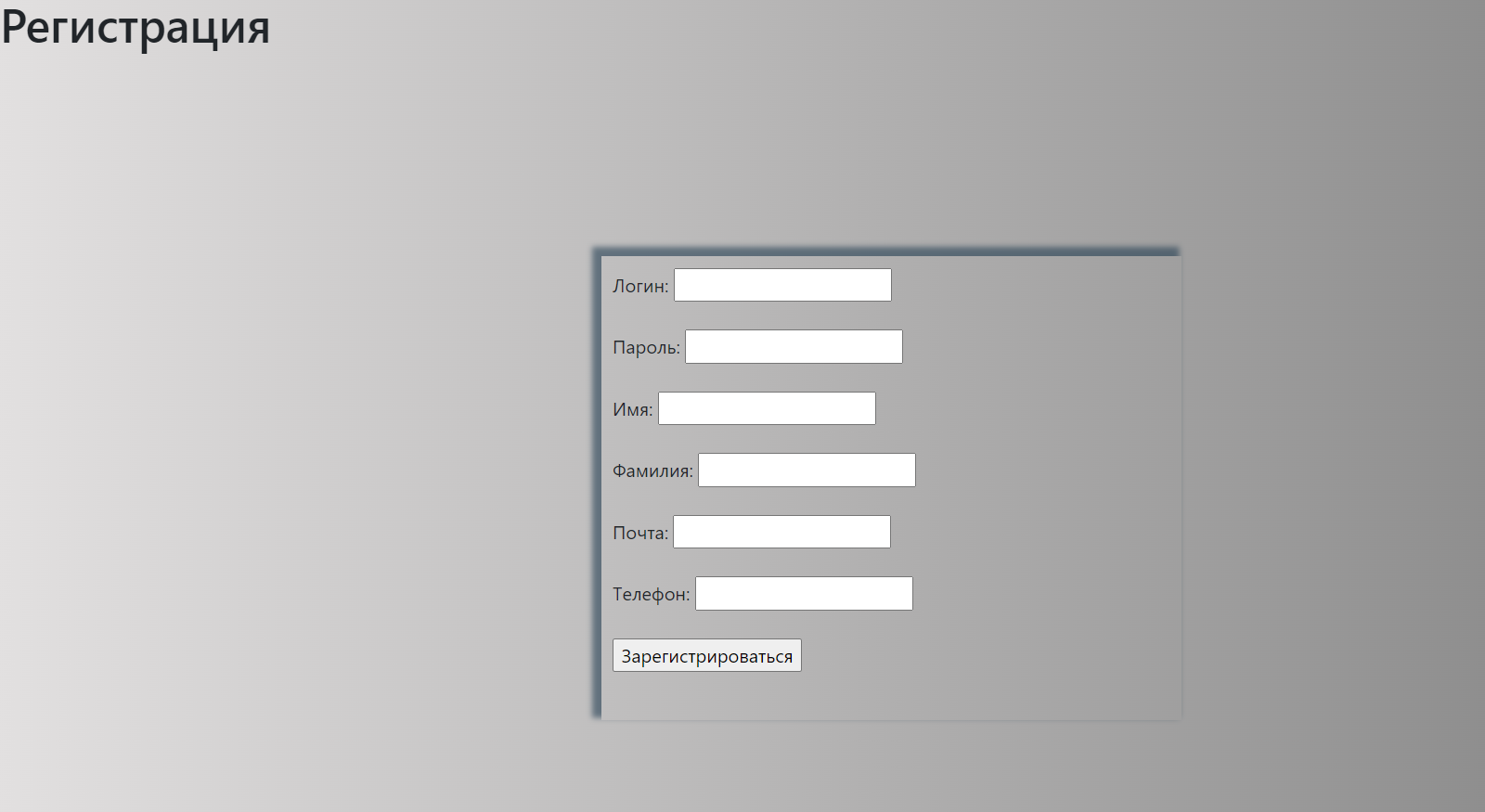


Рисунок 4.2 – Страница регистрации

После успешного входа в зависимости от роли пользователя будет открыто либо окно администратора, либо клиента, либо парикмахера. На рисунке 4.3 показана страница администратора.



Рисунок 4.3 – Страница администратора

На рисунке 4.4 будет показан страница клиента.



Рисунок 4.4 – Страница клиента

На рисунке 4.5 показана страница парикмахера.



Рисунок 4.5 – Страница парикмахера

Далее, для более наглядной демонстрации работы проекта будет предоставлена работа администратора.

При выборе функции управления клиентами, администратору предоставляется страница, на которой показаны все зарегистрированные клиенты, он может либо сразу удалить клиента, либо добавить нового. На рисунке 4.6 показана страница управления клиентами.

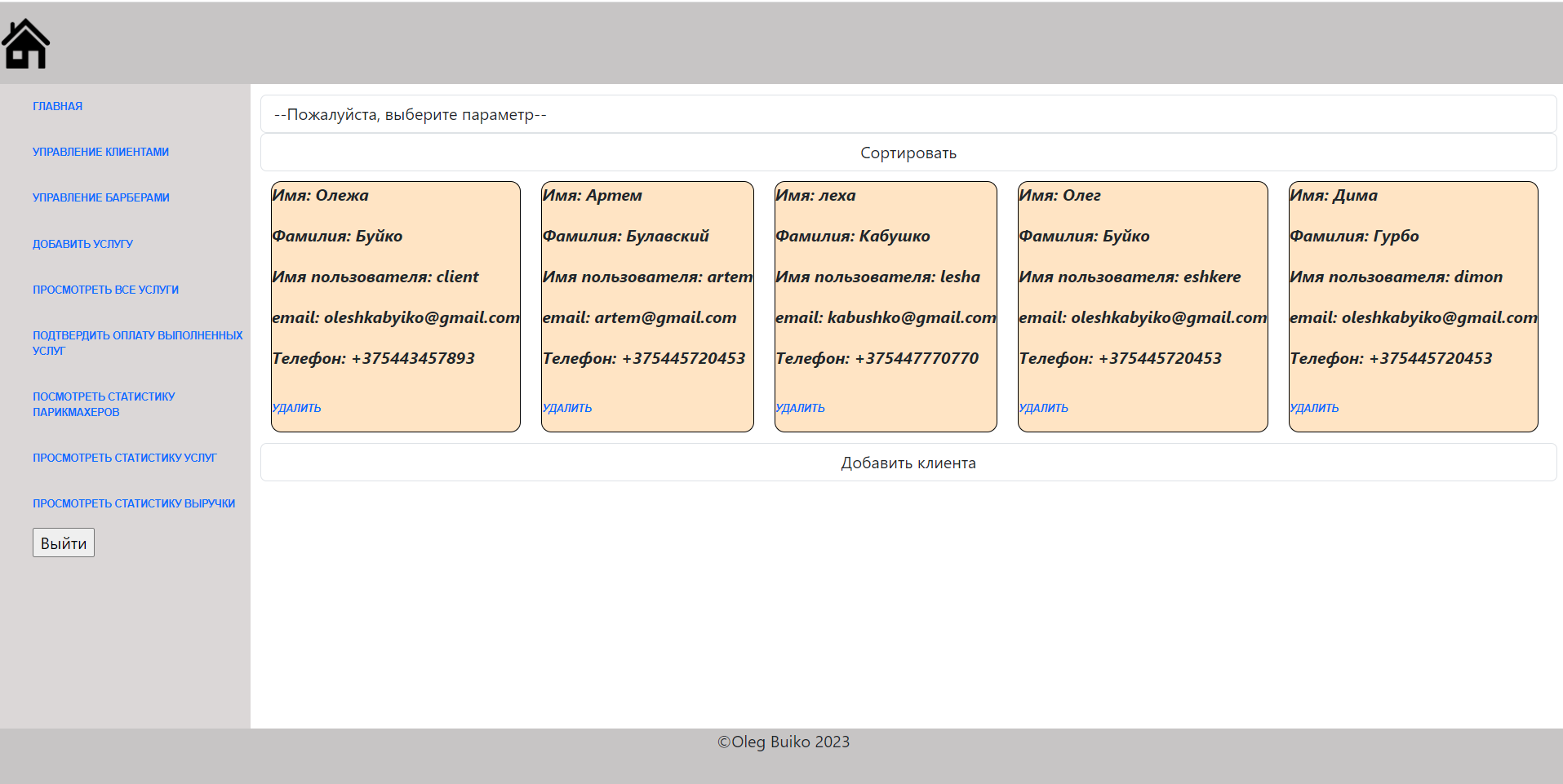


Рисунок 4.6 – Страница управления клиентами

На рисунке 4.7 показана страница для добавления нового клиента.



