

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

**Мобильное программное приложение для
поддержки процессов функционирования
косметического салона**

Студент

(подпись, дата)

Сочко Марина Сергеевна

(ФИО полностью)

Руководитель

(подпись, дата)

Князев В.Н.

(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.

(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

(подпись)

Козлов А.Ю.

(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

Попова Н.А.

(фамилия, инициалы)

Пенза, 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»

Заведующий кафедрой

А.Ю. Козлов

«20» декабря 2024 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Сочко Марина Сергеевна гр. 21ВП2 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Мобильное программное приложение для поддержки процессов
функционирования косметического салона

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы Князев В.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) *Разработать приложение для мобильных устройств,
поддержку процессов функционирования косметического салона.*

2) *Среда выполнения приложения: операционная система Android 9 и
Среда выполнения сервера: операционная система Windows 10 и выше*

Языковые средства разработки: C#, UML, SQL, JAVA

Инструментальные средства разработки: PostgreSQL,

Microsoft Visual Studio 2022, Microsoft Project, Visual Paradigm, Android Studio

Технологии разработки: RUP, ООП, Entity Framework, MAUI

3) *Основные функции мобильного приложения:*

а) запись к мастеру на процедуру;

б) просмотр списка услуг и их стоимости;

в) просмотр портфолио мастеров;

г) оставление отзывов о посещении.

5. Перечень подлежащих разработке вопросов

1) *Анализ предметной области и постановка задачи*

2) *Проектирование*

3) *Реализация*

4) *Контроль качества программного обеспечения*

5) *Оформление пояснительной записки*

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2025 – 13.04.2025	
2	Проектирование программных средств	20%	15.04.2025 – 25.04.2025	
3	Реализация программных средств	30%	26.04.2025 – 12.05.2025	
4	Тестирование программных средств	20%	13.05.2025 – 22.05.2025	
5	Оформление пояснительной записки	20%	23.05.2025 – 03.06.2025	

Дата выдачи задания «20» декабря 2024г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить

Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 284 листов, 61 рисунок, 16 таблиц, 35 использованных источника, 6 приложений.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, КОСМЕТИЧЕСКИЙ САЛОН,
ANDROID, POSTGRESQL, C#, JAVA, MAUI, .NET, RUP, UML

Объектом разработки является мобильное приложение для поддержки процессов функционирования косметического салона.

Цель работы – разработка мобильного программного приложения для поддержки процессов функционирования косметического салона, позволяющего создавать и отменять записи на к мастерам, смотреть информацию о мастерах и услугах, смотреть портфолио мастеров и управлять отзывами.

Разработка серверной части происходила на языке программирования C#. В качестве среды разработки использовалась Microsoft Visual Studio 2022. В качестве СУБД была использована СУБД PostgreSQL.

Разработка серверной части производилась на языке программирования C#. В качестве среды разработки использовалась среда программирования Microsoft Visual Studio 2022. В качестве СУБД была использована СУБД PostgreSQL. Разработка приложения производилось на языке программирования Java. В качестве среды разработки использовалась среда программирования Android Studio. Приложение предназначено для использования под управлением ОС Android версии 9.0 и выше, а его серверная часть работает под управлением операционной системы Windows 10 и выше.

					ПГУ 09.03.04 – 08БР212.26 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Мобильное программное приложение для поддержки процессов функционирования косметического салона. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Сочко М.С.						
Пров.		Князев В.Н.					4	284
Н. контр.		Такташкин Д.В.				Группа 21ВП2		
Утверд.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Анализ предметной области и требований.....	9
1.1 Анализ предметной области и постановка задачи.....	9
1.2 Анализ программных продуктов аналогичного назначения	9
1.2.1 Мобильное приложение «ElasticNails».....	9
1.2.2 Мобильное приложение «Lover».....	11
1.2.3 Мобильное приложение «Y2Best»	13
1.2.4 Сравнение аналогов	15
1.3 Анализ целевой аудитории	15
1.4 Анализ функциональных требований	18
1.5 Анализ нефункциональных требований	24
1.6 Планирование разработки и оценка бюджета.....	24
2 Проектирование мобильного приложения	38
2.1 Выбор технологий и инструментальных средств разработки.....	38
2.2 Проектирование архитектуры программного обеспечения.....	39
2.3 Проектирование интерфейса пользователя	40
2.4 Логическое проектирование базы данных	48
2.5 Проектирование серверной части приложения	51
3 Реализация мобильного приложения	53
3.1 Реализация базы данных.....	53
3.2 Разработка серверной части	53
3.3 Реализация клиентской части приложения	56
4 Тестирование программного продукта	60
4.1 Тестирование серверной части приложения	60
4.2 Тестирование клиентской части приложения	61
4.2.1 Функциональное тестирование	61
4.2.2 Модульное тестирование	62
4.3 Метрики кода.....	63
4.3.1 Метрики размера программы.....	65

4.3.2 Метрики Холстеда	66
Заключение	69
Список использованных источников	70
Приложение А (справочное) Глоссарий	74
Приложение Б (обязательное) Исходный код скриптов базы данных	76
Приложение В (обязательное) Исходный код серверной части	83
Приложение Г (обязательное) Исходный код клиентской части.....	180
Приложение Д (рекомендуемое) Результаты тестирования серверной части приложения	248
Приложение Е (рекомендуемое) Результаты тестирования клиентской части приложения	251

Введение

В современном мире информационные технологии приобретают всё большее распространение во всех сферах человеческой жизни. Год от года компьютеры все чаще применяют в различных частях сферы услуг красоты во всех уголках планеты.

Косметические салоны – это неотъемлемая часть современной индустрии услуг, центры, где каждый желающий может получить комплексный уход за внешностью и телом, улучшить свое самочувствие и повысить уверенность в себе. Это не просто места оказания услуг, но и пространства для релаксации, преображения и даже социального взаимодействия. Спектр услуг, предлагаемых современными салонами, чрезвычайно широк и постоянно расширяется, отвечая на растущие потребности и тренды в сфере красоты и здоровья.

Типичный косметический салон предлагает широкий перечень процедур, включающий в себя:

- парикмахерские услуги: стрижки, укладки, окрашивания любой сложности, химическая завивка, выпрямление волос, уходовые и восстанавливающие процедуры для волос;
- ногтевой сервис: маникюр, педикюр (классический, аппаратный, комбинированный), покрытие гель-лаком, наращивание ногтей, дизайн;
- косметология лица и тела: чистки, пилинги, уходовые процедуры по типу кожи, массаж лица, аппаратная косметология, инъекционные методики, коррекция бровей и ресниц, депиляция.

Посещение косметического салона – это не просто получение услуги, это целый ритуал самопознания и заботы о себе. Клиенты приходят сюда не только за идеальной прической или ухоженными руками, но и за атмосферой релаксации, профессиональными советами, возможностью отдохнуть от повседневной суеты и почувствовать себя обновленным. В условиях высокой конкуренции и постоянно меняющихся трендов, современный косметический салон сталкивается с рядом вызовов: эффективное управление записью

клиентов, ведение клиентской базы, информирование об акциях и новинках, поддержание лояльности и, конечно, обеспечение высокого уровня сервиса. Традиционные методы ведения бизнеса – телефонные звонки, бумажные журналы записей – становятся все менее эффективными и трудозатратными.

Эта тенденция не обошла стороной и Российскую Федерацию. В связи с тем, что интеграция информационных технологий способна многократно повысить удобство и эффективность этих самых процессов, правительство Российской Федерации в 2017 году утвердило программу «Цифровая экономика Российской Федерации». Данная программа подразумевает «развитие цифровой экономики Российской Федерации, в которой данные в цифровой форме являются ключевым фактором производства во всех сферах экономической деятельности, что повышает конкурентоспособность страны, качество жизни граждан, обеспечивает экономический рост и экономический суверенитет» [1].

Одной из важнейших особенностей развития информационных технологий в нашей стране, да и во всем современном мире, является повсеместное распространение мобильных устройств с доступом к интернету. При этом мобильный интернет постепенно оттесняет на второй план традиционный доступ к сети с использованием стационарных компьютеров по причине своего существенно большего удобства. В частности, как отмечено в государственной программе «Цифровая экономика», в 2016 году в нашей стране «на 100 человек приходилось 159,95 мобильного телефона и из 100 человек 71,29 человека использовали мобильный доступ к сети Интернет» [2]. Со временем эта тенденция лишь усиливается.

Опираясь на приведённые выше факторы, можно сделать вывод о том, что разработка мобильного приложения для косметического салона весьма актуальна и перспективна в условиях современного общества.

1 Анализ предметной области и требований

1.1 Анализ предметной области и постановка задачи

Разрабатываемое мобильное приложение должно обеспечить пользователям максимально удобный и интуитивно понятный интерфейс.

Таким образом, в приложении должны быть представлены следующие основные функции:

- регистрация новых пользователей;
- вход зарегистрированных пользователей в систему;
- запись к мастеру на процедуру;
- просмотр списка услуг и их стоимости;
- просмотр портфолио мастеров;
- оставление отзывов о посещении.

Помимо рядовых пользователей в системе будут присутствовать и администраторы, которые будут иметь более широкие возможности действий. В частности, они должны быть способны добавлять в систему новых пользователей вручную, например: уволившихся мастеров или клиентов, которые больше не пользуются услугами салона.

1.2 Анализ программных продуктов аналогичного назначения

В настоящее время существует целый ряд приложений, использующихся при работе косметических салонов. Рассмотрим наиболее известные российские мобильные приложения такого рода и произведём для них сравнительный анализ.

1.2.1 Мобильное приложение «ElasticNails»

ElasticNails – это система записи на ногтевой сервис, разработанная специально для косметических салонов и частных мастеров, специализирующихся на маникюре и педикюре [3]. Она предлагает спектр функций, которые позволяют эффективно управлять расписанием, оптимизировать запись клиентов и контролировать загруженность мастеров.

Система ElasticNails помогает салонам и мастерам, подобным себе, удерживать позиции на рынке и обеспечивать высокий уровень обслуживания клиентов. Система обладает функционалом для выполнения своих ключевых задач.

Функции управления услугами включают в себя:

- добавление и редактирование информации об услугах (название, описание, длительность, цена);
- категоризация услуг по типу (маникюр, педикюр, наращивание и т.д.).

Функции управления расписанием содержат следующее:

- создание и редактирование графика работы мастеров;
- блокировка времени для перерывов или нерабочих дней;
- отслеживание занятого и свободного времени;
- уведомления о новых записях и отменах.

Интерфейс мобильного приложения представлен на рисунке 1.

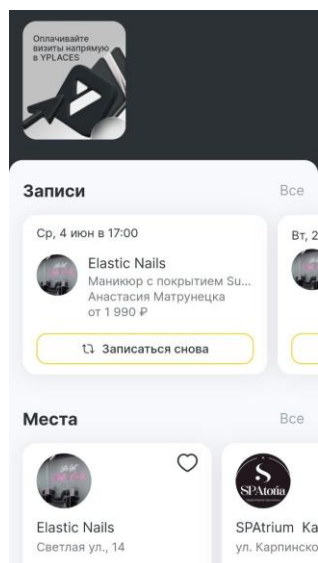


Рисунок 1 – Интерфейс ElasticNails

Система записи ElasticNails – это инструмент для косметических салонов и частных мастеров, специализирующихся на ногтевом сервисе, которые стремятся к успеху и развитию. Благодаря спектру функций и удобному интерфейсу, система ElasticNails помогает оптимизировать бизнес-процессы, повысить эффективность и увеличить прибыль.

1.2.2 Мобильное приложение «Lover»

Lover – это система записи на услуги визажиста, которая используется в крупных косметических салонах и студиях макияжа [4]. Эта система обладает рядом преимуществ, позволяющих управлять обширным спектром услуг и большим количеством мастеров. Однако, как и любая другая система, она имеет свои недостатки. Система обладает широким функционалом для выполнения своих ключевых задач.

Функции управления услугами включают в себя:

- добавление и редактирование информации об услугах макияжа;
- категоризация услуг по типу макияжа (дневной, вечерний, свадебный и т.д.), длительности и стоимости;
- анализ популярности услуг и сезонных трендов;
- установка доступности услуг для записи.

Функции управления мастерами содержат в себе:

- ведение базы данных визажистов с информацией об опыте, специализации и портфолио;
- формирование расписания работы мастеров с учетом их занятости и выходных;
- отслеживание загруженности каждого мастера.

Функции управления записями вмещают в себя:

- формирование записей клиентов с учетом выбранной услуги, мастера и времени;
- ведение истории посещений клиентов.

Недостатками системы записи Lover является следующее:

- сложность системы: большое количество функций и настроек может затруднить работу с системой для новых администраторов и потребовать дополнительного обучения;
- недостаточная гибкость: система может быть недостаточно гибкой для адаптации к быстро меняющимся условиям работы салона и потребностям

клиентов (например, при введении новых услуг или изменении графика мастеров);

- интеграция с другими системами: могут возникать сложности с интеграцией системы записи с другими системами, используемыми в салоне (например, с системами учета материалов);

- стоимость: внедрение и обслуживание системы записи может быть затратным для салона, особенно если речь идет о крупной студии с большим штатом мастеров.

Интерфейс мобильного приложения представлен на рисунке 2.

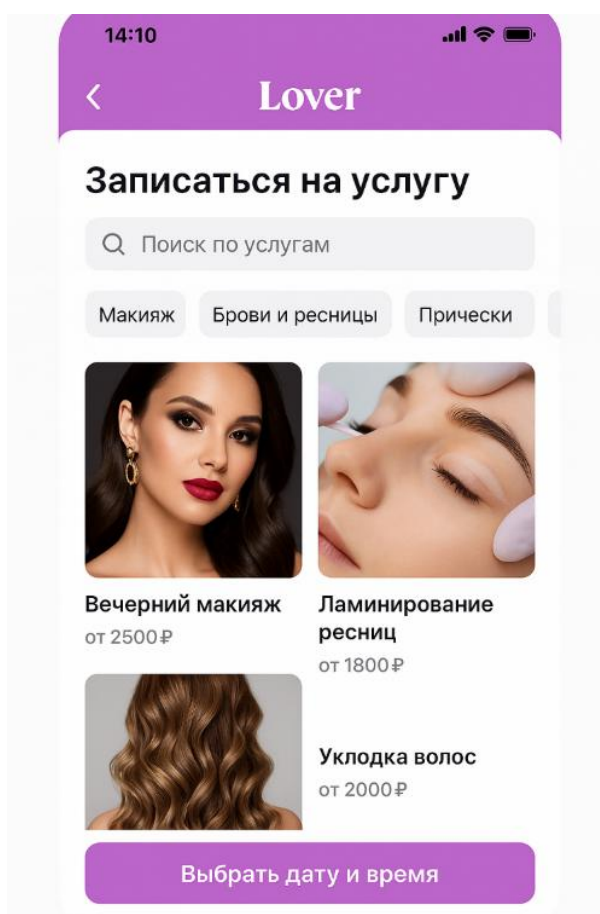


Рисунок 2 – Интерфейс Lover

Несмотря на недостатки, система записи Lover позволяет эффективно управлять записями для крупных салонов и студий макияжа. Однако, компании стоит обратить внимание на упрощение системы, повышение ее гибкости и улучшение интеграции с другими системам.

1.2.3 Мобильное приложение «Y2Best»

Y2Best – это система управления косметическим салоном, предназначенная для комплексного ведения всех видов услуг, бизнес-процессов и т.д. [5]. Система обладает широким функционалом для выполнения своих ключевых задач, но ее внедрение и использование могут быть затратными.

Функции управления услугами включают в себя:

- добавление и редактирование информации о всех услугах салона (парикмахерские услуги, ногтевой сервис, косметология, массаж и т.д.);
- категоризация услуг по типу, длительности, стоимости и специализации мастеров;
- отслеживание популярности различных услуг;
- управление прайс-листом и акциями.

Функции управления мастерами содержат следующее:

- ведение базы данных всех мастеров салона с информацией об их квалификации, специализации и графике работы;
- формирование и управление расписанием для всех мастеров;
- отслеживание загруженности каждого мастера;
- расчет заработной платы мастеров с учетом выполненных услуг.

Функции управления клиентами состоят из:

- ведения единой базы данных клиентов с контактной информацией, историей посещений и предпочтениями;
- возможности ведения электронных карточек клиентов с информацией о процедурах и рекомендациях;
- внедрения системы лояльности и управления бонусными программами;
- автоматических уведомлений о предстоящих записях и акциях.

Преимущества системы Y2Best состоят из:

- комплексное управление: система охватывает все основные аспекты работы косметического салона, обеспечивая единый центр управления;

- оптимизация процессов: автоматизация рутинных задач повышает эффективность работы персонала;
- улучшение обслуживания клиентов: единая база клиентов и история посещений позволяют предлагать персонализированный сервис;
- контроль и аналитика: наличие отчетов позволяет эффективно отслеживать показатели работы салона и принимать обоснованные решения.

Недостатками системы Y2Best являются следующие аспекты:

- высокая стоимость: внедрение и обслуживание такой комплексной системы требует значительных финансовых вложений;
- сложность настройки и использования: большое количество функций может потребовать длительного обучения персонала;
- потенциальная избыточность функций: для небольших салонов часть функционала может быть избыточной и невостребованной;
- зависимость от поставщика: любые изменения или доработки системы требуют обращения к разработчику.

Интерфейс мобильного приложения представлен на рисунке 3.