Рисунок 4.7 – Страница добавления клиента

Для управления парикмахерами администратору необходимо выбрать ссылку «Управление парикмахерами», далее, администратору необходимо выбрать действие: добавить парикмахера, удалить парикмахера либо настроить расписание парикмахера. На рисунке 4.8 представлено окно добавления.



Рисунок 4.8 – Окно добавления парикмахера

На рисунке 4.10 показана страница, на которой расположены функции удаления, добавления, сортировки и редактирования расписания конкретного парикмахера.

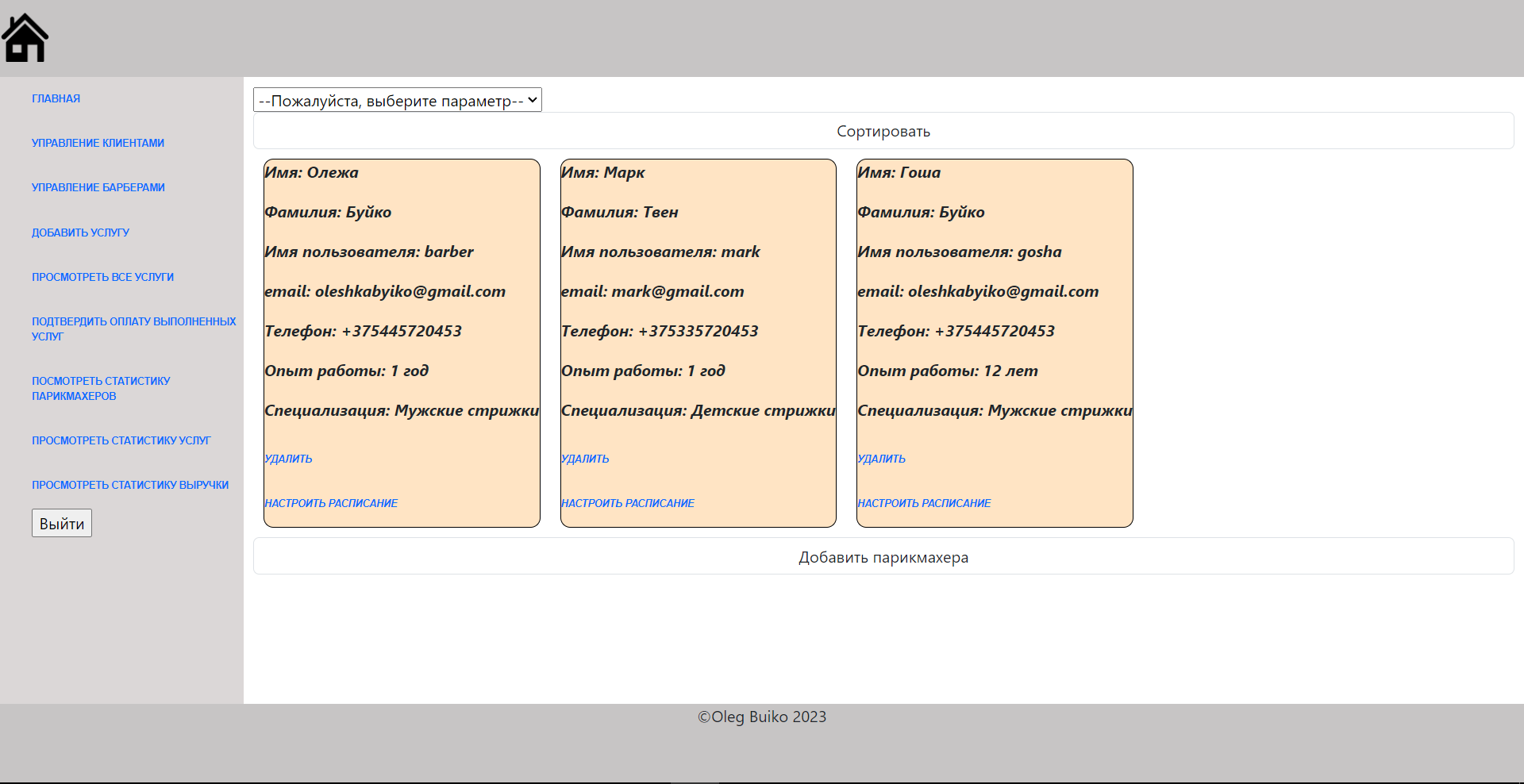


Рисунок 4.10 – Страница управления парикмахерами

На рисунке 4.11 показана страница редактирования расписания конкретного парикмахера.



Рисунок 4.11 – Страница редактирования расписания парикмахера

На рисунке 4.12 будет показана страница управления парикмахерами после сортировки парикмахеров по имени пользователя.

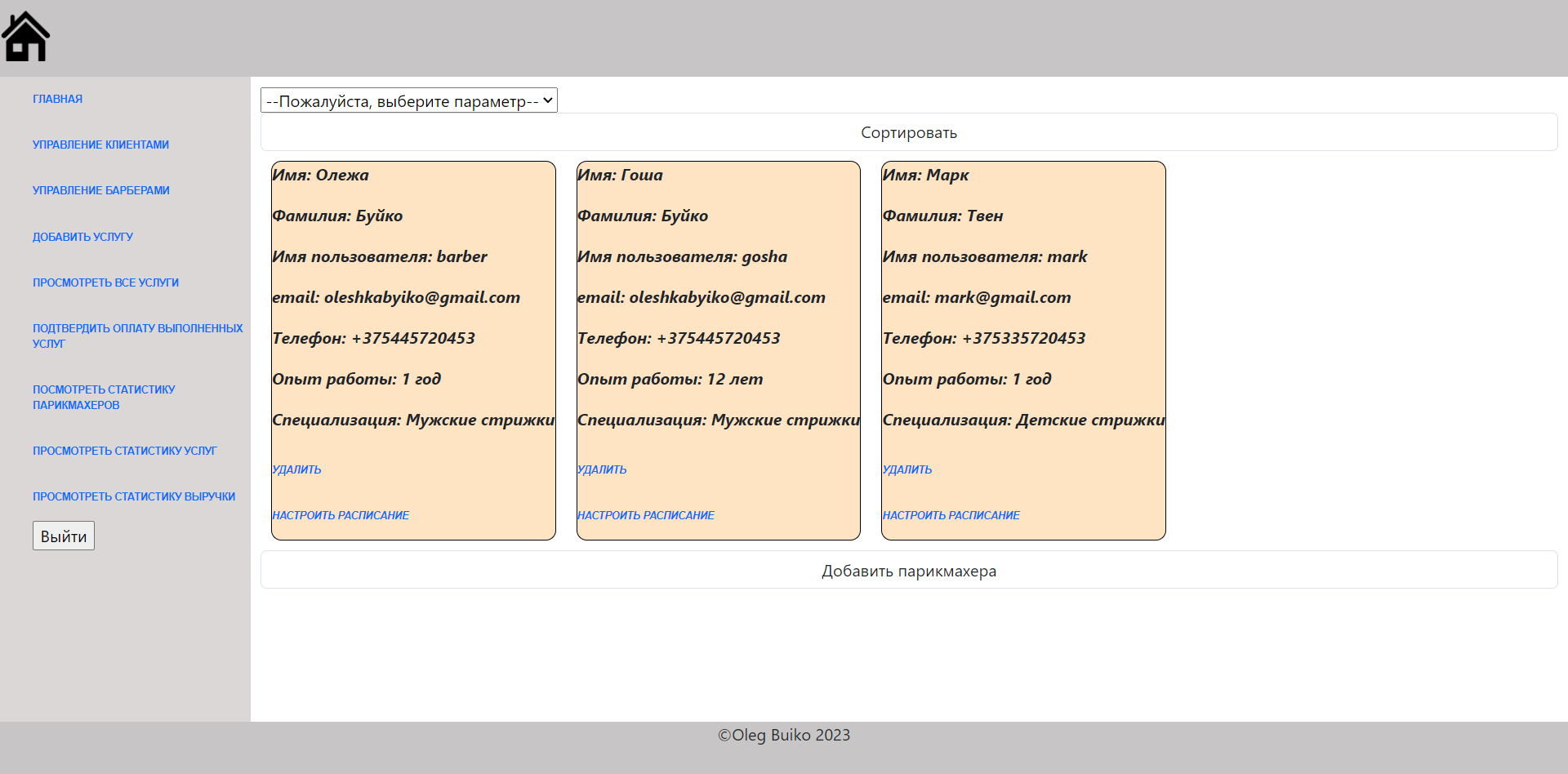


Рисунок 4.12 – Страница парикмахеров после сортировки

Администратор также может работать с услугами: добавит новую услугу, либо удалить уже имеющуюся. На рисунках 4.13 показана страница добавления новой услуги.

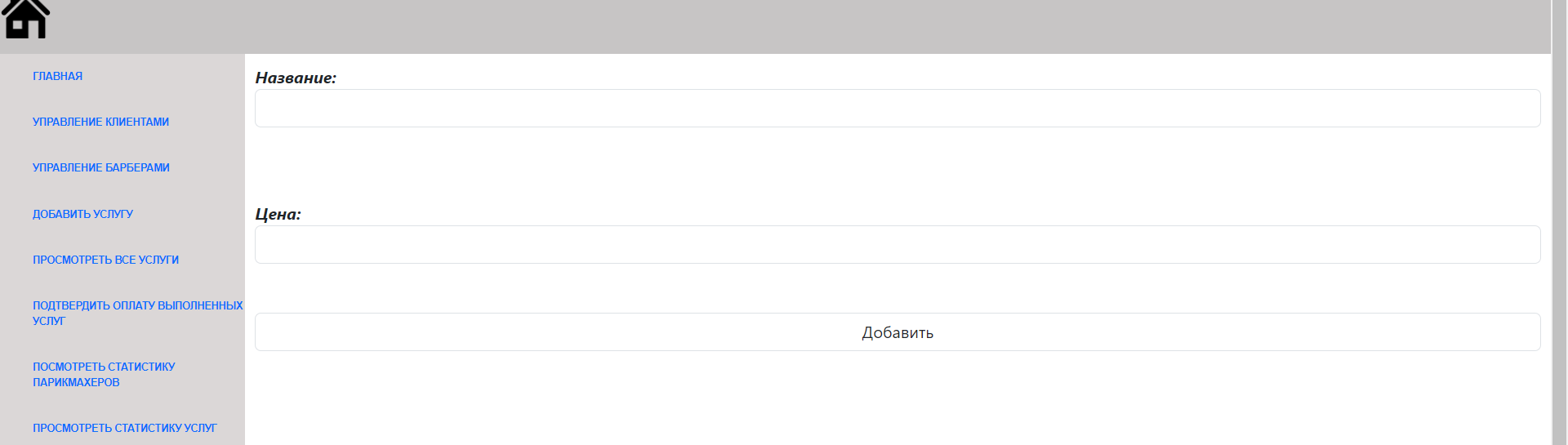


Рисунок 4.13 – Страница добавления услуги

На рисунке 4.14 показана страница, где расположены такие функции, как поиск услуги по названию, удаление услуги.

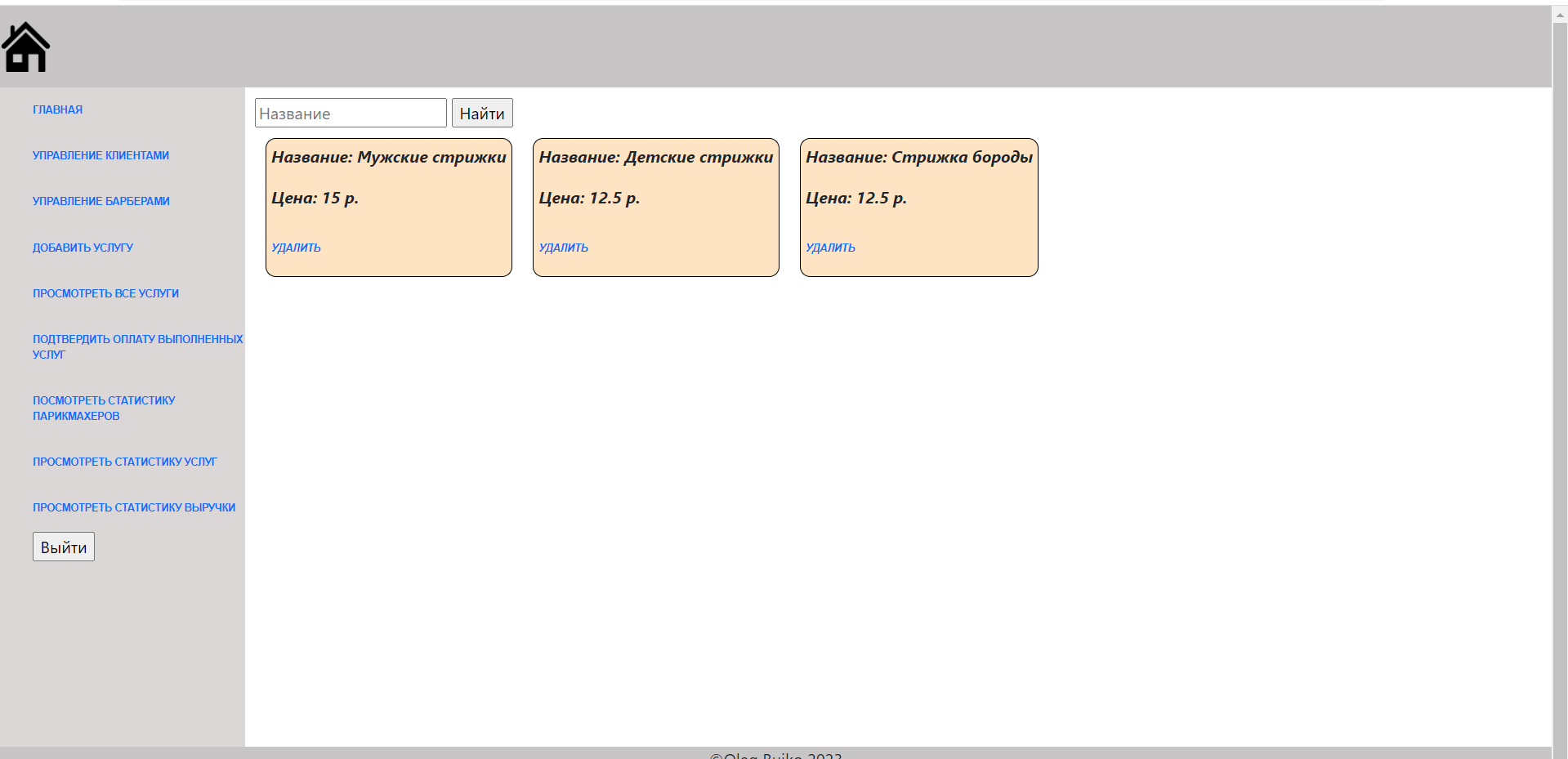


Рисунок 4.14 ­– Страница с услугами

На рисунке 4.15 будет показана страница с услугами после поиска конкретной услуги по ее названию.