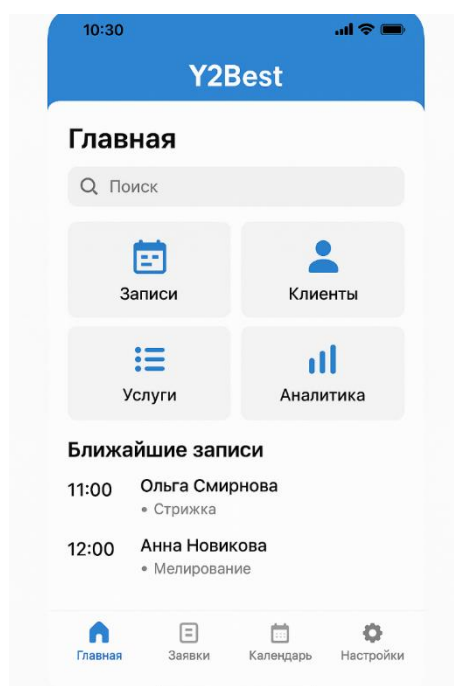


Рисунок 3 – Интерфейс Y2Best

Система Y2Best отвечает потребностям крупных косметических салонов, предоставляющих широкий спектр услуг.

1.2.4 Сравнение аналогов

Сравнительный анализ рассмотренных выше мобильных приложений представлен в таблице 1.

Таблица 1 – Сравнение аналогов

Мобильное приложение	Запись к мастеру	Оставление отзыва	Удобство интерфейса	Поддержка тёмной темы
ElasticNails	Да	Да	Перегруженный	Нет
Lover	Нет	Да	Понятный	Да
Y2Best	Да	Да	Перегруженный	Нет
Разрабатываемое приложение	Да	Да	Понятный	Да

Таким образом, в результате анализа систем аналогичного назначения было принято решение о разработке собственного приложения, которое будет лишено недостатков, выявленных у рассмотренных выше мобильных приложений.

1.3 Анализ целевой аудитории

Целевой аудиторией проекта являются клиенты и сотрудники косметических салонов. Согласно статистике РОССТАТ, – в прошлом году число косметических салонов превысило 15 тысяч. Также выросло общее количество посетителей косметических салонов до нескольких миллионов человек в год.

На рисунке 4 представлены доли основных операционных систем смартфонов на мировом рынке за последний год [12].



Рисунок 4 – Доли операционных систем смартфонов в мировом рынке

Можно заметить, что подавляющее большинство смартфонов на мировом рынке используют операционную систему Android. Единственное значимое исключение – компания Apple, использующая собственную систему iOS. Но, при

всей популярности Apple, ее продукция даже в периоды пикового спроса составляет лишь в районе 20% от мирового рынка, что в несколько раз меньше, чем доля устройств с Android. В РФ доля смартфонов Apple ниже, чем в среднем по планете [12]. Поэтому наиболее выгодным решением с точки зрения доступности продукта для пользователей будет реализация разрабатываемого приложения именно для операционной системы Android.

Статистика распространения разных версий системы Android представлена на рисунке 5 [13].



Рисунок 5 – Распространение различных версий системы Android

Из приведённой диаграммы можно увидеть, что 86,1 % пользователей системы Android использует версии не старше 9.0. Поэтому приложение будет разрабатываться таким образом, чтобы оно успешно функционировало для Android 9.0 и более новых версий системы.

Статистика популярности использования косметических салонов приведена на рисунке 6.

ПРОЦЕНТ АКТУАЛЬНОСТИ САЛОНОВ КРАСОТЫ

■ До 18 лет ■ 18-24 года ■ 25-44 года ■ 45-60 лет ■ 60 лет и старше

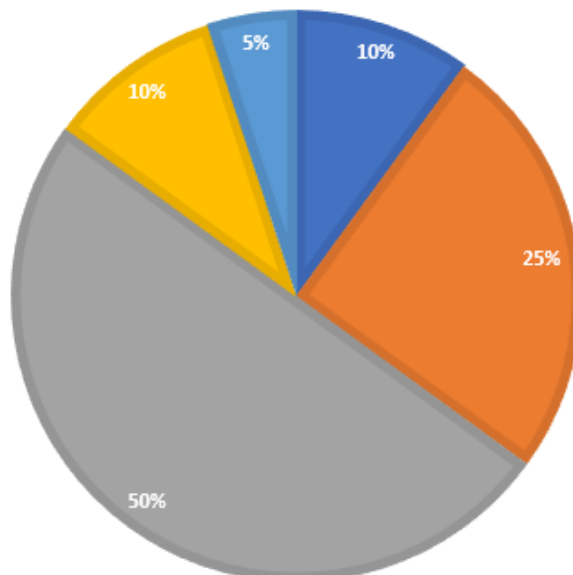


Рисунок 6 – Процент актуальности косметических салонов

Исходя из статистики, видим, что чаще всего услугами косметических салонов пользуются люди в возрасте 25-44 лет.

До 18 лет (подростки и юноши/девушки): ~ 5-10%.

Эта группа только начинает осваивать бьюти-услуги. Часто это маникюр/педикюр, коррекция бровей, простые стрижки. Решения о посещении могут приниматься с учетом мнения родителей или под влиянием трендов из социальных сетей.

18-24 года (молодежь): ~ 20-25%.

Активная группа, следящая за трендами. Часто экспериментируют с имиджем, активно пользуются услугами по уходу за волосами (сложные окрашивания, укладки), ногтевым сервисом, косметологией (чистки, уход). Бюджет ограничен, но желание выглядеть хорошо высоко.

25-44 года (основная активная аудитория): ~ 45-50%.

Наиболее обширная и платежеспособная группа. Люди в этом возрасте активно строят карьеру, ведут насыщенный образ жизни, и внешний вид играет

важную роль. Пользуются всем спектром услуг – от регулярных стрижек и окрашиваний до комплексных косметологических процедур (антивозрастные, аппаратные), массажа, наращивания ресниц, перманентного макияжа. Ценят качество, сервис и экономию времени.

45-60 лет (средний возраст): ~ 15-20%.

Эта группа ориентирована на поддержание молодости и ухоженного вида. Активно пользуются услугами по уходу за лицом (лифтинг, инъекции, пилинги), парикмахерскими услугами (частое окрашивание для маскировки седины), уходовыми процедурами для рук и ног. Ценят профессионализм мастеров и комфорт.

60 лет и старше (старшее поколение): ~ 5-10%.

Часто это постоянные клиенты, которые ценят привычных мастеров и базовые услуги – стрижки, укладки, окрашивание, простой маникюр. В меньшей степени склонны к экспериментам и дорогостоящим новым процедурам, но регулярно пользуются тем, что им необходимо для поддержания внешнего вида.

1.4 Анализ функциональных требований

Приложение может применяться мастерами и клиентами различных косметических салонов. Приложение позволяет создавать и отменять записи на процедуры, вести учет истории посещений и оставлять отзывы о мастерах. Также клиент может просматривать актуальный список услуг и их стоимость, а также ознакомиться с портфолио мастеров.

Посещение косметического салона подразумевает два основных аспекта: выбор мастера и услуги, и непосредственно сама процедура.

Выбор мастера и услуги клиент осуществляет самостоятельно, после чего может записаться на прием в соответствующем разделе своего приложения.

Процедуры проводят мастера и требуют предварительной записи на конкретную дату и время. Осуществить запись могут как клиент, так и администратор салона. При необходимости данная запись может быть отменена.

Полное завершение посещения клиента считается после проведения процедуры и возможности оставить отзыв о качестве оказанной услуги и работе мастера. Согласно внутренним правилам салона, важно отслеживать удовлетворенность клиентов и получать обратную связь.

Отслеживать историю посещений, записи и оставленные отзывы имеют возможность как клиент, так и администратор салона, в соответствующем разделе мобильного приложения.

В результате анализа требований задания была создана диаграмма вариантов использования (рисунок 7).

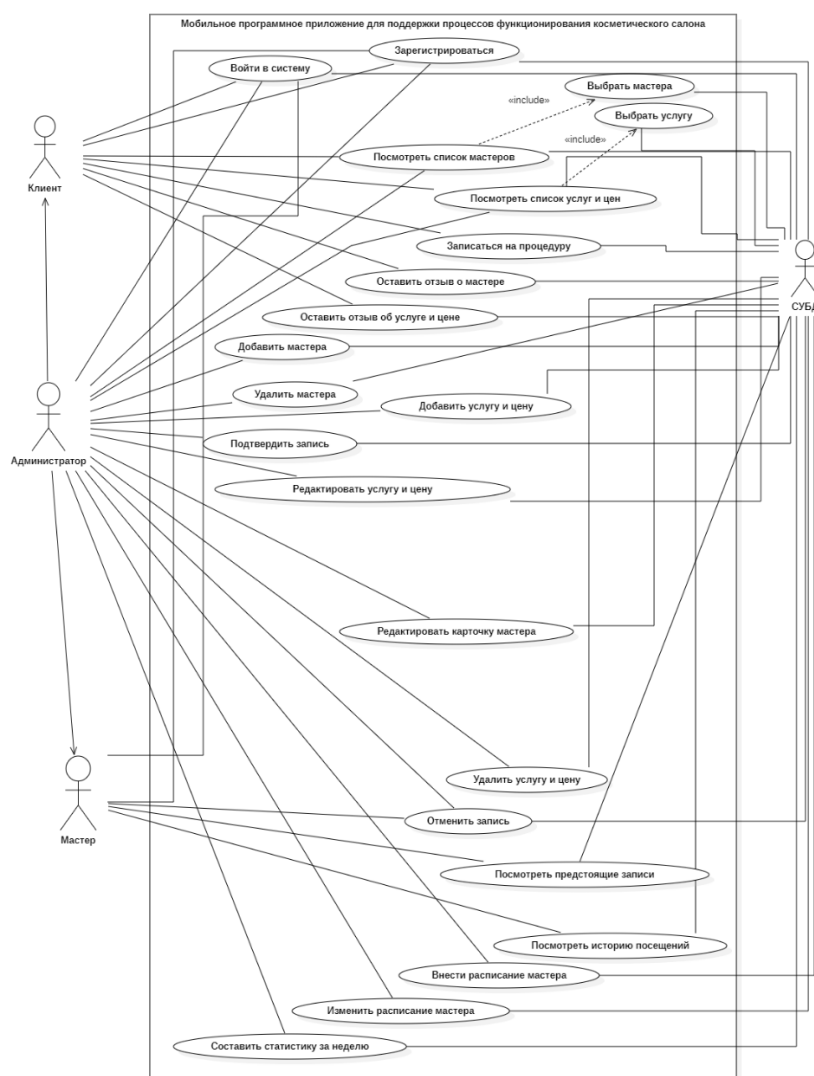


Рисунок 7 – Диаграмма вариантов использования мобильного приложения

Сценарии выполнения некоторых вариантов использования приведены на рисунках 8 - 10:

Наименование: регистрация аккаунта.
ID: 1
Краткое описание: клиент регистрируется в системе для получения индивидуальных предложений.
Действующие лица: клиент
Предусловие: клиент не зарегистрирован в системе
Основной поток: 1. Клиент вводит личные данные (ФИО, номер телефона, логин, пароль). 2. Клиент успешно проходит аутентификацию.
Постусловие: клиент зарегистрирован и авторизован в системе.

Рисунок 8 – Описание сценария варианта использования «Регистрация аккаунта» для незарегистрированного клиента

Наименование: запись на прием
ID: 2
Краткое описание: клиент записывается на прием
Действующие лица: клиент
Предусловие: -
Основной поток: 1. Клиент выбирает мастера 2. Клиент выбирает доступную дату 3. Клиент выбирает доступное время 4. Клиент выбирает доступные услуги 5. Клиент вводит персональные данные (если не авторизован в системе) 6. Клиент записан на прием
Постусловие: клиент записан на прием

Рисунок 9 – Описание сценария варианта использования «Запись на прием» для зарегистрированного или не зарегистрированного клиента

Наименование: оставить отзыв о мастере
ID: 3
Краткое описание: клиент оставляет отзыв о мастере
Действующие лица: клиент
Предусловие: клиент авторизован в системе
Основной поток: 1. Клиент открывает историю посещений студии 2. Клиент выбирает мастера 3. Клиент оставляет отзыв
Постусловие: отзыв о мастере

Рисунок 10 – Описание сценария варианта использования «Оставить отзыв о мастере» при успешной аутентификации пользователя

Сценарии выполнения некоторых вариантов использования приведены на рисунках 11 - 13:

Наименование: оставить отзыв о услуге
ID: 4
Краткое описание: клиент оставляет отзыв о услуге
Действующие лица: клиент
Предусловие: клиент авторизирован в системе
Основной поток: 1. Клиент открывает историю посещений студии 2. Клиент выбирает услугу 3. Клиент оставляет отзыв
Постусловие: отзыв о услуге

Рисунок 11 – Описание сценария варианта использования «Оставить отзыв о услуге» при успешной аутентификации пользователя

Наименование: удалить услугу
ID: 5
Краткое описание: администратор удаляет услугу
Действующие лица: администратор
Предусловие: администратор авторизирован в системе
Основной поток: 1. Администратор открывает список услуг 2. Администратор выбирает услугу 3. Администратор удаляет услугу
Постусловие: измененный список услуг

Рисунок 12 – Описание сценария варианта использования «Удалить услугу» при успешной аутентификации пользователя

Наименование: удалить мастера
ID: 6
Краткое описание: администратор удаляет мастера
Действующие лица: администратор
Предусловие: администратор авторизирован в системе
Основной поток: 1. Администратор открывает список мастеров 2. Администратор выбирает мастера 3. Администратор удаляет мастера
Постусловие: измененный список мастеров

Рисунок 13 – Описание сценария варианта использования «Удалить мастера» при успешной аутентификации пользователя

Для некоторых из рассмотренных выше вариантов использования было принято решение разработать диаграммы последовательности и деятельности, с целью рассмотреть данные варианты использования подробнее.

Для варианта использования «Записаться на процедуру» была составлена диаграмма последовательности, представленная на рисунке 14.

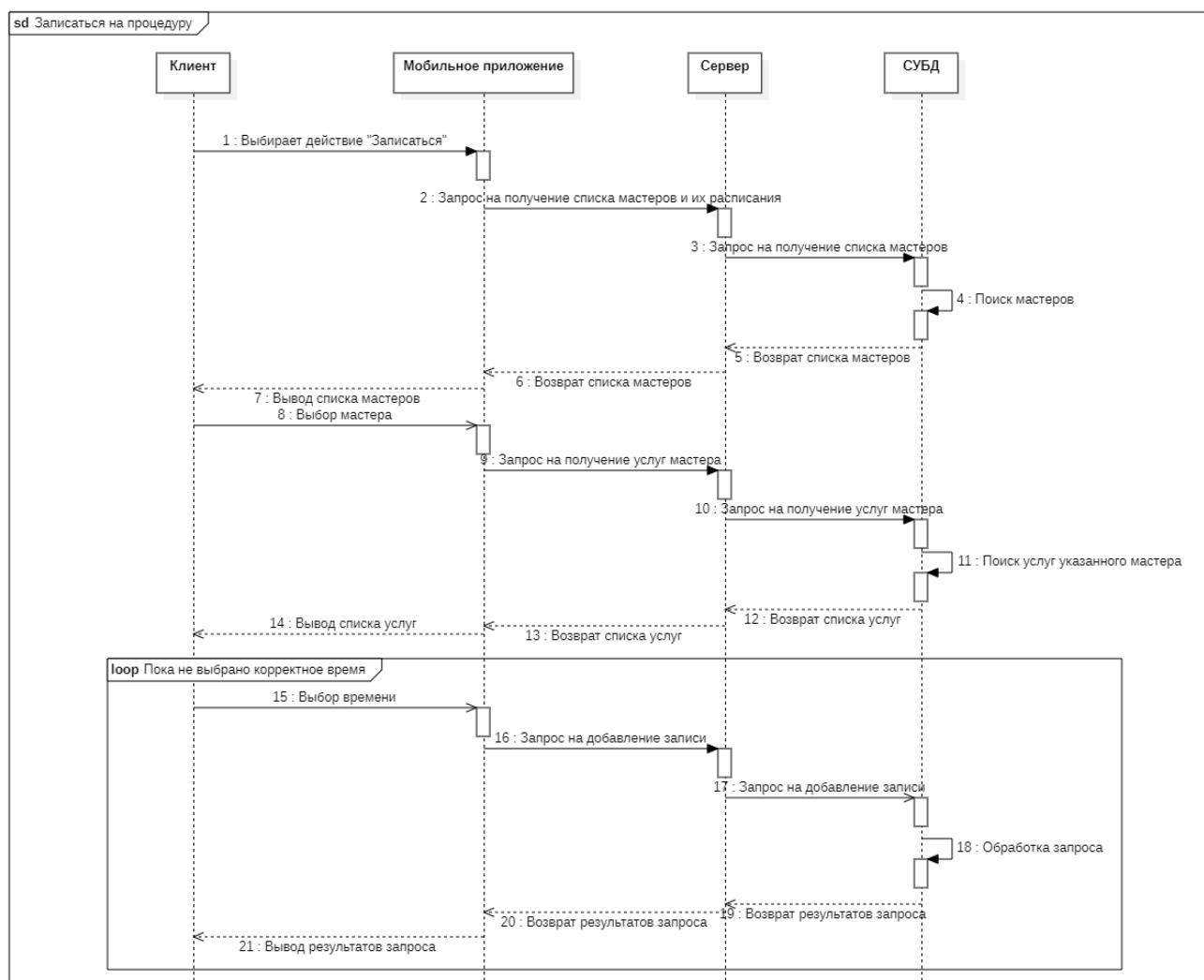


Рисунок 14 – Диаграмма последовательности варианта использования
«Записаться на процедуру»

Для варианта использования «Зарегистрироваться в системе косметического салона» составлена диаграмма деятельности, представленная на рисунке 15.