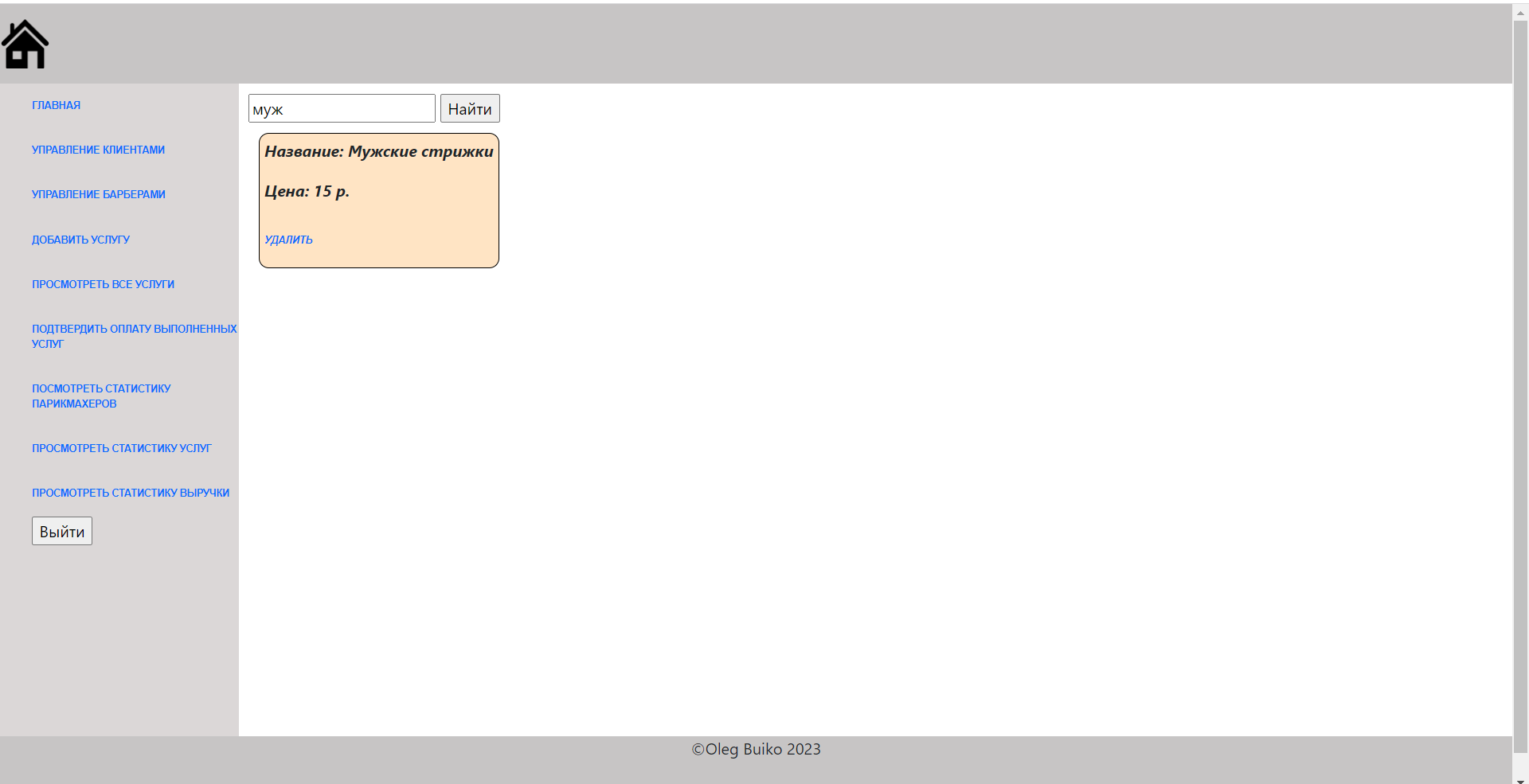


Рисунок 4.15 ­– Страница с услугами после поиска

Также администратор подтверждает оплату услуг, которые уже были предоставлены парикмахером. Администратор может просмотреть все услуги, которые уже были выполнены и пометить их как оплаченные. Страница подтверждения оплаты предоставлена на рисунке 4.16.

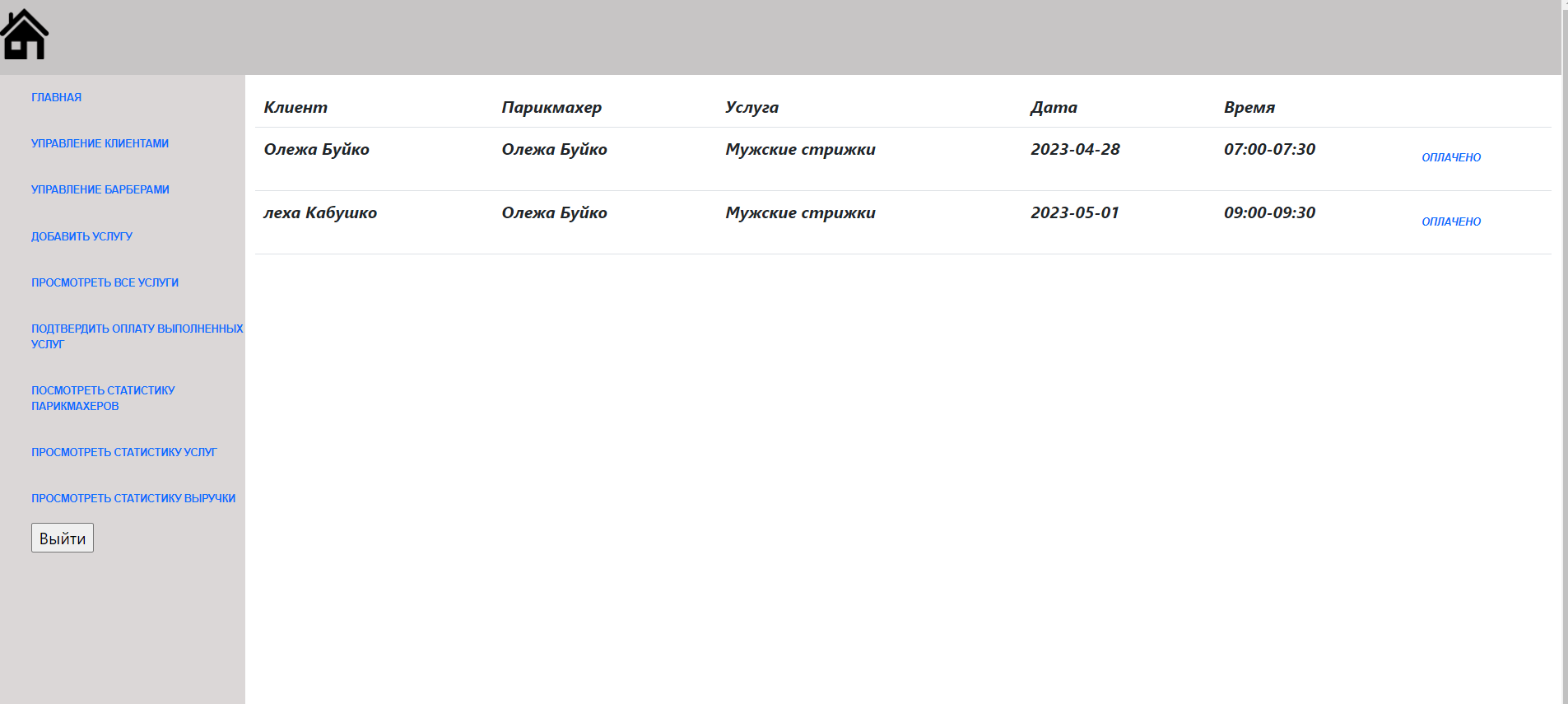


Рисунок 4.16 – Страница подтверждения оплаты

Также администратору доступен просмотр информации в графическом виде. В программном средстве предусмотрено отображение графиков о статистике парикмахеров, статистике предоставляемых услуг и выручке.

Пример графика посещаемости парикмахеров показан на рисунке 4.17.