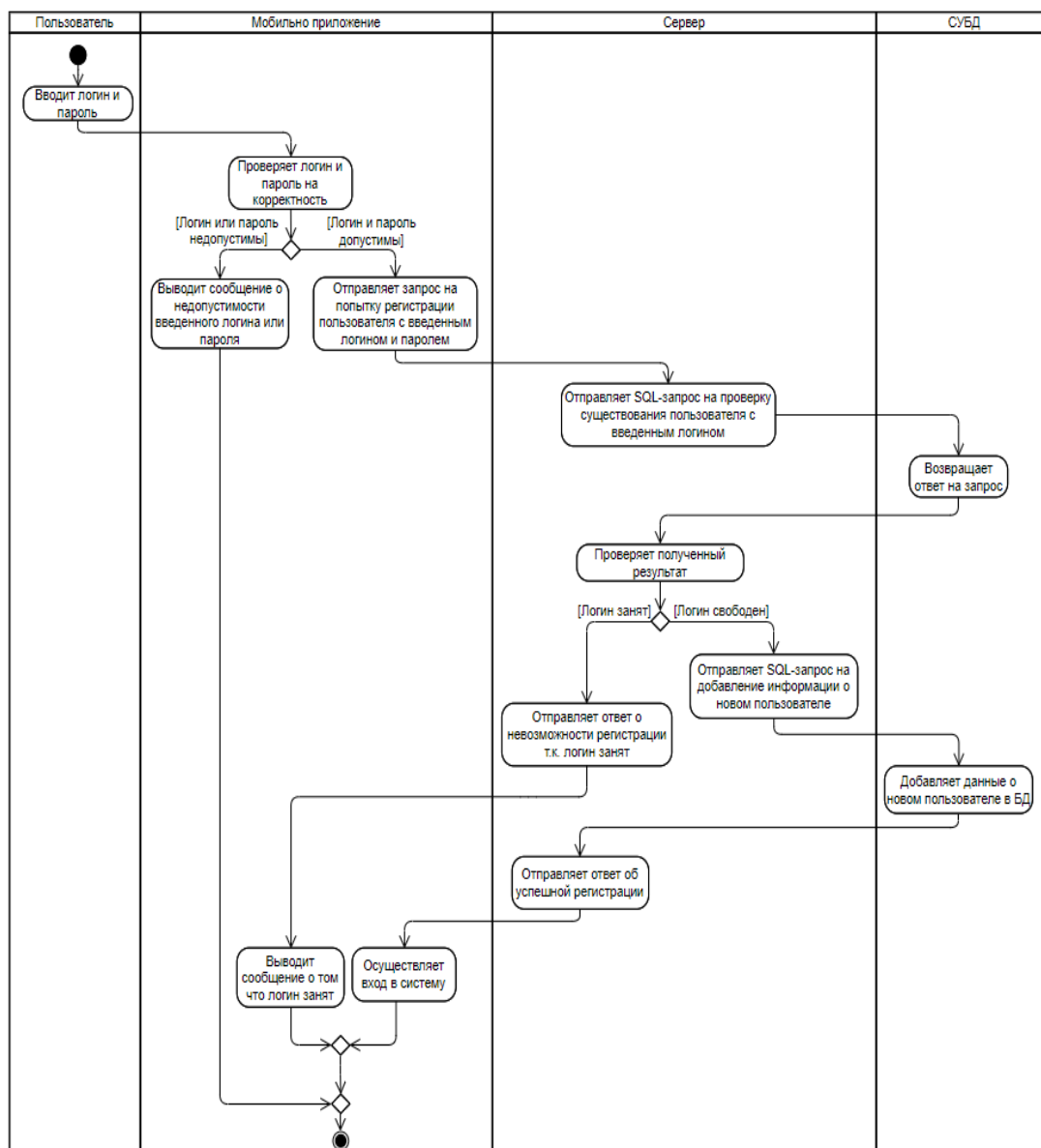


Рисунок 15 – Диаграмма деятельности варианта использования
«Зарегистрироваться»

В результате анализа функциональных требований были успешно выявлены основные функции разрабатываемого приложения. При этом для разрабатываемого приложения были построены диаграммы вариантов использования, деятельности и последовательности, а также для некоторых выявленных вариантов использования были составлены спецификации, которые отображают работу системы косметического салона.

1.5 Анализ нефункциональных требований

Разработка серверной части приложения должна быть выполнена на широкораспространённом языке программирования C#, в среде программирования Microsoft Visual Studio 2022. Данные на сервере хранятся в базе данных, реализованной с использованием СУБД PostgreSQL, запросы к базе данных осуществляются на языке SQL. Серверная должна работать под управлением операционной системы Windows 10 и выше.

Клиентская часть приложения должна быть реализована на языке программирования C# с использованием современного и эффективного кроссплатформенного фреймворка .NET MAUI, в среде программирования Microsoft Visual Studio 2022. Она должна представлять собой мобильное приложение с удобным и интуитивно понятным пользовательским интерфейсом. Клиентское приложение должно быть отлажено и должно успешно работать под управлением ОС Android 9.0 и выше.

1.6 Планирование разработки и оценка бюджета

Качественное выполнение работ по любому проекту, в том числе и в сфере информационных технологий, невозможно при отсутствии грамотного планирования, которое позволяет обеспечить выполнение работ над проектом таким образом, чтобы добиться выполнения всех поставленных целей за минимально возможное время, с минимально возможными затратами и с приемлемым качеством [15 – 20].

Задачи, представляющие собой этапы работы над проектом, представлены в таблице 2.

Таблица 2 – Перечень задач проекта

№	Название задачи	Длительность	Предшественники
1	Начало проекта	—	—
2	Анализ требований	—	—
3	Анализ предметной области и постановка задачи	2 дня	1
4	Анализ программных продуктов аналогичного назначения	4 дня	1

Продолжение таблицы 2

№	Название задачи	Длительность	Предшественники
5	Анализ функциональных требований	1 день	3; 4
6	Анализ требований к интерфейсу	1 день	5
7	Анализ целевой аудитории	2 дня	6
8	Анализ требований завершён	–	7
9	Выбор технологий разработки ПО	–	–
10	Анализ существующих технологий разработки ПО	2 дня	1
11	Выбор технологий разработки ПО	1 день	10
12	Выбор средств разработки ПО завершён	–	11
13	Проектирование программных средств	–	–
14	Проектирование базы данных	6 дней	8; 12
15	Проектирование архитектуры приложения	4 дня	14
16	Проектирование интерфейса пользователя	5 дней	8; 12
17	Проектирование приложения завершено	–	16; 15
18	Реализация программных средств	–	–
19	Реализация базы данных	8 дней	14; 15
20	Реализация серверной части приложения	21 день	17; 19
21	Реализация мобильного приложения	14 дней	17; 19
22	Реализация завершена	–	20; 21
23	Тестирование	–	–
24	Функциональное тестирование	3 дня	22
25	Модульное тестирование	3 дня	22
26	Тестирование завершено	–	24; 25
27	Составление и оформление документации	7 дней	26
28	Конец проекта	–	27

Таким образом, для реализации проекта нужно выполнить 28 задач.

В таблице 3 представлено описание затрачиваемых ресурсов.

Таблица 3 – Список ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Программист1	Трудовой	350 Р/ч	450 Р/ч
Программист2	Трудовой	350 Р/ч	450 Р/ч
Программист3	Трудовой	550 Р/ч	750 Р/ч
Дизайнер	Трудовой	250 Р/ч	350 Р/ч
Интернет	Затраты	2500 Р/месяц	
Компьютер	Материальный	500 Р	
Мобильное устройство	Материальный	300 Р	

Таким образом, в общей сложности, потребуется семь видов ресурсов.

Распределение ресурсов по задачам представлено в таблице 4.

Таблица 4 – Назначение ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало проекта	–	–
2	Анализ требований	–	–
3	Анализ предметной области и постановка задачи	Программист 3 Компьютер Интернет	50% 1 шт.
4	Анализ программных продуктов аналогичного назначения	Программист 1 Программист 2 Компьютер Интернет Мобильное устройство	50% 50% 2 шт. 2 шт.
5	Анализ функциональных требований	Программист 3 Программист 1 Компьютер Интернет	50% 50% 2 шт.
6	Анализ требований к интерфейсу	Программист 3 Программист 2 Дизайнер Компьютер Интернет	50% 50% 100% 3 шт.
7	Анализ целевой аудитории	Программист 1 Компьютер Интернет	50% 3 шт.
8	Анализ требований завершён	–	–
9	Выбор технологий разработки ПО	–	–
10	Анализ существующих технологий разработки ПО	Программист 3 Компьютер Интернет	25% 1 шт.
11	Выбор технологий разработки ПО	Программист 3 Программист 2 Программист 1 Компьютер Интернет	50% 50% 50% 3 шт.
12	Выбор средств разработки ПО завершён	–	–
13	Проектирование программных средств	–	–
14	Проектирование базы данных	Программист 3 Программист 1 Компьютер Интернет	50% 50% 2 шт.
15	Проектирование архитектуры приложения	Программист 3 Программист 1 Компьютер Интернет	50% 50% 2 шт.
16	Проектирование интерфейса пользователя	Программист 3 Программист 2 Дизайнер Компьютер Интернет	50% 50% 100% 3 шт.
17	Проектирование приложения завершено	–	–

Продолжение таблицы 4

№	Название задачи	Ресурс	Единицы
18	Реализация программных средств	—	—
19	Реализация базы данных	Программист 1 Компьютер Интернет	100% 1 шт.
20	Реализация серверной части приложения	Программист 3 Программист 1 Компьютер Интернет	50% 100% 2 шт.
21	Реализация мобильного приложения	Программист 3 Программист 2 Дизайнер Компьютер Интернет	50% 100% 100% 3 шт.
22	Реализация завершена	—	—
23	Тестирование	—	—
24	Функциональное тестирование	Программист1 Программист2 Компьютер Интернет Мобильное устройство	50% 50% 2 шт. 2 шт.
25	Модульное тестирование	Программист1 Программист2 Компьютер Интернет Мобильное устройство	50% 50% 2 шт. 2 шт.
26	Тестирование завершено	—	—
27	Составление и оформление документации	Программист 3 Программист 2 Программист 1 Компьютер Интернет	100% 100% 100% 3 шт.
28	Конец проекта	—	—

Таким образом, реализацией проекта будут заниматься три программиста и один дизайнер, которым для работы нужны компьютеры, мобильные устройства и доступ к интернету.

Для вычисления длительности реализации проекта и его критического пути был построен сетевой график, отображающий последовательность выполнения всех задач по проекту (рисунок 16).

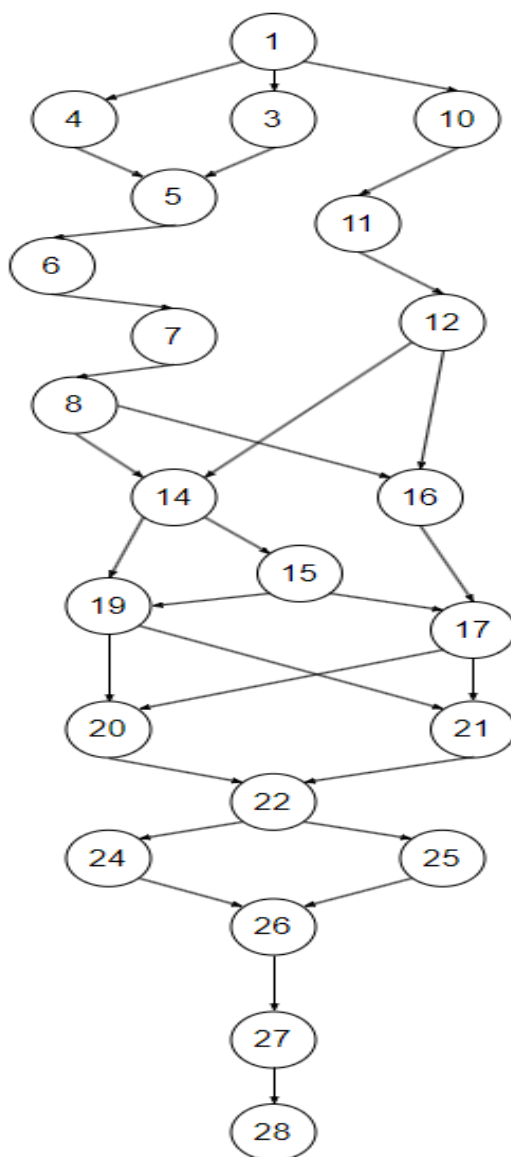


Рисунок 16 – Сетевой график проекта

Рассчитаем вручную критический путь проекта. В таблице 5 представлены расчёты раннего времени начала работ для его задач.

Таблица 5 – Расчёты раннего времени начала работ

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
1	—	—	—	—	0
3	1	0	0	$0+0$	0
4	1	0	0	$0+0$	0
5	3	0	2	$\text{Max}(0+2; 0+4)$	4
	4	0	4		
6	5	4	1	$4+1$	5
7	6	5	1	$5+1$	6
8	7	6	2	$6+2$	8

Продолжение таблицы 5

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
10	1	0	0	0+0	0
11	10	0	2	0+2	2
12	11	2	1	2+1	3
14	8	8	0	Max(8+0;3+0)	8
	12	3	0		
15	14	8	6	8+6	14
16	8	8	0	Max(8+0;3+0)	8
	12	3	0		
17	16	8	5	Max(8+5;14+4)	18
	15	14	4		
19	14	8	6	Max(8+6;14+4)	18
	15	14	4		
20	17	18	0	Max(18+0;18+8)	26
	19	18	8		
21	17	18	0	Max(18+0;18+8)	26
	19	18	8		
22	20	26	21	Max(26+21;26+14)	47
	21	26	14		
24	22	47	0	47+0	47
25	22	47	0	47+0	47
26	24	47	3	Max(47+3;47+3)	50
	25	47	3		
27	26	50	0	50+0	50
28	27	50	7	50+7	57

В таблице 6 рассчитано позднее время начала работ задач.

Таблица 6 – Расчёты позднего времени начала работ

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
28	—	—	—	—	57
27	28	57	0	57-0	57
26	27	57	7	57-7	50
25	26	50	0	50-0	50
24	26	50	0	50-0	50
22	24	50	3	Min(50-3; 50-3)	47
	25	50	3		
21	22	47	0	47-0	47
20	22	47	0	47-0	47
19	20	47	21	Min(47-21;47-14)	26
	21	47	14		
17	20	47	21	Min(47-21;47-14)	26
	21	47	14		
16	17	26	0	26-0	26

Продолжение таблицы 6

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
15	17 19	26 26	0 8	Min(26-0;26-8)	18
14	15 19	18 26	4 8	Min(18-4;26-8)	14
12	14 16	14 26	6 5	Min(14-6;26-5)	8
11	12	8	0	8-0	8
10	11	8	1	8-1	7
8	14 16	14 26	6 5	Min(14-6;26-5)	8
7	8	8	0	8-0	8
6	7	8	2	8-2	6
5	6	6	1	6-1	5
4	5	5	1	5-1	4
3	5	5	1	5-1	4
1	3 4 10	4 4 7	2 4 2	Min(4-2;4-4;7-2)	0

На основе найденных значений раннего и позднего времени начала работ можно найти критический путь. Соответствующие вычисления представлены в таблице 7.

Таблица 7 – Вычисление критического пути

Работа №	Раннее время начала	Позднее время начала	Резерв времени
1	0	0	0
3	0	4	4
4	0	4	4
5	4	5	1
6	5	6	1
7	6	8	2
8	8	8	0
10	0	7	7
11	2	8	6
12	3	8	5
14	8	14	6
15	14	18	4
16	8	26	18
17	18	26	8
19	18	26	8
20	26	47	21
21	26	47	21

Продолжение таблицы 7

Работа №	Раннее время начала	Позднее время начала	Резерв времени
22	47	47	0
24	47	50	3
25	47	50	3
26	50	50	0
27	50	57	7
28	57	57	0

На основе полученных можно сделать вывод, что критический путь проходит через работы с номерами 1, 3, 5, 6, 7, 8, 14, 15, 19, 20, 22, 24, 26, 27, 28, а общее время выполнения проекта составляет 57 дней.

На рисунке 17 представлен граф проекта, на котором штриховыми линиями указан критический путь.