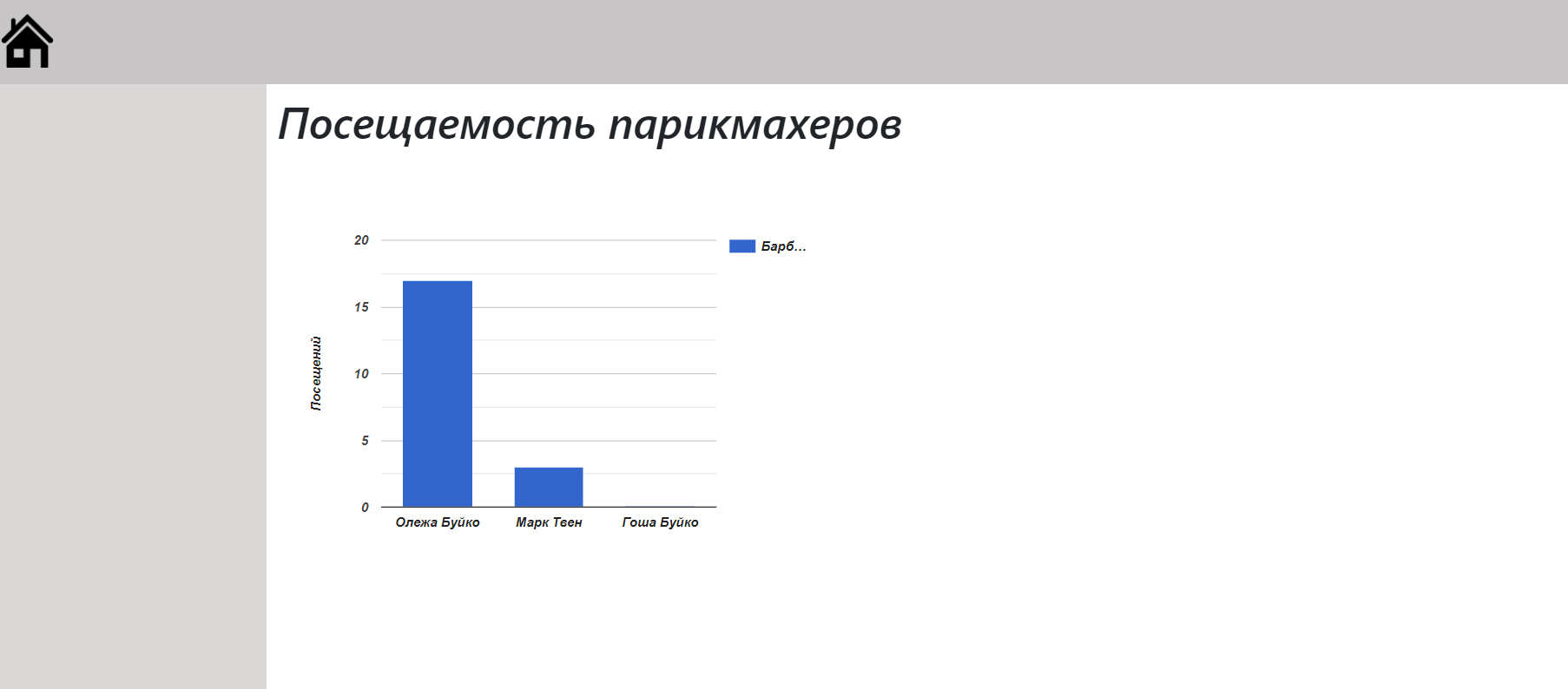


Рисунок 4.17 – Статистика парикмахеров

Пример графика популярности услуг предоставлен на рисунке 4.18.

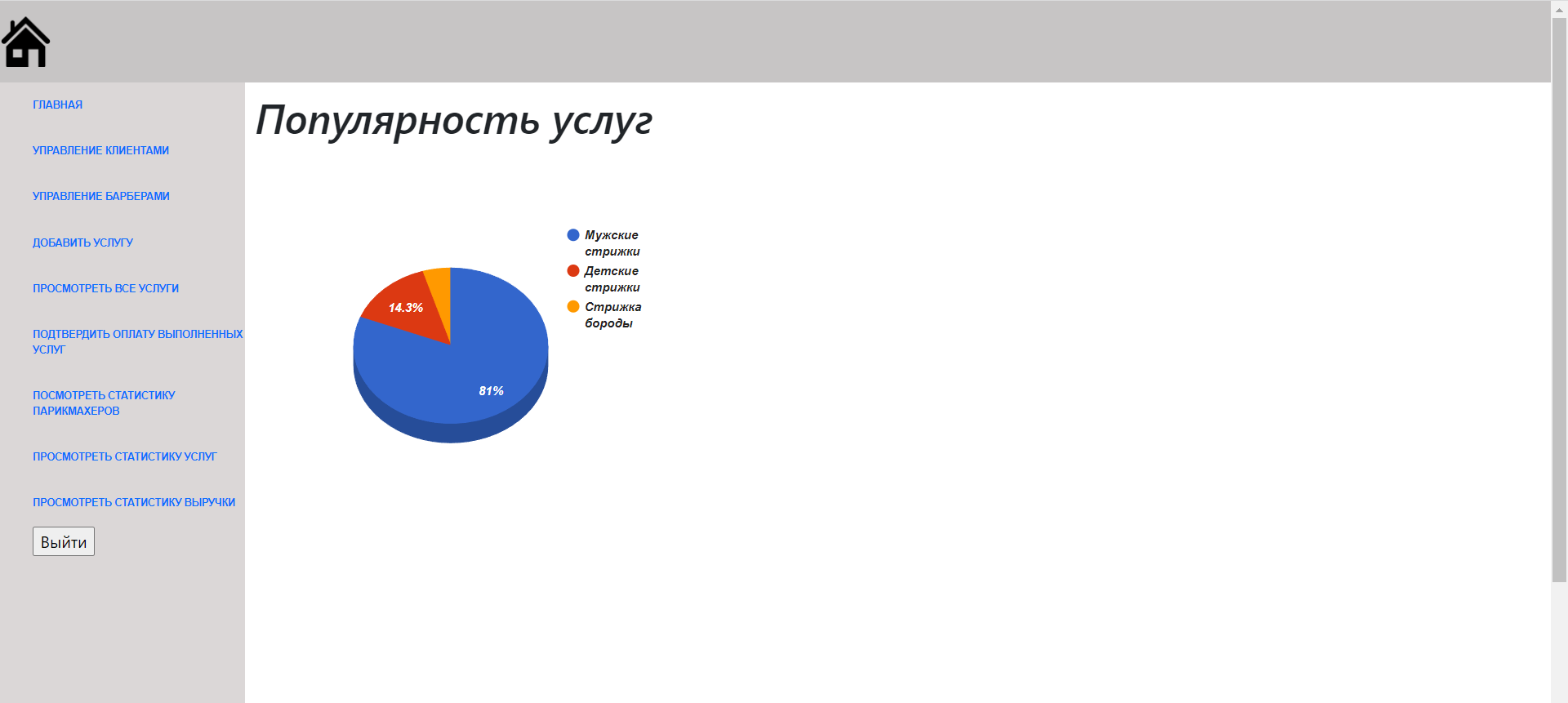


Рисунок 4.18 – Статистика предоставляемых услуг

На рисунке 4.19 показан пример графика, на котором изображена прибыль по месяцам.

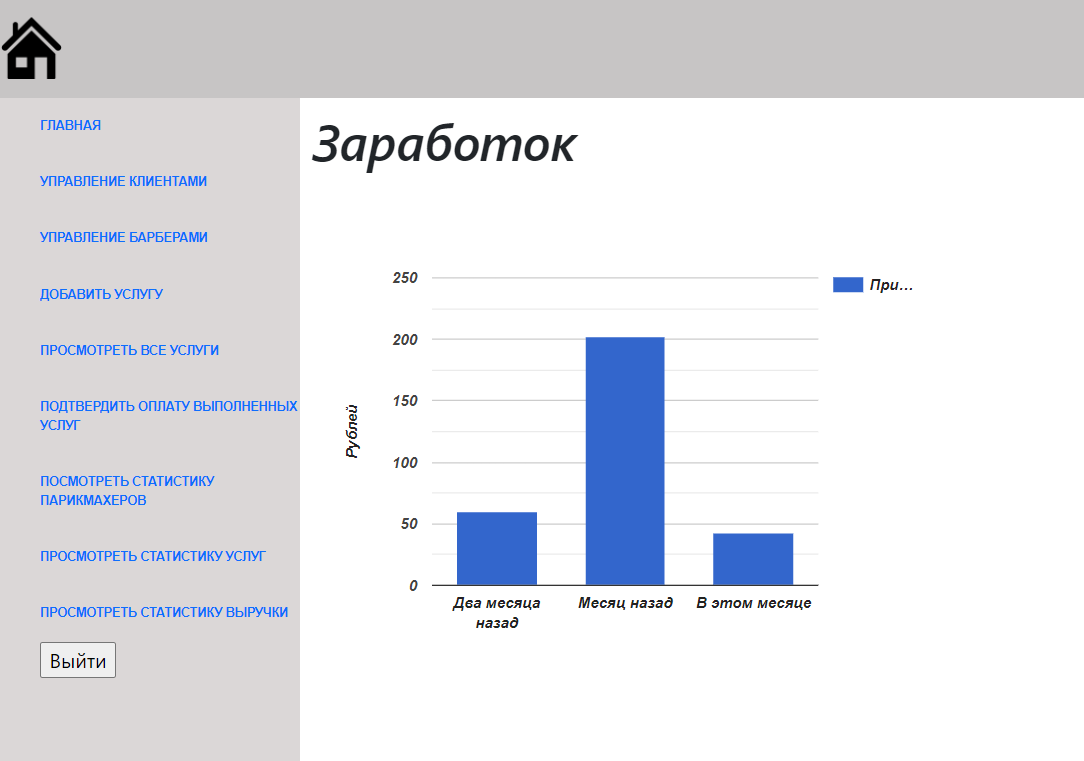


Рисунок 4.29 – Статистика выручки

При входе в программу как клиент, есть возможность записаться на услугу. Прежде чем записаться клиент должен выбрать услугу, парикмахера, дату и время. На рисунке 4.20 предоставлена страница клиента для записи на услугу.