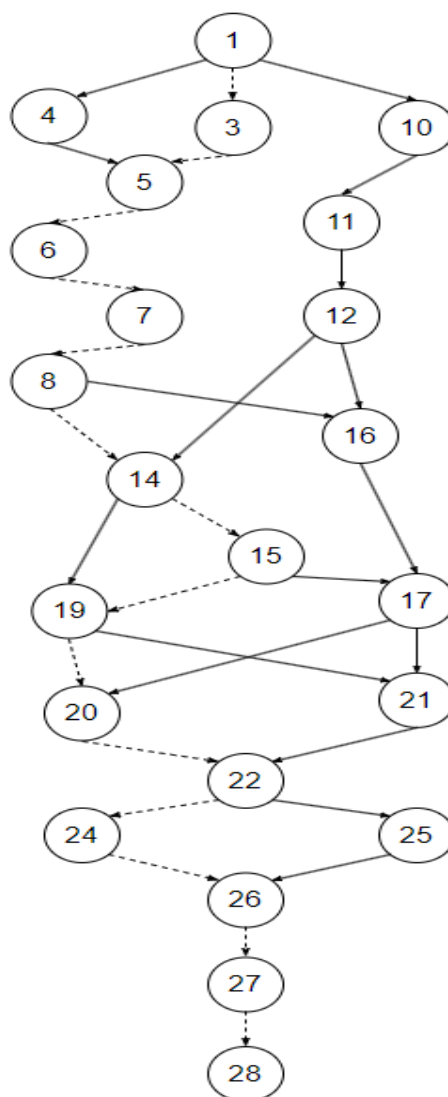


Рисунок 17 – Критический путь проекта

Для того, чтобы проверить корректность расчётов продолжительности проекта, а также вычислить величину его бюджета, воспользуемся системой управления проектами Microsoft Project, которая позволяет автоматически произвести все необходимые вычисления [15 – 20].

Список задач из таблицы 3, внесённый в систему Microsoft Project, представлен на рисунке 18. В качестве времени начала проекта была принята дата 26.02.2025.

		Название задачи	Длительность
1		Начало проекта	0 дней
2		Анализ требований	8 дней
3		Анализ предметной области и постановка задачи	2 дней
4		Анализ программных продуктов аналогичного назначения	4 дней
5		Анализ функциональных требований	1 день
6		Анализ требований к интерфейсу	1 день
7		Анализ целевой аудитории	2 дней
8		Анализ требований завершен	0 дней
9		Выбор технологий разработки ПО	3 дней
10		Анализ существующих технологий разработки ПО	2 дней
11		Выбор технологий разработки ПО	1 день
12		Выбор средств разработки ПО завершен	0 дней
13		Проектирование программных средств	10 дней
14		Проектирование базы данных	6 дней
15		Проектирование архитектуры приложения	4 дней
16		Проектирование интерфейса пользователя	5 дней
17		Проектирование приложения завершенно	0 дней
18		Реализация программных средств	29 дней
19		Реализация базы данных	8 дней
20		Реализация серверной части приложения	21 дней
21		Реализация мобильного приложения	14 дней
22		Реализация завершена	0 дней
23		Тестирование	3 дней
24		Функциональное тестирование	3 дней
25		Модульное тестирование	3 дней
26		Тестирование завершенно	0 дней
27		Составление и оформление документации	7 дней
28		Конец проекта	0 дней

Рисунок 18 – Задачи проекта в среде Microsoft Project

В среде Microsoft Project на основе введённых данных была построена диаграмма Ганта. На её основании можно понять, что критический путь проходит через задачи 1, 3, 5, 6, 7, 8, 14, 15, 19, 20, 22, 24, 26, 27, 28. Можно

заметить, что вычисленный программой критический путь совпал с критическим путём, вычисленным вручную.

Диаграмма Ганта представлена на рисунке 19.



Рисунок 19 – Диаграмма Ганта

На рисунке 20 представлен график трудового ресурса «Программист1».

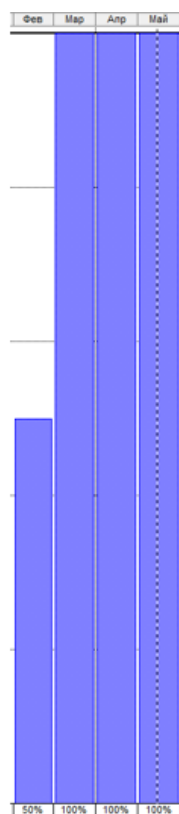


Рисунок 20 – График ресурсов для ресурса «Программист1»

На рисунках 21, 22 представлены графики трудовых ресурсов «Программист2», «Программист3».

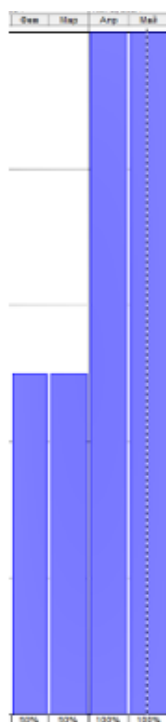


Рисунок 21 – График ресурсов для ресурса «Программист2»

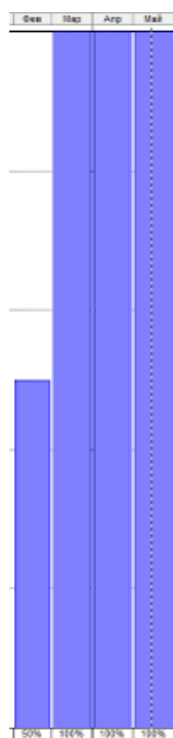


Рисунок 22 – График ресурсов для ресурса «Программист3»

На рисунке 23 представлен график трудового ресурса «Дизайнер».

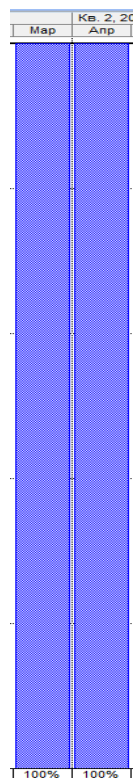


Рисунок 23 – График ресурсов для ресурса «Дизайнер»

На рисунке 24 представлен график материального ресурса «Компьютер».



Рисунок 24 – График ресурсов для ресурса «Компьютер»

На рисунке 25 представлен график материального ресурса «Мобильное устройство».

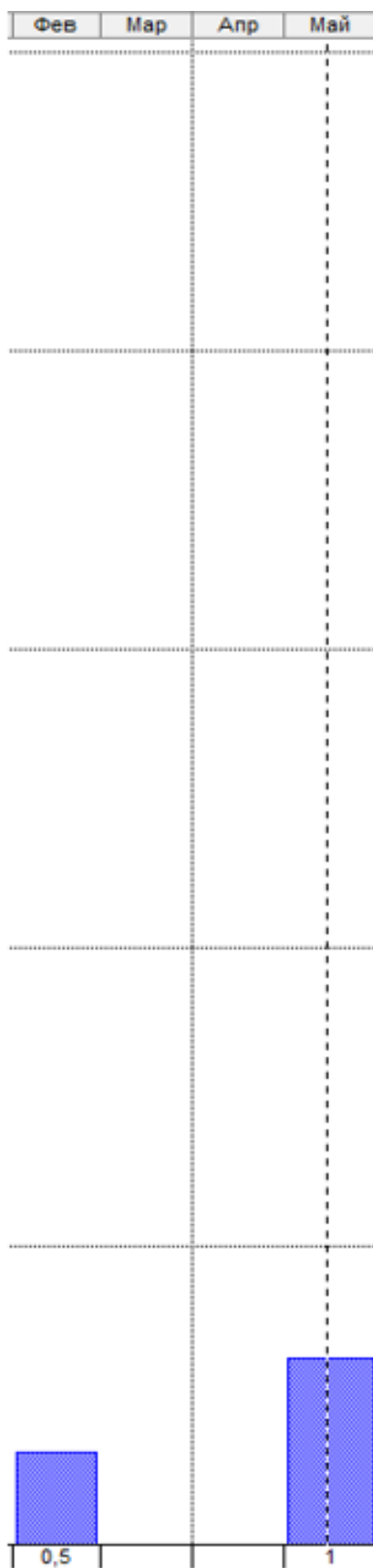


Рисунок 25 – График ресурсов для ресурса «Мобильное устройство»

На рисунке 26 представлен обзор проекта. Из рисунка видно, что общая длительность проекта составляет 57 дней (что совпадает со значением, вычисленным вручную), а предполагаемые затраты – 429 000 руб.

Проект.mpr			
на дату СБ 26.02.25			
Даты			
Начало:	Пн 26.02.25	Окончание:	Вт 14.05.25
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
Длительность			
Текущий план:	57 дней	Оставшаяся:	57 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	0 дней	Процент завершения:	0%
Трудозатраты			
Текущий план:	1 060 ч	Оставшиеся:	1 060 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	0 ч	Процент завершения:	0%
Затраты			
Текущий план:	429 000,00 Р	Оставшиеся:	429 000,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	0,00 Р		
Состояние задачи		Состояние ресурса	
Еще не начатые задачи	28	Трудовые ресурсы:	5
Выполняющиеся задачи	0	Трудовые ресурсы с прев. доп.	0
Завершенные задачи:	0	Материальные ресурсы:	2
Итого задач:	28	Итого ресурсов:	7

Рисунок 26 – Обзорный отчет проекта

На основе произведённых оценок стоимости и сроков проекта можно сделать вывод о его экономической целесообразности.

2 Проектирование мобильного приложения

2.1 Выбор технологий и инструментальных средств разработки

В процессе разработки приложения было решено использовать методологию RUP. Разработка приложения в соответствии с методологией RUP носит итеративный характер. На каждой итерации перед разработчиками ставятся определённые цели, которые должны быть достигнуты к её концу. В результате в конце каждой итерации должна получиться промежуточная функциональная версия разрабатываемого ПО. В методологии RUP существуют 4 основных фазы, каждая из которых может включать в себя одну или несколько итераций [21]:

- фаза начала проекта – формируется общее видение проекта и экономическое обоснование;
- уточнение – анализ предметной области и построение архитектуры продукта;
- построение – реализация основного функционала продукта;
- внедрение – получение финальной версии продукта и его передача заказчику, фаза включает в себя бета-тестирование и устранение выявленных в его процессе ошибок, а также обучение пользователей.

Каждая фаза завершается вехой, на которой команда разработчиков оценивает выполнение поставленных целей и принимает критически важные решения о ходе дальнейшей работы над продуктом.

Выбор в пользу данной методологии продиктован рядом её достоинств. С одной стороны, методология RUP обеспечивает гибкость процесса разработки и позволяет быстро вносить коррективы в проект в случае выявления в нем ошибок или изменения требований к разрабатываемому приложению. Кроме того, благодаря данной методологии удаётся выявить основные риски уже на ранних стадиях разработки. При этом RUP, в отличие от некоторых других популярных методологий, например XP, не требует постоянного взаимодействия с заказчиком, которое в рамках данного проекта невозможно.

Для анализа требований, а также моделирования структуры проекта в соответствии с принципами RUP было решено использовать диаграммы на языке моделирования UML [22]. Для построения диаграмм использовалось средство UML-моделирования Visual Paradigm.

На этапе планирования проекта было решено использовать приложение Microsoft Project. Оно позволяет обеспечить эффективное управление проектом на всех стадиях его выполнения. Microsoft Project предоставляет возможность выполнить структуризацию проекта, составить его календарный план, а также рассчитать затраты трудовых, материальных и денежных ресурсов в процессе работы над проектом.

Для создания мобильного приложения было решено использовать среду программирования Android Studio, и язык программирования Java, а также современный кроссплатформенный фреймворк .NET MAUI. Серверную часть было решено разрабатывать на языке C# с использованием среды программирования Visual Studio 2022. Для хранения данных используется БД, работающая под управлением СУБД PostgreSQL. Проектирование базы данных выполнялось с использованием приложения Visual Paradigm. Реализация базы данных производилась с использованием языка программирования SQL.

2.2 Проектирование архитектуры программного обеспечения

Было принято решение использовать в приложении двухуровневую архитектуру «клиент-сервер». Её схема представлена на рисунке 27.

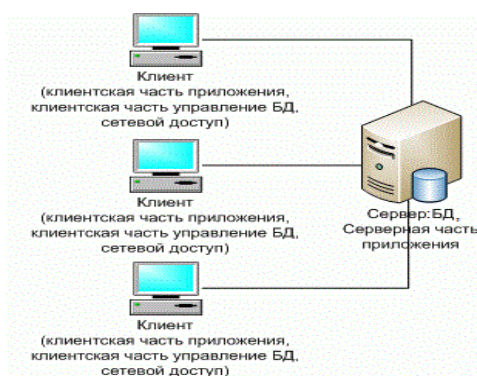


Рисунок 27 – Двухуровневая архитектура «клиент-сервер»

В данной архитектуре на сервере располагаются база данных и СУБД, а также серверная часть приложения, которая отвечает за непосредственное взаимодействие с базой данных и, как правило, реализуется в виде интерфейса прикладного программирования (API).

Клиентская же часть разрабатываемого ПО обеспечивает для пользователя удобный интерфейс, позволяющий пользователю просматривать необходимые ему данные из БД и взаимодействовать с ними, а также реализует возможность удалённо обращаться к серверу. В нашем случае его роль выполняет мобильное приложение.

Взаимодействие клиента и сервера осуществляется следующим образом: клиент делает HTTP-запрос серверу. Серверная часть приложения, в свою очередь, на основе полученного запроса создаёт SQL-запрос к базе данных. СУБД обрабатывает SQL-запрос и возвращает его результаты серверу, который на основе полученных результатов формирует ответ на запрос клиента. Получив ответ, клиент представляет его пользователю в удобной для того форме.

Достоинством архитектуры является то, что все данные хранятся на стороне сервера. Это позволяет минимизировать объем клиентского приложения. Кроме того, поскольку серверная часть приложения не зависит от клиентской, можно создать несколько вариантов клиента, отличающихся друг от друга, что может быть выгодно в случае потенциального распространения приложения и на другие платформы, помимо Android.

2.3 Проектирование интерфейса пользователя

На основе выявленных функциональных требований был спроектирован интерфейс мобильного приложения, который отображает удобный пользовательский интерфейс работы косметического салона. Он включает в себя страницы входа, регистрации, главную страницу приложения, просмотр мастеров, просмотр услуг, просмотр портфолио, запись на процедуру.

Страница входа представлена на рисунке 28.

The diagram illustrates the layout of a login page. It consists of a large rectangular container. Inside, there are four main elements arranged vertically: 1. A rectangular input field for the username. 2. A rectangular input field for the password. 3. A rectangular button labeled "Войти" (Login). 4. A link labeled "Еще не зарегистрировались?" (Not registered yet?) followed by a rectangular button labeled "Регистрация" (Registration). To the right of the container, four horizontal lines with numbers 1, 2, 3, and 4 point to the respective elements: the first input field, the second input field, the "Войти" button, and the "Регистрация" button.

Рисунок 28 – Страница входа

Рассмотрим основные элементы интерфейса страницы входа: 1 – поле ввода логина пользователя, 2 – поле ввода пароля пользователя, 3 – кнопка «Вход», 4 – кнопка перехода на страницу регистрации. При нажатии на кнопку «Вход» система проверяет, существует ли на самом деле пользователь с таким логином и паролем. Если такого пользователя нет, будет выведено сообщение о неверном логине или пароле. Иначе авторизация будет считаться выполненной успешно.

Страница регистрации представлена на рисунке 29.

The diagram shows a registration page layout within a rectangular frame. It contains four main components, each with a number and a leader line pointing to it from the right:

- 1: A rectangular input field for the username.
- 2: A rectangular input field for the password.
- 3: A rectangular button labeled "Регистрация" (Registration).
- 4: A rectangular button labeled "Войти" (Login).

Between the registration button and the login button, the text "Уже зарегистрировались?" (Already registered?) is displayed.

Рисунок 29 – Страница регистрации

Рассмотрим основные элементы интерфейса страницы регистрации:

1 – поле ввода логина пользователя, 2 – поле ввода пароля пользователя, 3 – кнопка «Регистрация», 4 – кнопка перехода на страницу входа. При нажатии кнопки «Регистрация» система проверяет, являются ли введенные логин и пароль допустимыми. Если это условие не выполняется, будет выведено сообщение о недопустимом логине или пароле. В противном случае будет проверено, не занят ли данный логин. Если он занят, выводится соответствующее сообщение. Иначе авторизация будет считаться выполненной успешно.

Главная страница приложения представлена на рисунке 30.