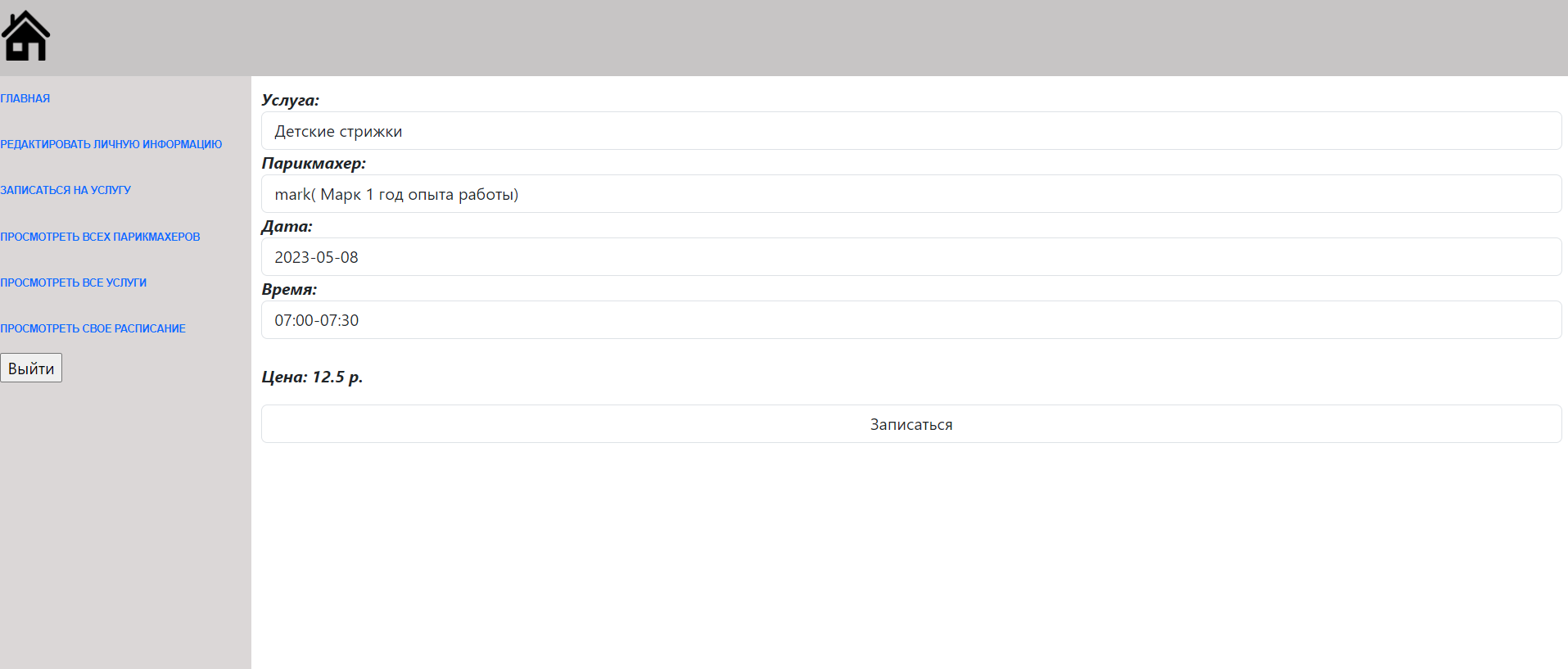


Рисунок 4.01 – Страница записи клиента на услугу

Также клиент может просмотреть свое расписание походов в парикмахерскую. На рисунке 4.21 предоставлена страница расписания одного из клиентов.

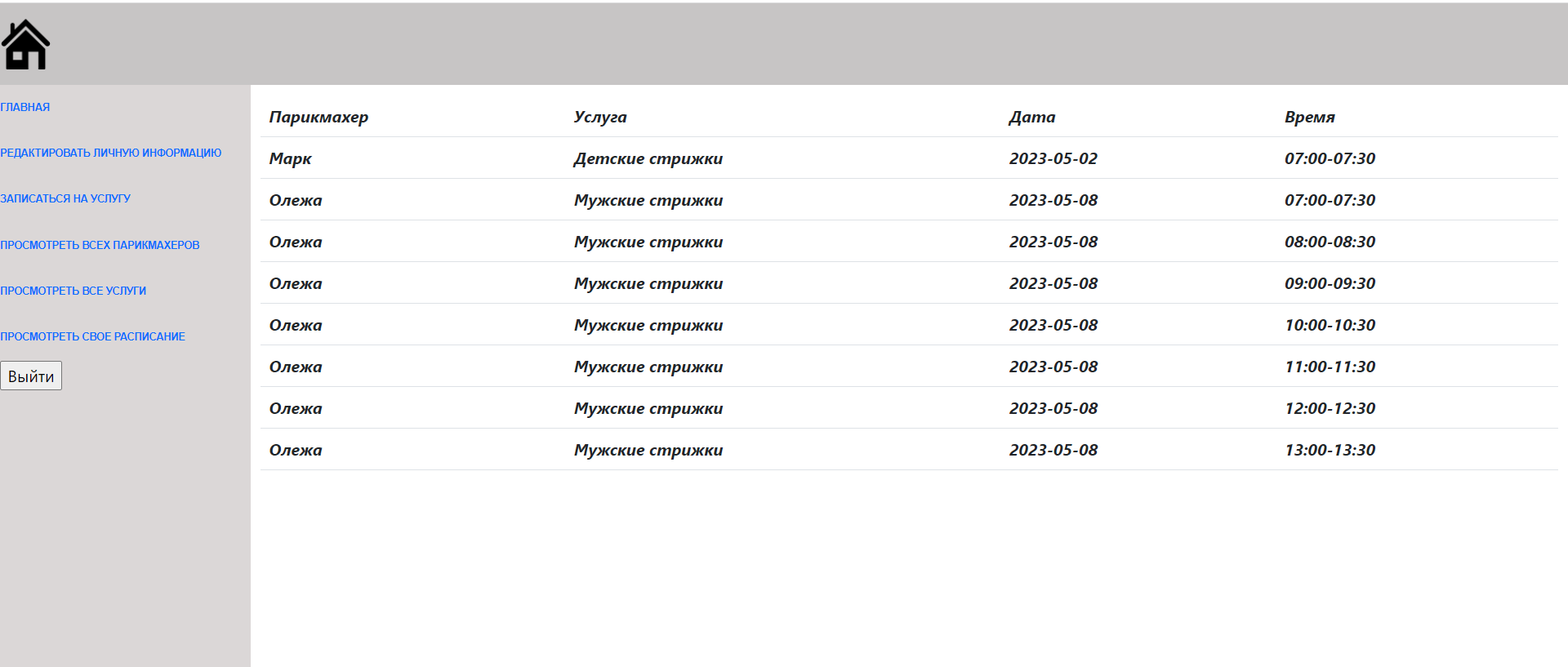


Рисунок 4.21 – Страница расписания

Таким образом, выполнены проектирование и разработка программного средства. Выполнена постановка задачи и определены основные методы ее решения; в ходе объектного моделирования системы построен ряд *UML*-диаграмм; разработана информационная модель системы, представленная в виде схемы базы данных; описаны основные алгоритмы работы программного средства; разработано руководство пользователя; выполнено тестирование системы.

### ЗАКЛЮЧЕНИЕ

В результате выполнения данного курсового проекта было разработано программное приложение, позволяющие оптимизировать работу парикмахерской. В нем предусмотрена система администрирования, а также проверка учетных данных при авторизации пользователей.

Приложение удовлетворяет основным характеристикам, которые были заявлены. Оно удобно в эксплуатации, целостно, конкретизировано в рамках заданной предметной области.

При разработке данного программного продукта была учтена логика пользователя, и интерфейс данной программы сделан удобным и понятным.

Достигнута основная цель курсового проекта: исследованы основные принципы работы парикмахерских, и на основе них был разработан программный продукт, который можно внедрить в работу любой парикмахерской.

Использование данной программы в реальной парикмахерской сделает работу автоматизированной, более быстрой и упрощенной. В дальнейшем данная программа может редактироваться и совершенствоваться в соответствии с требованиями предметной области. Подводя итог, можно сказать, что цели и задачи, поставленные перед данной работой, успешно достигнуты и выполнены.

Выполнено проектирование и разработка программного средства: выполнена постановка задачи и определены основные методы ее решения; в ходе объектного моделирования системы построен ряд *UML*-диаграмм; разработана информационная модель системы, представленная в виде схемы базы данных; описаны основные алгоритмы работы программного средства; разработано руководство пользователя; выполнено тестирование системы, показавшее ее соответствие функциональным требованиям, поставленным в задании на разработку.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Цуканова, О. А. Методология и инструментарий моделирования бизнес-процессов / О. А. Цуканова. – СПб.: Университет ИТМО, 2015. – 100 с.
2. Новиков Ф.А. Учебно-методическое пособие по дисциплине «Анализ и проектирование на UML» / Ф.А. Новиков. ­– СПб.: Университет ИТМО, 2007 – 283 с.
3. Фаулер М., Скотт К. UML. Основы. – Пер. с англ. – СПб: Символ-Плюс, 2002. – 192с.
4. Мейнджер Д. JAVA: Основы программирования. С.Пб: BHV-СанктПетербург, 1997. 320 с
5. Википедия [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/wiki/Интерфейс_пользователя>.
6. Statechart [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://it-gost.ru/articles/view_articles/97.html>.
7. Sequence diagram [Электронный ресурс]. – Электронные данные. – Режим доступа: https://flexberry.github.io/ru/fd\_sequence-diagram.html.
8. Deployment diagram [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.guru99.com/deployment-diagram-uml-example.html>.
9. Component diagram [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.lucidchart.com/pages/uml-component-diagram.html>.
10. Spring framework [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/wiki/Spring_Framework.html>.
11. Bootstrap [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://itchief.ru/bootstrap/introduction.html>.
12. ECharts [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://echarts.apache.org/en/index.html>.
13. Spring в действии. К. Уоллс – ДМК Пресс.

### ПРИЛОЖЕНИЕ А

**(обязательное)**

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

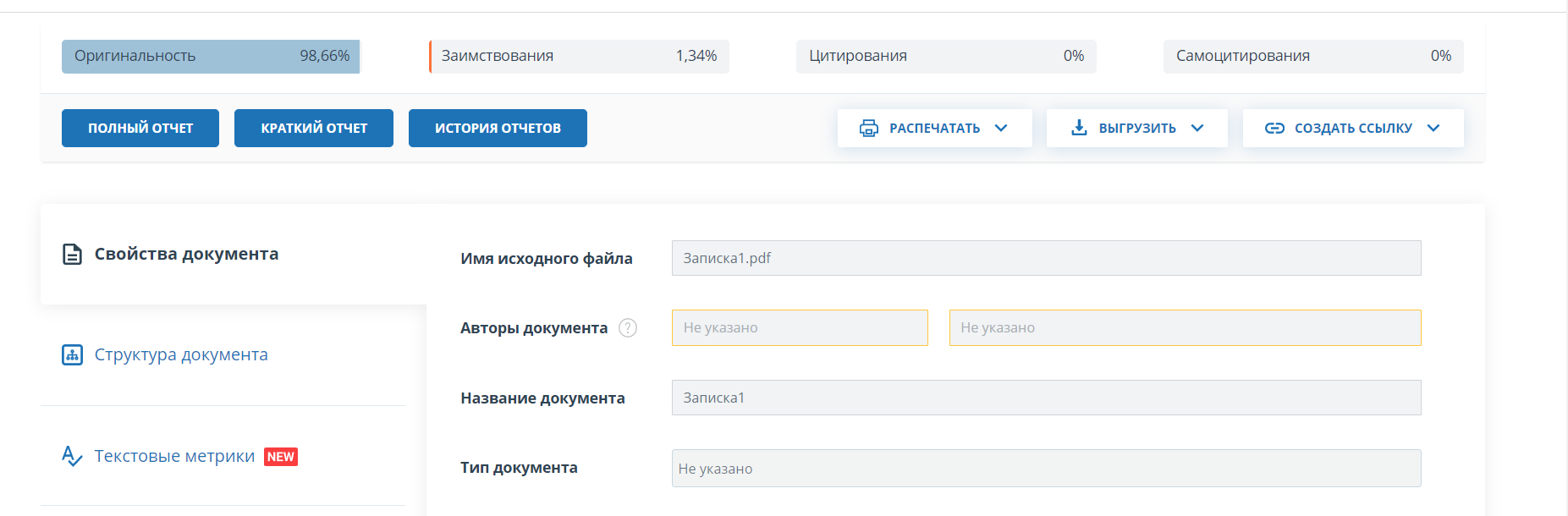
****

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

### ПРИЛОЖЕНИЕ Б

**(обязательное)**

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

Файл *BarberController*.*java*

package barbershop.web.admin;  
  
import barbershop.data.\*;  
import barbershop.entities.\*;  
import barbershop.security.RegistrationForm;  
import jakarta.validation.Valid;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.validation.BindingResult;  
import org.springframework.validation.Errors;  
import org.springframework.validation.ObjectError;  
import org.springframework.web.bind.annotation.\*;  
  
import java.text.Normalizer;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Comparator;  
import java.util.Date;  
import java.util.List;  
  
@Controller  
@RequestMapping("/admin/manageBarbers")  
public class BarberController {  
  
 private BarberRepository barberRepo;  
 private PasswordEncoder passwordEncoder;  
  
 private UserRepository userRepo;  
  
 private SpecializationRepository specializationRepo;  
  
 private TimetableRepository timetableRepo;  
  
 private AppointmentRepository appointmentRepo;  
  
 @Autowired  
 public BarberController(BarberRepository barberRepo, PasswordEncoder passwordEncoder  
 ,SpecializationRepository specializationRepo, UserRepository userRepo, TimetableRepository timetableRepo,  
 AppointmentRepository appointmentRepo){  
 this.barberRepo = barberRepo;  
 this.passwordEncoder = passwordEncoder;  
 this.specializationRepo = specializationRepo;  
 this.userRepo = userRepo;  
 this.timetableRepo = timetableRepo;  
 this.appointmentRepo = appointmentRepo;  
 }

Продолжение приложения Б

barberRepo.findAllByUserRole("barber"));  
 }  
  
 @ModelAttribute(name = "formToAddBarber")  
 public FormToAddBarber form(){return new FormToAddBarber();}  
  
 @ModelAttribute  
 public void addSpecializations(Model model){  
 model.addAttribute("specializations", specializationRepo.findAll());  
 }  
  
 @GetMapping  
 public String showUsers(){  
 return "barbers";  
 }  
  
 @GetMapping("/addBarber")  
 public String showPageForAddUser(){  
 return "addBarber";  
 }  
  