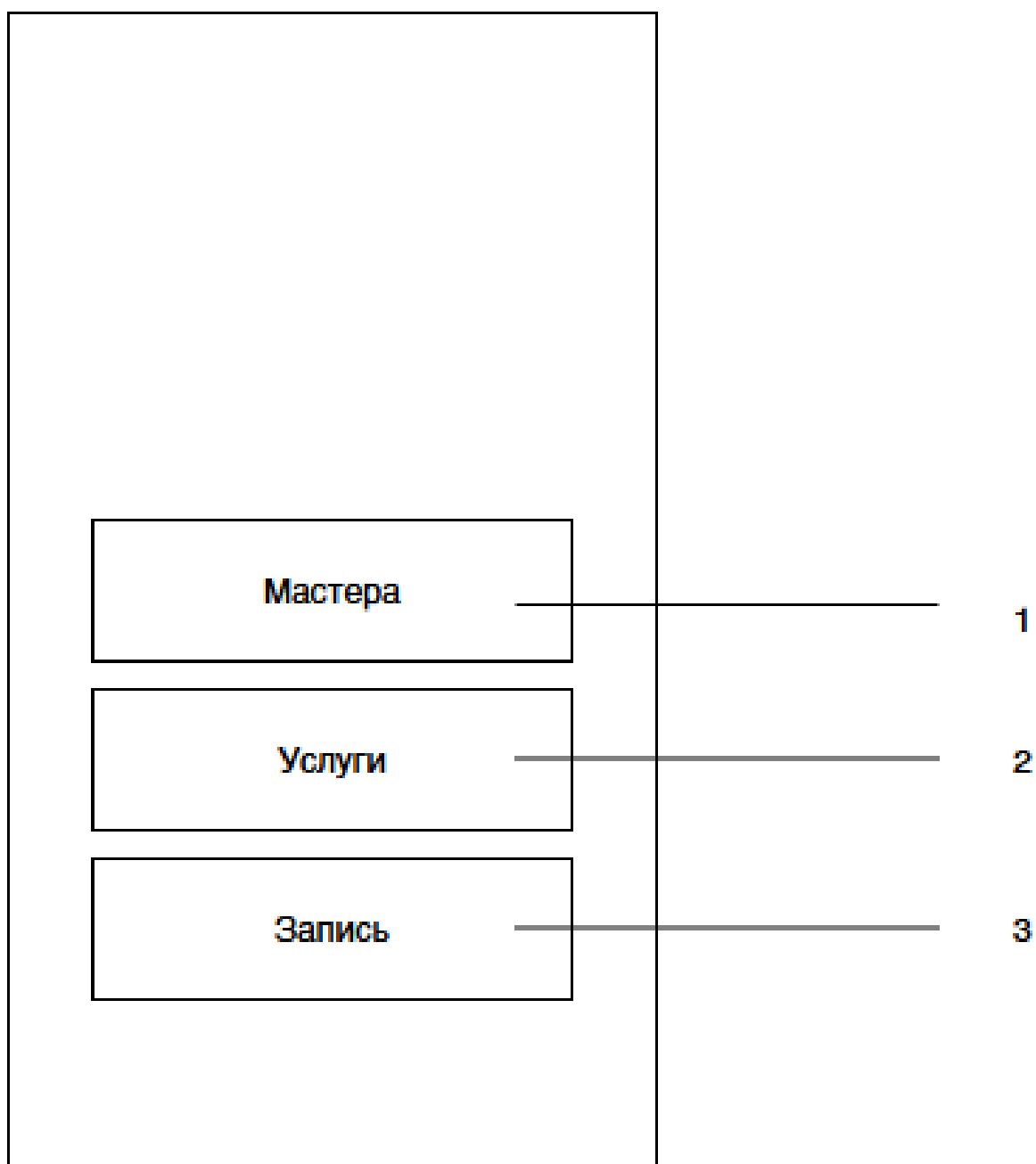


Рисунок 30 – Главная страница приложения

Рассмотрим основные элементы интерфейса главной страницы приложения: 1 – кнопка перехода на страницу мастеров, 2 – кнопка перехода на страницу услуг, 3 – кнопка перехода на страницу записи на процедуру.

Страница записи на процедуру представлена на рисунке 31.

The image shows a rectangular frame representing a web page. Inside the frame, there are three main components arranged vertically. The top component is a horizontal rectangle divided into two parts: a large empty box on the left and a smaller box on the right containing a downward-pointing chevron (dropdown arrow). A line points from the number '1' to this dropdown arrow. The middle component is identical to the top one, also with a line pointing from the number '2' to its dropdown arrow. The bottom component is a single horizontal rectangle containing the word 'Записаться' (Book) in a bold, sans-serif font. A line points from the number '3' to this button.

Рисунок 31 – Страница записи на процедуру

Рассмотрим основные элементы интерфейса страницы записи на процедуру: 1 – выпадающий список для выбора мастера, 2 – выпадающий список для выбора услуги, 3 – кнопка для выбора времени и записи. При нажатии кнопки «Записаться» система проверяет, свободно ли выбранное время для записи. Если условие не выполняется, будет выведено сообщение о невозможности записи. В противном случае запись будет создана.

Страница просмотра мастеров представлена на рисунке 32.

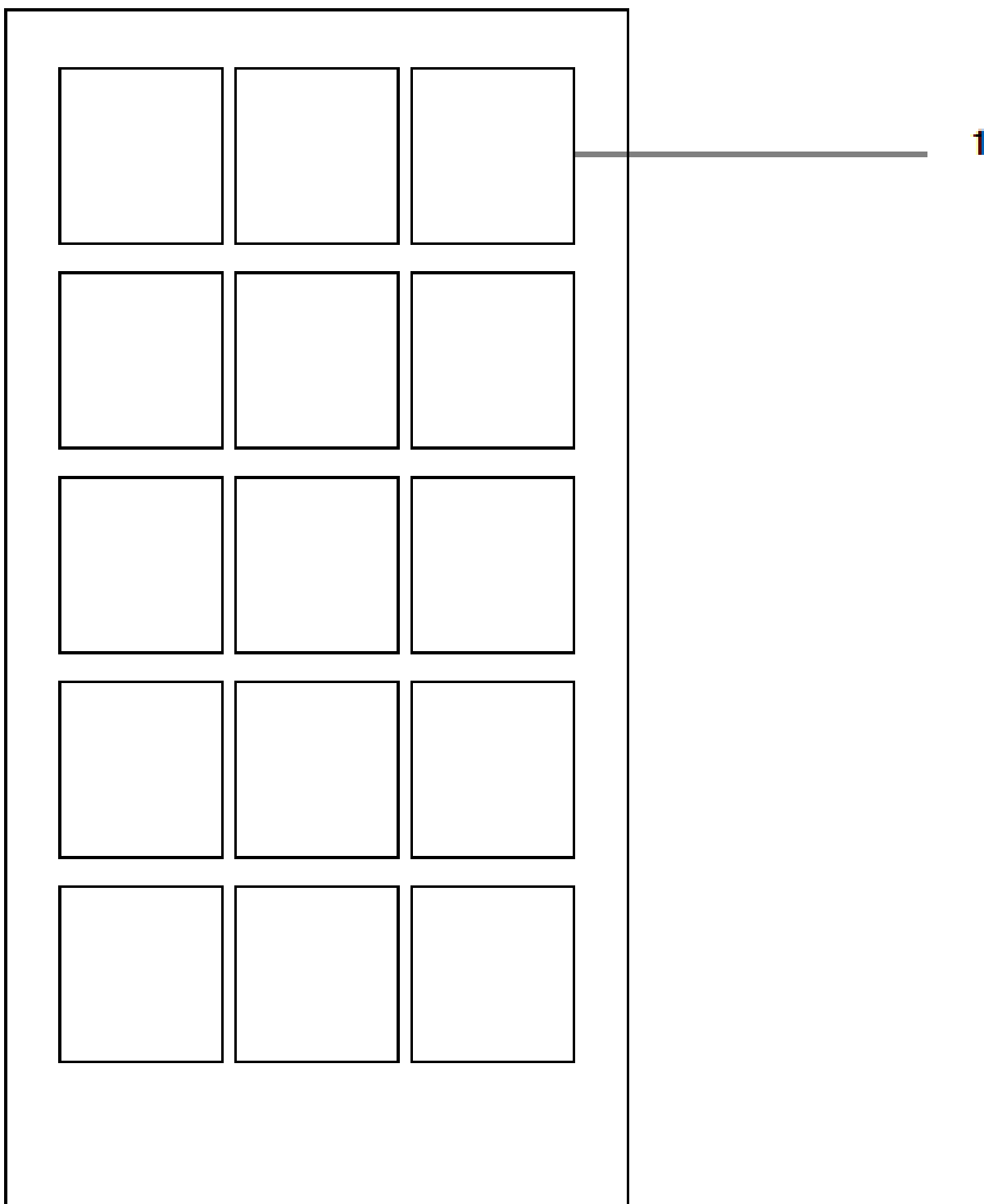


Рисунок 32 – Страница списка мастеров

Рассмотрим основные элементы интерфейса страницы просмотра списка мастеров: 1 – карточка мастера.

Страница с списком услуг представлена на рисунке 33.

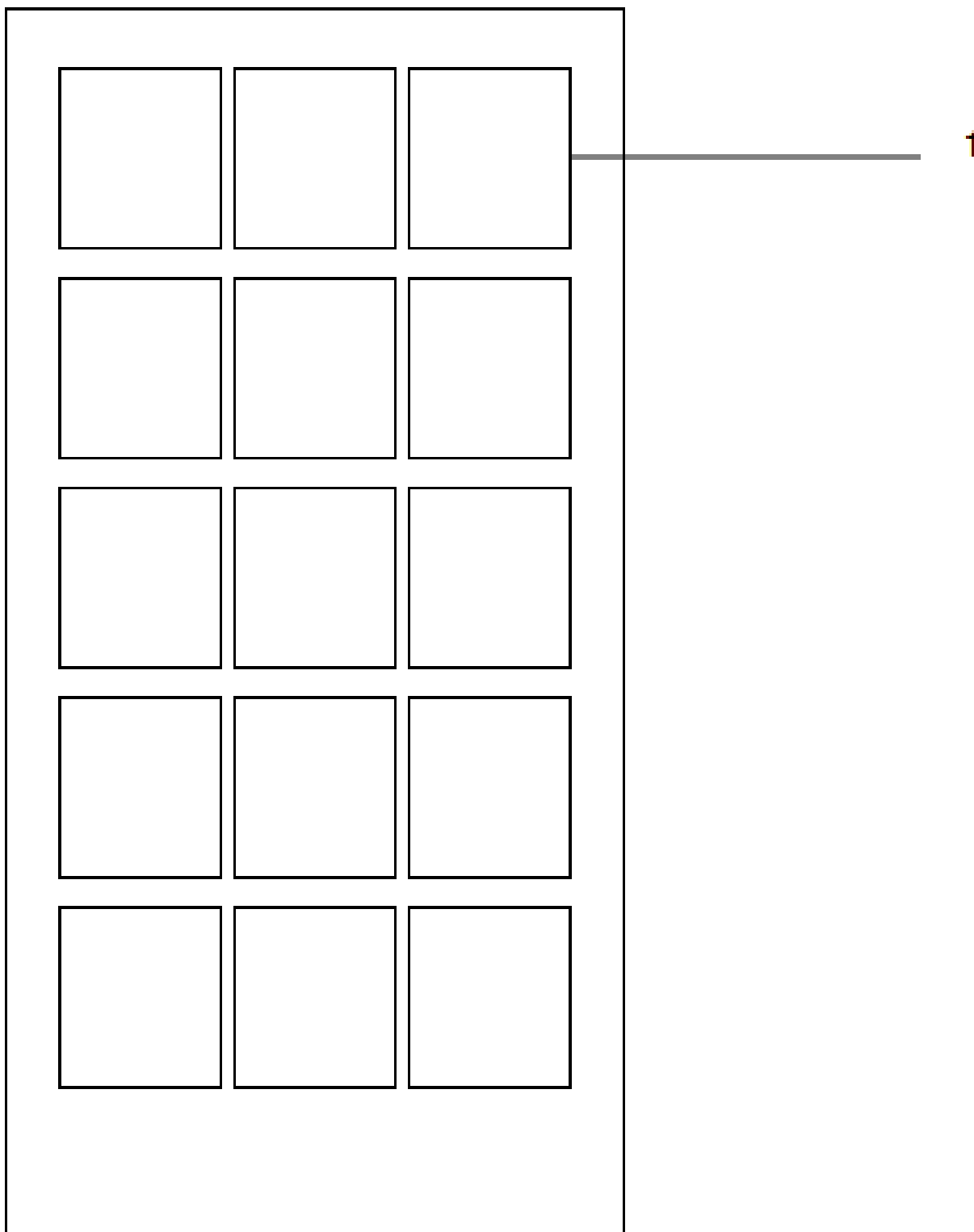


Рисунок 33 – Страница списка услуг

Рассмотрим основные элементы интерфейса страницы просмотра списка услуг: 1 – карточка услуги.

Страница списка портфолио выбранного мастера представлена на рисунке 34.

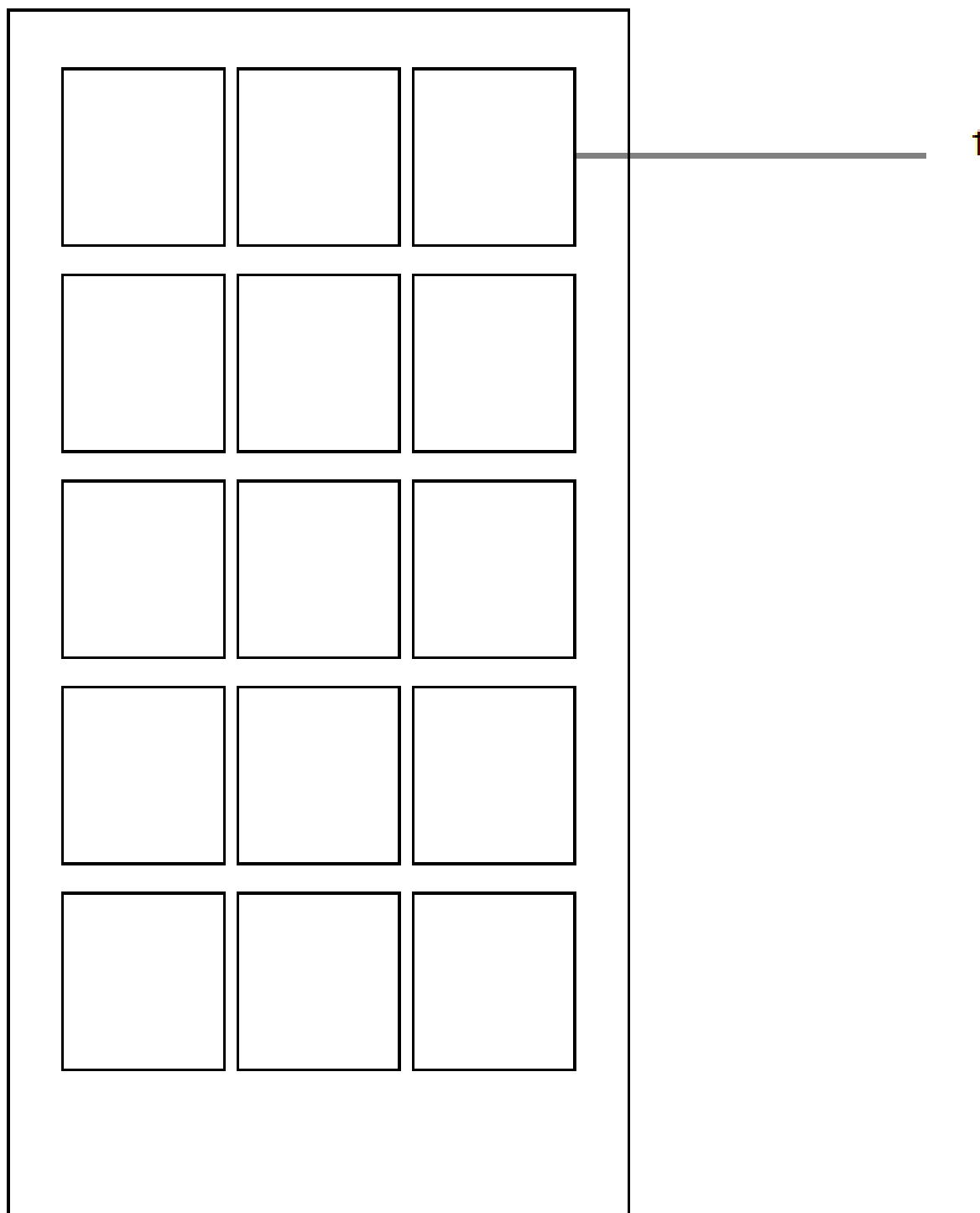


Рисунок 34 – Страница списка портфолио выбранного мастера

Рассмотрим основные элементы интерфейса страницы списка портфолио выбранного мастера: 1 – список выполненных и опубликованных мастером работ.

2.4 Логическое проектирование базы данных

На основе анализа предметной области и функциональных требований была построена концептуальная модель базы данных [23]. Она представлена в виде диаграммы «сущность-связь» (рисунок 35).

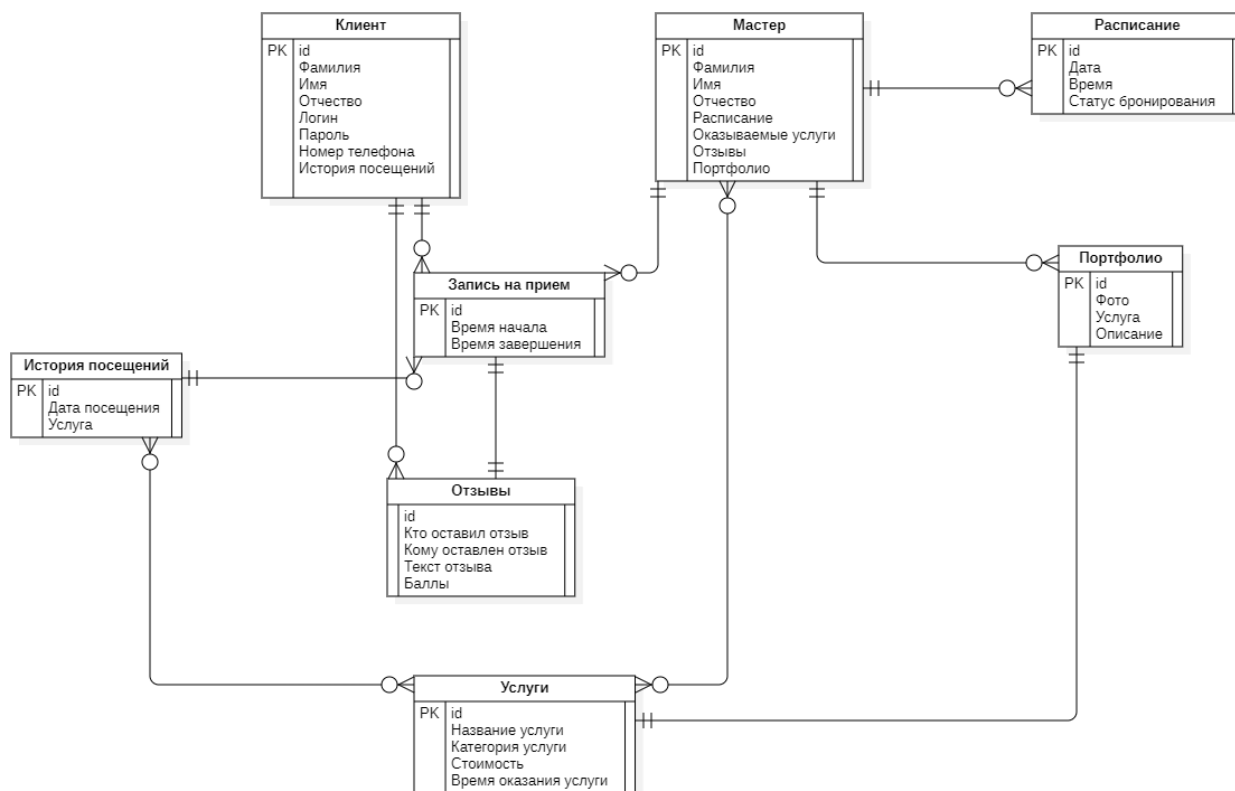


Рисунок 35 – Диаграмма «сущность-связь»

Сущности концептуальной модели данных описаны в таблице 8.

Таблица 8 – Сущности предметной области

№ п/п	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Клиент	Список зарегистрированных клиентов	Стержневая	200	90% в 2 месяца
2	Мастер	Список специалистов, оказывающих услуги	Стержневая	20	50% в 3 месяца
3	Услуги	Список доступных услуг	Стержневая	10	Нет
4	Расписание	График доступности мастеров	Ассоциативная	500	100% в неделю

Продолжение таблицы 8

№ п/п	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
5	Запись на прием	Записи клиентов на прием к мастерам	Ассоциативная	300	80% в неделю
6	История посещений	Архив завершенных визитов	Ассоциативная	1500	100% в 2 месяца
7	Отзывы	Отзывы и комментарии клиентов	Ассоциативная	500	Нет
8	Портфолио	Примеры выполненных работ мастера	Ассоциативная	200	30% в 3 месяца

В таблице 9 представлено описание связей между сущностями предметной области.

Таблица 9 – Описание связей между сущностями предметной области

№ п/п	Сущность 1	Сущность 2	Семантика связи	Мощность	Обязательность	Ограничения
1	Клиент	Запись на прием	Клиент может записываться на прием	1:M	Необязательно для клиента – клиент может не записываться Обязательно для записи – должна быть привязка к клиенту	Клиент должен существовать
2	Клиент	История посещений	У клиента есть история посещений	1:M	Необязательно для клиента – может не быть посещений Обязательно для записи в истории – должен быть клиент	Клиент должен существовать
3	Мастер	Расписание	У каждого мастера есть расписание Расписание относится к конкретному мастеру	1:M	Обязательно для инструктора – у мастера только одно расписание Обязательно для расписания – расписание используется только одним мастером	Клиент должен существовать
4	Мастер	Услуги	Мастер оказывает определенные услуги	1:M	Обязательно для мастера и услуги – у каждого расписания должен быть мастер	Один мастер может иметь много записей в расписании

Продолжение таблицы 9

№ п/п	Сущность 1	Сущность 2	Семантика связи	Мощность	Обязательность	Ограничения
5	Мастер	Отзывы	Мастеру могут оставлять отзывы	1:M	Обязательно для отзыва – должен быть адресован мастеру	Мастер должен существовать
6	Мастер	Портфолио	Портфолио принадлежит конкретному мастеру	1:M	Обязательно для элемента портфолио – должен быть привязан к мастеру	Мастер должен существовать
7	Клиент	Отзывы	Клиент может оставлять отзывы	1:M	Обязательно для отзыва – должен быть клиент, оставивший отзыв	Клиент должен существовать
8	Запись на прием	Расписание	Запись осуществляется в конкретный слот расписания	M:1	Обязательно для записи – должна быть привязка к расписанию	Расписание должно существовать
9	История посещений	Услуги	В истории указывается, какая услуга была оказана	M:1	Обязательно для истории – должна быть оказана услуга	Услуга должна существовать
10	Портфолио	Услуги	В портфолио указано, к какой услуге относится работа	M:1	Обязательно для портфолио - должна быть оказана услуга	Услуга

На основании концептуальной модели данных в среде Visual Paradigm была построена логическая модель базы данных.

Её структура описана диаграммой в нотации IDEF1X, представленной на рисунке 36.

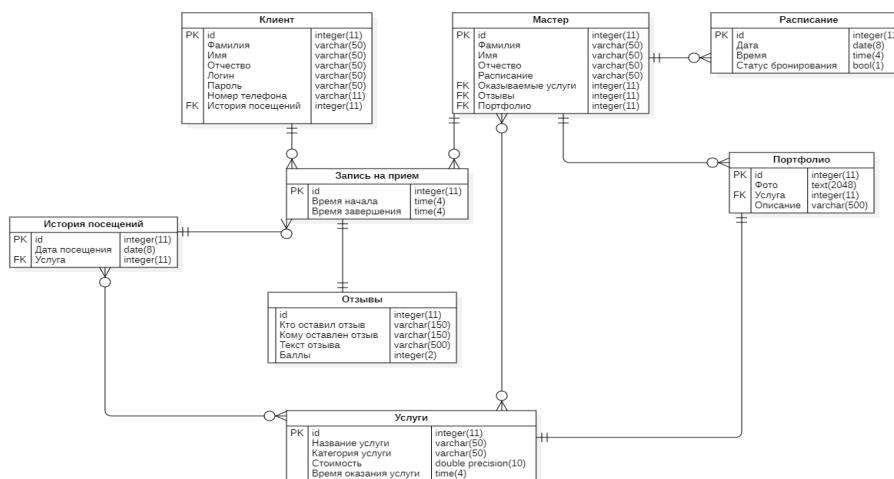


Рисунок 36 – Логическая модель базы данных

На представленной логической модели спроектированная структура БД отображена уже более подробно. Здесь не только присутствуют типы сущностей и связи между ними, но и хранится информация о конкретных атрибутах сущностей [24]. Однако на данный момент сущности все ещё представлены в абстрактной форме, без привязки к конкретной реализации базы данных.

2.5 Проектирование серверной части приложения

Серверная часть приложения представляет собой REST API. Доступ к данным осуществляется при помощи протокола HTTP с использованием его стандартных методов (GET применяется для получения данных ресурса, POST – для создания нового ресурса, PUT – для обновления существующего, DELETE – для удаления ресурса). Обращение к ресурсам происходит при помощи уникальных идентификаторов ресурсов (URI).

В приложение используется получившая широкое распространение архитектура MVC (Model-View-Controller) [25]. Её схема представлена на рисунке 37.

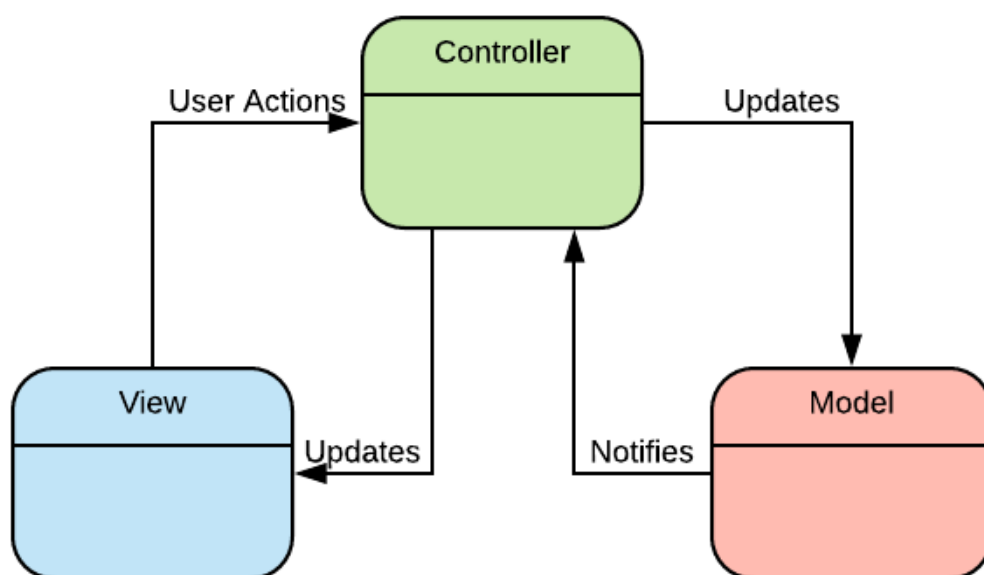


Рисунок 37 – Архитектура MVC

Серверная часть приложения включает в себя классы модели, отвечающие за организацию доступа к данным (т.е. за непосредственное взаимодействие с БД) и классы контроллера, отвечающие за организацию

взаимодействия пользователя с клиентом. Именно они обрабатывают HTTP-запросы клиента и формируют ответы на них. К клиенту же отойдёт только уровень представления, отвечающий за взаимодействие с пользователем.

использовалась технология Entity Framework [30]. Диаграмма классов серверной части представлена на рисунке 41.

В соответствии с используемой архитектурой MVC все классы серверной части можно разделить на два слоя: слой модели, предназначенный для взаимодействия с БД, и слой контроллеров, предназначенный для обработки HTTP-запросов к серверу.

Диаграмма классов слоя модели представлена на рисунке 39. Все эти классы были автоматически сгенерированы средой программирования Visual Studio с использованием технологии Entity Framework. Генерация производилась на основе предварительно разработанной до этого базы данных (подход Database First) [31, 32].

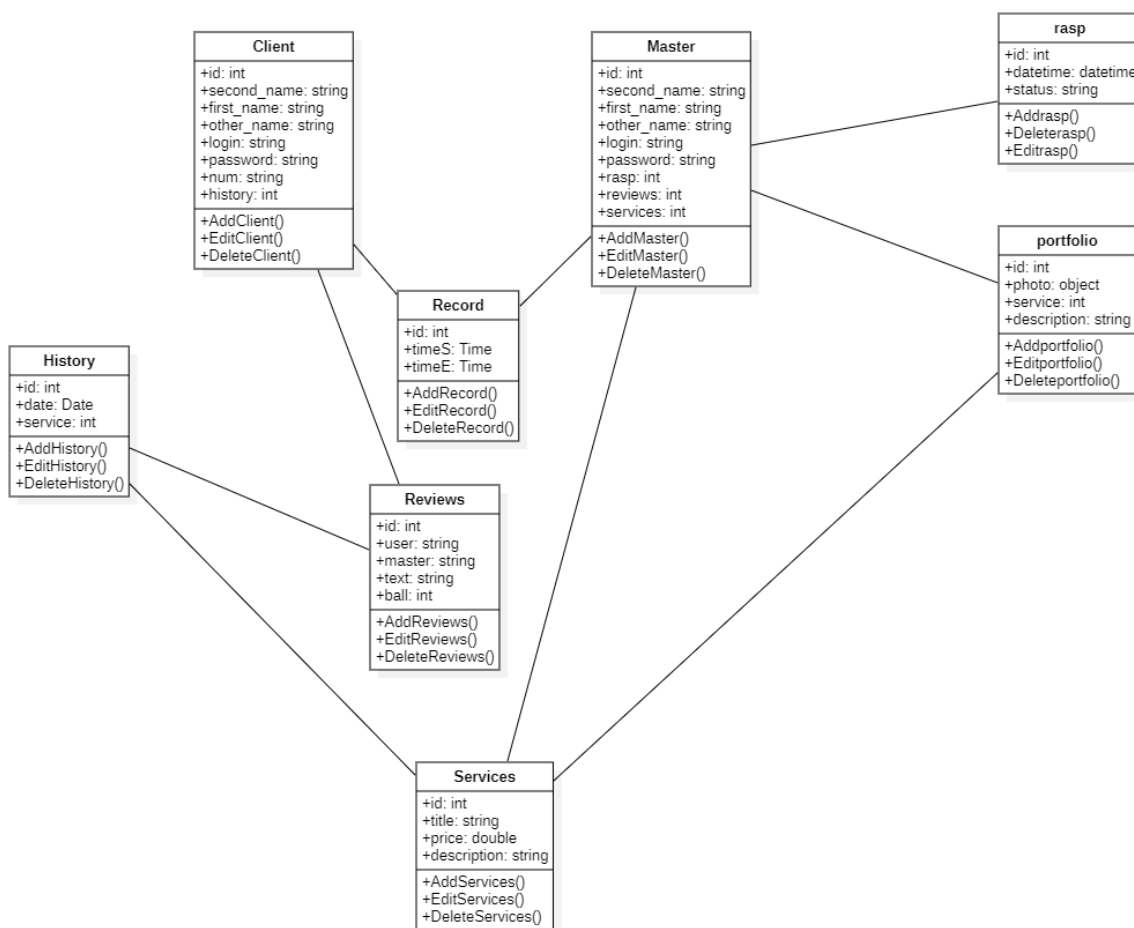


Рисунок 39 – Диаграмма классов серверной части (слой модели)

Как можно заметить, классы Client, Master, Record, History, Reviews, Services, Rasp, Portfolio были созданы на основании одноименных таблиц базы данных.

Диаграмма классов слоя контроллеров представлена на рисунке 40.

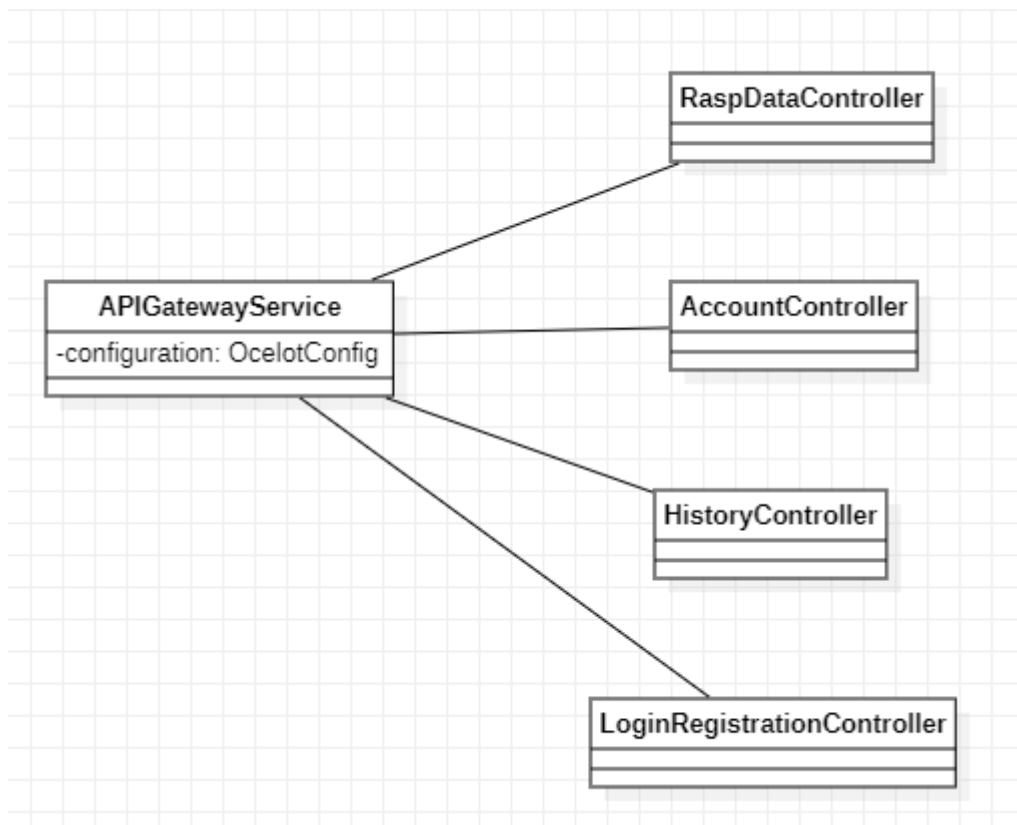


Рисунок 40 – Диаграмма классов серверной части (слой контроллеров)

Более подробное описание классов слоя контроллеров и их назначений представлено в таблице 10.

Таблица 10 – Описание классов слоя контроллеров

Класс	Назначение
APIGatewayService	Отвечает за обработку запросов по API
RaspDataController	Отвечает за обработку запросов записи
AccountController	Отвечает за обработку запросов связанных с взаимодействием с аккаунтом
HistoryController	Отвечает на обработку запросов на сохранение истории записи
LoginRegistrtrtionC ontroller	Отвечает за обработку запросов связанных с авторизацией и регистрацией

Листинг серверной части представлен в приложении В.

3.3 Реализация клиентской части приложения

Разработка мобильного приложения производилась на языке Java с использованием среды программирования Android Studio. Диаграмма компонентов, отражающая структуру программы, представлена на рисунке 41.