 @PostMapping("/addBarber")  
 public String processBarber(@Valid FormToAddBarber form, Errors errors, BindingResult result){  
  
  
 if(errors.hasErrors()){  
 for (int i = 0;i<form.getTimetables().size(); i++){  
 if (errors.hasFieldErrors("timetables[" + i + "].time")){  
 String err = "Пожалуйста, введите время в формате (07:00-15:00). Помните, что продолжительность рабочего дня\n не должна превышать восьми часов!";  
 ObjectError error = new ObjectError("globalError2", err);  
 result.addError(error);  
 break;  
 }  
 }  
  
 return "addBarber";  
 }  
  
 if(userRepo.findByUsername(form.getUsername())!=null){  
 String err = "Это имя пользователя уже занято!";  
 ObjectError error = new ObjectError("globalError", err);  
 result.addError(error);  
 }  
  
 for (int i = 0;i<form.getTimetables().size(); i++){  
 if(countDifferenceBetweenTime(form.getTimetables().get(i).getTime())<0||  
 countDifferenceBetweenTime(form.getTimetables().get(i).getTime())>28800000){  
 String err = "Пожалуйста, введите время в формате (07:00-15:00). Помните, что продолжительность рабочего дня\n не должна превышать восьми часов!";  
 ObjectError error = new ObjectError("globalError1", err);  
 result.addError(error);  
 break;

Продолжение приложения Б

}

}  
  
 if(result.hasErrors()){  
 return "addBarber";  
 }  
 Barber barber = form.toBarber(passwordEncoder);  
 barberRepo.save(barber);  
 for(Timetable timetable:form.getTimetables()){  
 timetable.setBarber(barber);  
 timetableRepo.save(timetable);  
 }  
  
  
 return "redirect:/admin/manageBarbers";  
 }  
  
 @GetMapping("/deleteBarber")  
 public String deleteBarber(Long id){  
  
 if(id!=null){  
 List<Timetable> timetables = timetableRepo.findAllByBarberId(id);  
 timetableRepo.deleteAll(timetables);  
 List<Appointment> appointments = appointmentRepo.findAllByBarberId(id);  
 appointmentRepo.deleteAll(appointments);  
 barberRepo.deleteById(id);  
 }  
  
  
 return "redirect:/admin/manageBarbers";  
 }  
  
 @GetMapping("/setSchedule")  
 public String setBarberSchedule(Long id, Model model){  
 model.addAttribute("barberToSetSchedule", barberRepo.findById(id));  
 ScheduleForm scheduleForm = new ScheduleForm();  
 scheduleForm.setTimetables(timetableRepo.findAllByBarberId(id));  
 model.addAttribute("scheduleForm", scheduleForm);  
 return "setBarberSchedule";  
 }  
  
 @PostMapping("/setSchedule")  
 public String processScheduleForm(@ModelAttribute ScheduleForm scheduleForm, BindingResult result, Model model){  
 for(Timetable timetable: scheduleForm.getTimetables()){  
 if(timetable.getTime().equals(null)||!timetable.getTime().matches("[0-9]{2}:[0-9]{2}-[0-9]{2}:[0-9]{2}")||  
 countDifferenceBetweenTime(timetable.getTime())<0||countDifferenceBetweenTime(timetable.getTime())>28800000){  
 String err = "Пожалуйста, введите время в формате (07:00-15:00). Помните, что продолжительность рабочего дня\n не должна превышать восьми часов!";  
 ObjectError error = new ObjectError("globalError2", err);  
 result.addError(error);

Продолжение приложения Б

break;

}  
 }  
  
 if(result.hasErrors()){  
 return "setBarberSchedule";  
 }  
  
 Barber barber = (Barber)model.getAttribute("barberToSetSchedule");  
  
 for(Timetable timetable:scheduleForm.getTimetables()){  
 timetable.setBarber(timetableRepo.findById(timetable.getId()).get().getBarber());  
 timetableRepo.save(timetable);  
 }  
  
 return "redirect:/admin/manageBarbers";  
 }  
  
 @GetMapping("/sort")  
 public String sort(@RequestParam(value = "sortType", required = false) String sortType  
 , @ModelAttribute("barbers") List<Barber> barbers){  
 Comparator<Barber> comparator = null;  
 if(sortType.equals("")){  
 return "barbers";  
 }  
 if(sortType.equals("name")){  
 comparator = new Comparator<Barber>() {  
 @Override  
 public int compare(Barber o1, Barber o2) {  
 return o1.getUser().getName().compareTo(o2.getUser().getName());  
 }  
 };  
 }  
  
 if(sortType.equals("surname")){  
 comparator = new Comparator<Barber>() {  
 @Override  
 public int compare(Barber o1, Barber o2) {  
 return o1.getUser().getSurname().compareTo(o2.getUser().getSurname());  
 }  
 };  
 }  
  
 if (sortType.equals("username")){  
 comparator = new Comparator<Barber>() {  
 @Override  
 public int compare(Barber o1, Barber o2) {  
 return o1.getUser().getUsername().compareTo(o2.getUser().getUsername());  
 }  
 };

Продолжение приложения Б

}  
  
 barbers.sort(comparator);  
  
 return "barbers";  
 }  
  
 public long countDifferenceBetweenTime(String time){  
 String []times = getTwoTimes(time);  
 SimpleDateFormat simpleDateFormat = new SimpleDateFormat("HH:mm");  
 try{  
 Date date1 = simpleDateFormat.parse(times[0]);  
 Date date2 = simpleDateFormat.parse(times[1]);  
 return (date2.getTime() - date1.getTime());  
 } catch (ParseException e) {  
 throw new RuntimeException(e);  
 }  
  
 }  
  
 public String[] getTwoTimes(String unsplitted){  
 return unsplitted.split("-");  
 }  
}

### ПРИЛОЖЕНИЕ В

**(обязательное)**

**Листинг скрипта генерации базы данных**

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

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

-- Schema mydb

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

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

-- Schema barbershop

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

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

-- Schema barbershop

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

CREATE SCHEMA IF NOT EXISTS `barbershop` ;

USE `barbershop` ;

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

-- Table `barbershop`.`user`

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

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

`user\_id` INT NOT NULL AUTO\_INCREMENT,

`username` VARCHAR(45) NOT NULL,

`password` VARCHAR(45) NOT NULL,

`role` VARCHAR(45) NOT NULL,

`name` VARCHAR(45) NULL,

`surname` VARCHAR(45) NULL,

PRIMARY KEY (`user\_id`))

ENGINE = InnoDB;

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

-- Table `barbershop`.`admin`

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

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

`admin\_id` INT NOT NULL AUTO\_INCREMENT,

`email` VARCHAR(45) NULL,

`user\_user\_id` INT NOT NULL,

PRIMARY KEY (`admin\_id`, `user\_user\_id`),

INDEX `fk\_admin\_user1\_idx` (`user\_user\_id` ASC) VISIBLE,

CONSTRAINT `fk\_admin\_user1`

Продолжение приложения В

FOREIGN KEY (`user\_user\_id`)

REFERENCES `barbershop`.`user` (`user\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `barbershop`.`specialization`

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

CREATE TABLE IF NOT EXISTS `barbershop`.`specialization` (

`id\_specialization` INT NOT NULL AUTO\_INCREMENT,

`specialization` VARCHAR(45) NULL,

`price` VARCHAR(45) NULL,

PRIMARY KEY (`id\_specialization`))

ENGINE = InnoDB;

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

-- Table `barbershop`.`barber`

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

CREATE TABLE IF NOT EXISTS `barbershop`.`barber` (

`barber\_id` INT NOT NULL AUTO\_INCREMENT,

`experience` VARCHAR(45) NULL,

`user\_user\_id` INT NOT NULL,

`specialization\_id\_specialization` INT NOT NULL,

PRIMARY KEY (`barber\_id`, `user\_user\_id`, `specialization\_id\_specialization`),

INDEX `fk\_barber\_user1\_idx` (`user\_user\_id` ASC) VISIBLE,

INDEX `fk\_barber\_specialization1\_idx` (`specialization\_id\_specialization` ASC) VISIBLE,

CONSTRAINT `fk\_barber\_user1`

FOREIGN KEY (`user\_user\_id`)

REFERENCES `barbershop`.`user` (`user\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_barber\_specialization1`

FOREIGN KEY (`specialization\_id\_specialization`)

REFERENCES `barbershop`.`specialization` (`id\_specialization`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `barbershop`.`client`

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

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

`client\_id` INT NOT NULL AUTO\_INCREMENT,

`phone` VARCHAR(45) NULL,

`user\_user\_id` INT NOT NULL,

# ВЕДОМОСТЬ ДОКУМЕНТОВ КУРСОВОГО ПРОЕКТА