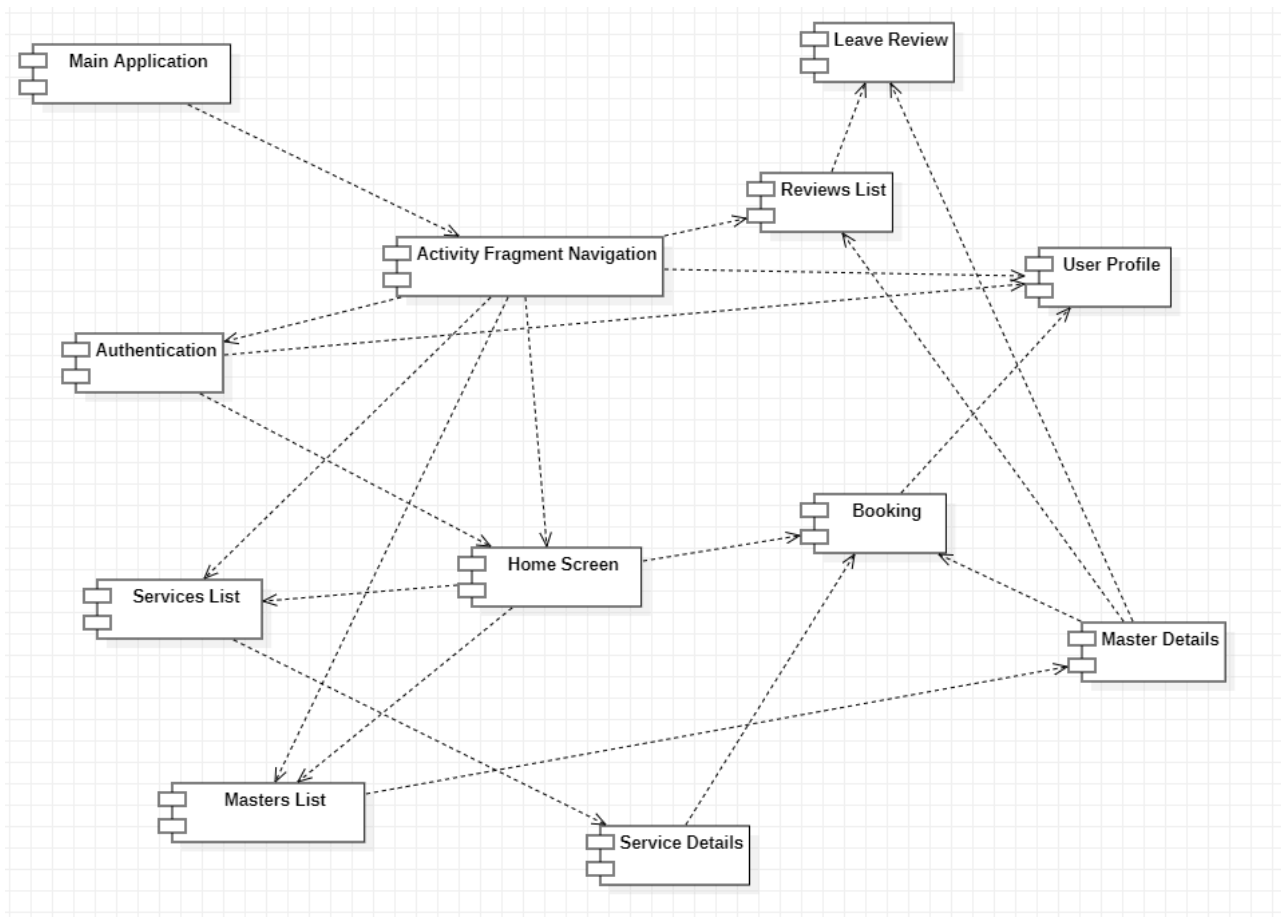


Рисунок 41 – Диаграмма компонентов мобильного приложения

Описание компонентов представлено в таблице 11.

Таблица 11 – Компоненты клиентского приложения

Компонент	Назначение
AndroidManifest.xml	XML-файл манифеста, описывающий структуру, разрешения и компоненты приложения
Application Class	Пользовательский класс Application для инициализации глобальных ресурсов
MainActivity.java	Основная Activity приложения, которая может содержать навигацию между фрагментами

Продолжение таблицы 11

Компонент	Назначение
Navigation Components	Фреймворк для управления навигацией между экранами/фрагментами приложения
SignInActivity.java	Java-класс Activity, реализующий интерфейс страницы авторизации
SignInViewModel.java	Класс ViewModel для управления логикой и данными страницы авторизации, взаимодействует с API
SignUpActivity.java	Java-класс Activity, реализующий интерфейс страницы регистрации нового пользователя
SignUpViewModel.java	Класс ViewModel для управления логикой и данными страницы регистрации
ForgotPasswordActivity.java	Java-класс Activity, реализующий интерфейс страницы восстановления пароля
ForgotPasswordViewModel.java	Класс ViewModel для управления логикой и данными страницы восстановления пароля
HomeFragment.java	Java-класс Fragment, реализующий интерфейс домашней страницы с обзором
HomeViewModel.java	Класс ViewModel для управления логикой и данными домашней страницы (например, отображение акций)
ServicesListFragment.java	Java-класс Fragment, реализующий интерфейс страницы со списком всех услуг
ServicesListViewModel.java	Класс ViewModel для управления логикой и данными списка услуг, фильтрацией
ServiceDetailsFragment.java	Java-класс Fragment, реализующий интерфейс страницы с детальной информацией об услуге
ServiceDetailsViewModel.java	Класс ViewModel для управления логикой и данными детальной страницы услуги
MastersListFragment.java	Java-класс Fragment, реализующий интерфейс страницы со списком всех мастеров
MastersListViewModel.java	Класс ViewModel для управления логикой и данными списка мастеров, фильтрацией
MasterDetailsFragment.java	Java-класс Fragment, реализующий интерфейс страницы с детальной информацией о мастере
MasterDetailsViewModel.java	Класс ViewModel для управления логикой и данными детальной страницы мастера

Окончание таблицы 11

Компонент	Назначение
BookingFragment.java	Java-класс Fragment, реализующий интерфейс страницы для выбора даты, времени и записи
BookingViewModel.java	Класс ViewModel для управления логикой и данными процесса записи, доступностью слотов
ReviewsListFragment.java	Java-класс Fragment, реализующий интерфейс страницы со списком отзывов
ReviewsListViewModel.java	Класс ViewModel для управления логикой и данными списка отзывов
LeaveReviewFragment.java	Java-класс Fragment, реализующий интерфейс страницы для оставления нового отзыва
LeaveReviewViewModel.java	Класс ViewModel для управления логикой и данными формы отзыва, отправкой на сервер
UserProfileFragment.java	Java-класс Fragment, реализующий интерфейс страницы профиля пользователя
UserProfileViewModel.java	Класс ViewModel для управления логикой и данными профиля пользователя, обновлением информации

Диаграмма развертывания, на которой описывается физическое расположение компонентов системы, представлена на рисунке 42.

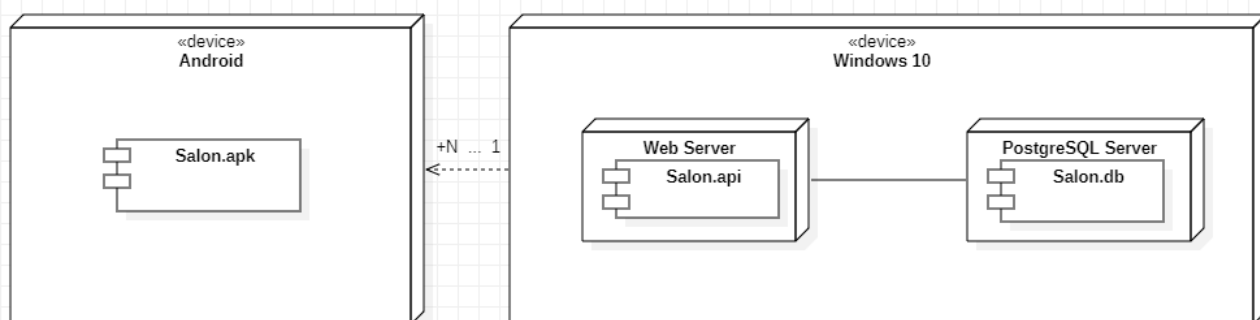


Рисунок 42 – Диаграмма развертывания мобильного приложения

Как можно увидеть на этой диаграмме, клиентское приложение работает на мобильном устройстве под управлением системы Android. API и база данных располагаются на сервере, работающем под управлением ОС Windows.

Взаимодействие между серверной и клиентской частями приложения происходит с использованием протокола HTTP.

Листинг клиентской части программы представлен в приложении Г.

4 Тестирование программного продукта

4.1 Тестирование серверной части приложения

Для начала необходимо убедиться в корректности работы серверной части приложения. С этой целью было произведено тестирование серверной части приложения с использованием инструментального средства Postman. Пример результатов тестирования функционала серверной части представлены в таблице 12. Скриншоты тестов находятся в приложении Д.

Таблица 12 – Пример результатов тестирования серверной части

№	Описание теста	Запрос		Результат
		Метод	Параметры	
1	Добавление в систему пользователя с логином test и паролем test	POST	login = test, password = test	Тест пройден. Пользователь успешно добавлен в систему. Код ответа HTTP – 201 (рисунок Д.1).
2	Попытка входа в систему пользователя с логином test и паролем test	GET	login = test, password = test	Тест пройден. Пользователь успешно авторизован. Код ответа HTTP – 200 (рисунок Д.2).
3	Получение сведений о пользователе с идентификатором 1	GET	id = 1	Тест пройден. Информация о пользователе успешно получена. Код ответа HTTP – 200 (рисунок Д.3).
4	Изменение пароля пользователя с идентификатором 8 с test на test2	PUT	id= 8, oldpassword= test, newpassword= test2	Тест пройден. Пароль успешно изменен. Код ответа HTTP – 200 (рисунок Д.4).
5	Создание записи на процедуру	POST	userId=39, date = 2025-06-06, time =15:30	Тест пройден. Запись успешно создана. Код ответа HTTP – 201 (рисунок Д.5).
6	Отмена записи на процедуру	DELETE	userId=39, date = 2025-06-06, time =15:30	Тест пройден. Запись успешно отменена. Код ответа HTTP – 200 (рисунок Д.6).
7	Сохранение отзыва	POST	userId=39, correct = true, review=»Понравился мастер»	Тест пройден. Ответ успешно сохранен. Код ответа HTTP – 200 (рисунок Д.7).

Все тесты серверной части были пройдены успешно, что свидетельствует о корректности работы серверной части приложения и позволяет перейти к тестированию клиентской части разрабатываемого мобильного приложения.

4.2 Тестирование клиентской части приложения

4.2.1 Функциональное тестирование

Для проверки корректности работы клиентской части приложения было проведено функциональное тестирование. Пример результатов представлен в таблице 13. Скриншоты тестов находятся в приложении Е.

Таблица 13 – Пример функционального тестирования

Вариант использования	Тест и параметры	Результат
Зарегистрироваться в системе	Ввести валидные данные для регистрации. Логин: test2. Пароль: test2.	Тест пройден. Пользователь зарегистрирован в системе (рисунок Е.1).
	Ввести данные с логином существующего в системе пользователя. Логин: test2.	Тест пройден. Выведено уведомление о том, что логин уже занят (рисунок Е.2).
	Попытаться задать логин и пароль с недопустимыми символами. Логин: логин. Пароль: пароль.	Тест пройден. Выведено уведомление о вводе недопустимых данных (рисунок Е.3).
	Попытаться задать логин и пароль недопустимой длины. Логин: m. Пароль: qwe1.	Тест пройден. Выведено уведомление о вводе недопустимых данных (рисунок Е.4).
	Произвести попытку задать пароль без цифр. Логин: mylogin1 Пароль: qwertyuiop	Тест пройден. Выведено уведомление о вводе недопустимых данных (рисунок Е.5).
Войти в систему	Ввести валидные данные. Логин: test2. Пароль: test2..	Тест пройден. Авторизация выполнена успешно (рисунок Е.6).
	Ввести неверные данные. Логин: test2. Пароль: tttt.	Тест пройден. Выведено уведомление о неверном логине или пароле (рисунок Е.7).
	Ввести данные заблокированного пользователя. Логин: blockedLogin. Пароль: test3.	Тест пройден. Выведено сообщение о блокировке (рисунок Е.8).

Продолжение таблицы 13

Вариант использования	Тест и параметры	Результат
Сделать запись на процедуру	Записаться на свободное время к мастеру на услугу.	Тест пройден. Выведено сообщение о том, что запись успешно создана (рисунок Е.9).
	Записаться на забронированное время.	Тест пройден. Выведено сообщение о том, что выбранное время уже занято (рисунок Е.10).
Оставить отзыв	Оставили отзыв «Мастер понравился»	Тест пройден. Отзыв добавился (рисунок Е.11).

Функциональное тестирование показало корректность работы клиентской части мобильного приложения.

4.2.2 Модульное тестирование

Модульные тесты – автоматизированные тесты, позволяющие оценить корректность работы некоторой автономной единицы программного кода (обычно подпрограммы) путем сравнения фактического результата работы с ожидаемым. Основное достоинство модульных тестов состоит в том, что они позволяют быстро выявить и устранить ошибки, возникающие при модификации кода.

В рамках данного проекта было решено покрыть модульными тестами методы класса PerformanceOnReviewsController Create и Update, проверяющие на корректность предложенные пользователем при регистрации значения логина и пароля соответственно.

Для проведения модульных тестов был разработан тестовый класс PerformanceOnReviewsControllerTests, код которого представлен в листинге 1.

Листинг 1 – Исходный код модульных тестов

```
using BeatySalonApp.API.Models;

namespace BeatySalonApp.API.Controllers.Tests
{
    [TestClass()]
    public class PerformanceOnReviewsControllerTests
    {
        private readonly ClientDbContext _context = new();
```

```

[TestMethod()]
public void CreateTest()
{
    var controller = new
PerformanceOnReviewsController(_context);

    var response = controller.Create(39, true, 108);

    Assert.IsTrue(response.IsCompletedSuccessfully);
}

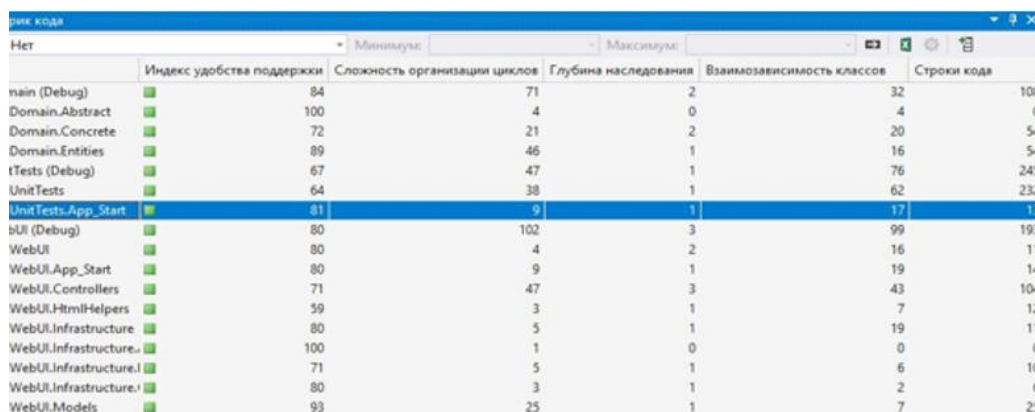
[TestMethod()]
public void UpdateTest()
{
    var controller = new
PerformanceOnReviewsController(_context);

    var response = controller.Update(39, false, 108);

    Assert.IsTrue(response.IsCompletedSuccessfully);
}
}
}

```

Результат проведения модульных тестов представлен на рисунке 43.



Имя	Индекс удобства поддержки	Сложность организации циклов	Глубина наследования	Взаимозависимость классов	Строки кода
main (Debug)	84	71	2	32	108
Domain.Abstract	100	4	0	4	0
Domain.Concrete	72	21	2	20	54
Domain.Entities	89	46	1	16	54
tTests (Debug)	67	47	1	76	245
UnitTests	64	38	1	62	232
UnitTests.App_Start	81	9	1	17	11
UI (Debug)	80	102	3	99	193
WebUI	80	4	2	16	11
WebUI.App_Start	80	9	1	19	14
WebUI.Controllers	71	47	3	43	104
WebUI.HtmlHelpers	59	3	1	7	12
WebUI.Infrastructure	80	5	1	19	11
WebUI.Infrastructure.	100	1	0	0	0
WebUI.Infrastructure.I	71	5	1	6	10
WebUI.Infrastructure.I	80	3	1	2	6
WebUI.Models	93	25	1	7	25

Рисунок 43 – Результаты модульного тестирования

Все тесты были успешно пройдены, что свидетельствует о корректной работе тестируемых методов.

4.3 Метрики кода

Метриками кода называются некоторые количественные величины, позволяющие подсчитать, насколько код программы обладает тем или иным свойством. Существует огромное количество самых разнообразных метрик,

имеющих различное назначение. Наиболее известными являются следующие группы метрик:

- метрики размера программы (SLOC-метрики);
- метрики сложности потока управления программ (метрики Холстеда и цикломатическая сложность).

Рассмотрим метод `button1_Click` класса `SetReview`, отвечающий за отправку и сохранение отзыва пользователя в базе данных. Его код представлен в листинге 2.

Листинг 2 – Метод обработки нажатия на кнопку «Отправить отзыв»

```

/// <summary>
/// Метод обработки нажатия на кнопку "Отправить отзыв"
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button1_Click(object sender, EventArgs e)
{
    var sqlExpression = $"select ID from Review where
ReviewNumber='{qn}'";
    var sqlCommand = new SqlCommand(sqlExpression, sc);
    var ReviewId = sqlCommand.ExecuteScalar();
    sqlExpression = $"select*from PerformanceOnReviews where
UserID=39 and ReviewID='{ReviewId}'";
    sqlCommand = new SqlCommand(sqlExpression, sc);
    var resul = sqlCommand.ExecuteScalar();
    //Проверка - корректный ли отзыв
    if (resul != null)
    {
        if (radioButton1.Checked)
        {
            sqlExpression = $"update PerformanceOnReviews set
CorrectAnswer=1 where UserID=UID and ReviewID='{ReviewId}'";
            sqlCommand = new SqlCommand(sqlExpression, sc);
            sqlCommand.ExecuteNonQuery();
        }
        else
        {
            sqlExpression = $"update PerformanceOnReviews set
CorrectAnswer=0 where UserID=UID and ReviewID='{ReviewId}'";
            sqlCommand = new SqlCommand(sqlExpression, sc);
            sqlCommand.ExecuteNonQuery();
        }
    }
    else
    {
        if (radioButton1.Checked)

```

```

        {
            sqlExpression = $"insert
PerformanceOnReviews(CorrectAnswer, UserID, ReviewID)" +
            $" values (1, '{UID}', '{ReviewId}')";
            sqlCommand = new SqlCommand(sqlExpression, sc);
            sqlCommand.ExecuteNonQuery();
        }
        else
        {
            sqlExpression = $"insert
PerformanceOnReviews(CorrectAnswer, UserID, ReviewID)" +
            $" values (0, '{UID}', '{ReviewId}')";
            sqlCommand = new SqlCommand(sqlExpression, sc);
            sqlCommand.ExecuteNonQuery();
        }
    }
    this.Hide();
}

```

Данный метод находится в клиентской части приложения и запускается при нажатии пользователем на кнопку «Отправить отзыв». Он проверяет на корректность отзыва пользователя, и сохраняет результат в базе данных.

Для данного метода будут вычислены две группы метрик: метрики размера программы и метрики Холстеда.

4.3.1 Метрики размера программы

Данные метрики представляют собой подсчет количества строк программного кода, а также числа строк с комментариями. Вычислим метрики размера программы для приведенного выше программного кода. Результаты представлены в таблице 14.

Таблица 14 – Метрики размера программы

Метрика	Количество
Количество строк кода	42
Строки-комментарии	7

Также вычислим процент комментариев. Он равен $(7/42) \cdot 100 = 16,67$. Это свидетельствует о достаточном количестве комментариев в коде для его хорошего понимания при отсутствии избыточных комментариев, переполняющих программу.

4.3.2 Метрики Холстеда

Для вычисления метрик Холстеда необходимо измерить четыре характеристики программы [33 – 35]:

- число уникальных операторов;
- число уникальных операндов;
- общее число операторов;
- общее число операндов.

С целью вычисления этих характеристик были составлен словарь операторов (таблица 15) и словарь операндов (таблица 16).

Таблица 15 – Словарь операторов

Операторы	Число вхождений
=	14
!=	1
if	3
else	3
{}	8
()	1
+	2
;	18
var	4
new	6
.	3
sqlCommand	6
ExecuteNonQuery	4
Hide	1
n1 = 16	N1 = 76

Таблица 16 – Словарь операндов

Операнды	Число вхождений
sqlExpression	12
sqlCommand	12
ReviewId	6
sc	6
radioButton1	2
"select ID from Review where ReviewNumber='{qn}'"	1
"select*from PerformanceOnReviews where UserID=39 and ReviewID='{ReviewId}'"	1
"update PerformanceOnReviews set CorrectAnswer=1 where UserID=39 and ReviewID='{ReviewId}'"	1
"update PerformanceOnReviews set CorrectAnswer=0 where UserID=39 and ReviewID='{ReviewId}'"	1

Продолжение таблицы 16

Операнды	Число вхождений
"insert PerformanceOnReviews(CorrectAnswer,UserID,ReviewID)"	1
" values (1,39,'{ReviewId}')"	1
"insert PerformanceOnReviews(CorrectAnswer,UserID,ReviewID)"	1
" values (0,39,'{ReviewId}')"	1
null	1
n2 = 14	N2 = 47

Таким образом, число уникальных операторов программы $n1 = 14$, число уникальных операндов программы $n2 = 14$, общее число операторов в программе $N1 = 76$, общее число операндов в программе $N2 = 47$.

На основе метрик Холстеда можно вычислить целый ряд характеристик программы. Рассмотрим некоторые из них.

Расчёт словаря программы производится по формуле:

$$n = n1 + n2 = 14 + 4 = 28,$$

где $n1$ – количество операторов в программе;

$n2$ – количество операндов программе.

Расчёт длины программы производится по формуле:

$$N = N1 + N2 = 76 + 19 = 90,$$

где $N1$ – число вхождений всех операторов в программе;

$N2$ – число вхождений всех операндов программе.

Расчёт объёма программы производится по формуле:

$$V = N * \log_2 n = 90 * \log_2 28 = 432,$$

где N – длина программы;

n – словарь программы.

Расчёт сложности программы производится по формуле:

$$D = (n1/2) * (N2/n2) = (13 / 2) * (47 / 14) = 21,82,$$

где $n1$ – количество операторов в программе;

$n2$ – количество операндов программе;

$N2$ – число вхождений всех операндов программе.

Расчёт оценки интеллектуальных усилий производится по формуле:

$$E = D * V = 21,82 * 432 = 9426,24,$$

где D – сложность программы;

V – объем программы.

Полученные характеристики показывают, что программный код имеет достаточно высокий уровень качества и удобства для дальнейшей поддержки.

Заключение

В ходе выполнения выпускной квалификационной работы было разработано мобильное программное приложение для поддержки процессов функционирования косметического салона.

На этапе анализа требований и предметной области был выполнен анализ программных продуктов аналогичного назначения, в результате которого были выявлены их основные достоинства и недостатки, определены функциональные и нефункциональные требования к создаваемому приложению и произведён анализ целевой аудитории.

Было осуществлено планирование разработки приложения и оценка бюджета. Результаты планирования показали, что общая длительность проекта составляет 57 дней, а предполагаемые затраты на реализацию – 429000 рублей, что свидетельствует об экономической целесообразности данного программного проекта.

На этапе проектирования была составлена логическая модель базы данных и разработана архитектура мобильного приложения. База данных была создана с использованием языка программирования SQL и развёрнута под управлением СУБД PostgreSQL 2019. Серверная часть приложения была разработана на языке C# с использованием среды программирования Microsoft Visual Studio 2022. Клиентская часть была разработана на языке Java с использованием среды программирования Android Studio.

Было выполнено тестирование серверной и клиентской частей разрабатываемого приложения. Оно показало корректность его работы. Также рассчитаны метрики программного кода для оценки его качества.

В дальнейшем планируется опубликовать разработанное приложение в магазинах мобильных приложений для систем Android (Google Play), а также развернуть его серверную часть на облачном сервисе.

Таким образом, в ходе выполнения выпускной квалификационной работы были успешно выполнены все поставленные задачи.

Список использованных источников

1. Распоряжение Президента Российской Федерации от 9 мая 2017 г. № 203 «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы». [Электронный ресурс]. URL: http://zmedu.hostedu.ru/files/ykaz_7668.pdf (дата обращения: 25.04.2025).
2. Программа «Цифровая экономика Российской Федерации». Утверждена распоряжением Правительства Российской Федерации от 28 июля 2017 г. № 1632–р. [Электронный ресурс]. URL: <http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf> (дата обращения: 26.04.2025).
3. Система записи в студию красоты / [Электронный ресурс] // Elastic Nails - сеть маникюрных салонов : [сайт]. URL: elasticnails.ru (дата обращения: 19.05.2025).
4. Система записи на услуги к визажисту / [Электронный ресурс] // Lover : [сайт]. URL: <https://lover.fm/r.php> (дата обращения: 19.05.2025).
5. Система управления салоном красоты / [Электронный ресурс] // Y2Best : [сайт]. URL: <https://y2best.com> (дата обращения: 19.05.2025).
6. Вендров А.М. Структурный подход к проектированию ИС // Портал научно-практических публикаций [Электронный ресурс]. URL: http://citforum.ru/database/case/glava2_1.shtml (дата обращения: 07.05.2025).
7. Приложение с открытым исходным кодом для персональных компьютеров // ArgoUML [Электронный ресурс]. URL: <https://argouml.softonic.ru> (Дата обращения: 15.05.2025).
8. Построение UML-модели в среде Visual Paradigm 16.1. Система обработки заказов [Электронный ресурс]. URL: <http://sp.cs.msu.ru/ooap/exerb2020.html> (Дата обращения: 18.05.2025).
9. Маклаков, С. В. BPwin и ERwin: CASE-средства для разработки информационных систем // Портал научно-практических публикаций [Электронный ресурс].

URL: <https://elib.pnzgu.ru/files/eb/doc/DSA5gziAMfNS.pdf> (дата обращения: 13.05.2025).

10. Проектирование информационных систем [Электронный ресурс]. URL: <https://moodle.kstu.ru/course/view.php?id=4638&> (Дата обращения: 06.05.2025).

11. Варианты использования // Диаграмма вариантов использования [Электронный ресурс]. URL: http://www.telenir.net/uchebniki/samouchitel_uml/p4.php?ysclid=l4hg07qpf139464158 (Дата обращения: 28.04.2025).

12. Что такое интерфейс пользователя? [Электронный ресурс]. URL: <https://www.internet-technologies.ru/articles/newbie/polzovatelskiy-interfeys.html> (Дата обращения: 01.05.2025).

13. Простое руководство по диаграмме компонентов: [Электронный ресурс] // So creately URL: <https://creately.com/blog/ru/uncategorized-ru/учебное-пособие-по-компонентной-диаг/>. (Дата обращения: 04.05.2025).

14. Об одном подходе к разбиению на классы эквивалентности: [Электронный ресурс] // cyberleninka.ru. Теория. URL: <https://cyberleninka.ru/article/n/ob-odnom-podhode-k-razbieniyu-na-klassy-ekvivalentnosti> (дата обращения: 26.03.2025).

15. Граничные условия в неизолированных системах: [Электронный ресурс] // cyberleninka.ru. Теория. URL: <https://cyberleninka.ru/article/n/granichnye-usloviya-v-neizolirovannyh-sistemah>. (Дата обращения: 26.03.2025).

16. Простое сложное программирование: [Электронный ресурс] // <http://habr.com>. Теория. URL: <https://habr.com/ru/companies/timeweb/articles/551754/>. (Дата обращения: 26.03.2025).

17. Применение метрик Холстеда для количественного оценивания характеристик программ ЭВМ: [Электронный ресурс] // cyberleninka.ru. Теория. URL: <https://cyberleninka.ru/article/n/primenenie-metrik-holsteda-dlya->

kolichestvennogo-otsenivaniya-harakteristik-programm-evm. (Дата обращения: 26.03.2025).

18. Грабер, Мартин Введение в SQL [Текст] / Мартин Грабер. — 2-е изд. — Москва: Лори, 2000. — 228 с.

19. Морозов, Е.А. Логическое проектирование баз данных [Текст] / Е. А. Морозов. — 1-е изд. — Москва: МИСиС, 2003. — 67 с.

20. PostgreSQL 2019: официальная документация [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/sql/sql-server/> (дата обращения: 10.05.2025).

21. Rational Unified Process (RUP): методология разработки ПО [Электронный ресурс]. URL: <https://www.ibm.com/ru-ru/cloud/learn/rational-unified-process> (дата обращения: 12.05.2025).

22. UML 2.5: спецификация языка моделирования [Электронный ресурс]. URL: <https://www.omg.org/spec/UML/> (дата обращения: 14.05.2025).

23. Entity Framework Core: документация [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/ef/core/> (дата обращения: 16.05.2025).

24. .NET MAUI: кросс-платформенная разработка [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/dotnet/maui/> (дата обращения: 18.05.2025).

25. Android Studio: руководство разработчика [Электронный ресурс]. URL: <https://developer.android.com/studio> (дата обращения: 20.05.2025).

26. REST API: принципы проектирования [Электронный ресурс]. URL: <https://restfulapi.net/> (дата обращения: 22.05.2025).

27. Microsoft Project: управление проектами [Электронный ресурс]. URL: <https://www.microsoft.com/ru-ru/microsoft-365/project/project-management-software> (дата обращения: 24.05.2025).

28. Visual Paradigm: инструмент UML-моделирования [Электронный ресурс]. URL: <https://www.visual-paradigm.com/> (дата обращения: 26.05.2025).

29. Postman: тестирование API [Электронный ресурс]. URL: <https://www.postman.com/> (дата обращения: 28.05.2025).

30. Холстед, М.Х. Начала науки о программах [Текст] / М.Х. Холстед. — Москва: Мир, 1981. — 304 с.
31. Макконнелл, С. Совершенный код [Текст] / С. Макконнелл. — Москва: Русская редакция, 2019. — 896 с.
32. Фаулер, М. UML. Основы [Текст] / М. Фаулер. — Санкт-Петербург: Символ-Плюс, 2018. — 192 с.
33. Рихтер, Дж. CLR via C# [Текст] / Дж. Рихтер. — Москва: Питер, 2020. — 896 с.
34. Троелсен, Э. Язык программирования C# 9.0 [Текст] / Э. Троелсен. — Москва: Вильямс, 2021. — 1328 с.
35. Фримен, Э. Паттерны проектирования [Текст] / Э. Фримен, Э. Робсон. — Санкт-Петербург: Питер, 2022. — 656 с.

Приложение А
(справочное)
Глоссарий

Android – мобильная операционная система для смартфонов, планшетов и других мобильных устройств, разрабатываемая и выпускаемая американской компанией Google и используемая на подавляющем большинстве современных мобильных устройств.

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

RUP (Rational Unified Process) – методология разработки программного обеспечения, в основе которой лежит итеративная модель разработки.

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания и модификации данных в реляционной базе данных, работающей под управлением соответствующей системой управления базами данных, а также для управления этими данными.

HTTP (HyperText Transfer Protocol) – протокол прикладного уровня передачи данных, основанный на технологии «клиент–сервер».

UML (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес–процессов, системного проектирования и отображения организационных структур.

.NET MAUI (.NET Multi–Platform App UI) – это кросс–платформенная платформа для создания собственных мобильных и классических приложений с помощью Java и XAML.

Entity Framework – представляет специальную объектно–ориентированную технологию на базе фреймворка .NET для работы с данными, обладает высоким уровнем абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища.

Приложение Б
(обязательное)
Исходный код скриптов базы данных

Листинг Б.1 – Исходный код скрипта создания таблиц базы данных

```
-- Создание таблицы 'Clients' (Клиент)

CREATE TABLE Clients (

    client_id INT PRIMARY KEY AUTO_INCREMENT, -- PK, Id клиента

    first_name VARCHAR(50) NOT NULL,          -- Имя

    last_name VARCHAR(50) NOT NULL,           -- Фамилия

    patronymic VARCHAR(50),                   -- Отчество

    login VARCHAR(50) UNIQUE NOT NULL,        -- Логин
    (уникальный)

    password VARCHAR(50) NOT NULL,            -- Пароль (в
    реальной системе должен быть хеширован)

    phone_number VARCHAR(11) UNIQUE NOT NULL  -- Номер телефона
    (уникальный)

    -- history_id INT, -- FK к 'VisitHistory', будет добавлено
    позже как связь Many-to-Many или через промежуточную таблицу

);

-- Создание таблицы 'Services' (Услуги)

CREATE TABLE Services (

    service_id INT PRIMARY KEY AUTO_INCREMENT, -- PK, Id услуги

    service_name VARCHAR(150) NOT NULL,         -- Название услуги

    category VARCHAR(50),                       -- Категория
    услуги (например, "Парикмахерские услуги", "Маникюр")

    price DECIMAL(10, 2) NOT NULL,              -- Стоимость
    (double precision в оригинале, DECIMAL(10,2) лучше для денег)

    duration TIME                               -- Время оказания
    услуги (time(4) в оригинале, TIME)

);

-- Создание таблицы 'Masters' (Мастер)

CREATE TABLE Masters (
```

```

    master_id INT PRIMARY KEY AUTO_INCREMENT,      -- PK, Id мастера
    first_name VARCHAR(50) NOT NULL,                -- Имя
    last_name VARCHAR(50) NOT NULL,                 -- Фамилия
    patronymic VARCHAR(50),                          -- Отчество

    -- schedule_id INT, -- FK к 'Schedule', будет управляться
    -- через связь many-to-many или как часть 'Appointments'

    -- services_offered_id INT, -- FK к 'Services', подразумевает
    -- связь many-to-many

    -- reviews_id INT, -- FK к 'Reviews', подразумевает связь
    -- many-to-many

    -- portfolio_id INT -- FK к 'Portfolio', подразумевает связь
    -- many-to-many
);

-- Создание таблицы 'Portfolio' (Портфолио)

-- Предполагается, что одно портфолио может принадлежать одному
-- мастеру

-- и содержать несколько работ, каждая из которых связана с
-- конкретной услугой.

-- Однако по диаграмме 'Услуга' - это FK в 'Портфолио', а не
-- 'Мастер'.

-- Это означает, что запись в портфолио относится к услуге, а не
-- напрямую к мастеру.

-- Если портфолио для мастера, то нужна FK на мастера. Я сделаю
-- как на диаграмме,

-- но отмечу, что это не типичная структура для портфолио мастера.

CREATE TABLE Portfolio (

    portfolio_item_id INT PRIMARY KEY AUTO_INCREMENT, -- PK, Id
    -- элемента портфолио

    master_id INT NOT NULL,                          -- FK к 'Masters'
    (Добавлено, так как портфолио принадлежит мастеру)

    photo_data TEXT NOT NULL,                         -- Фото
    (text(2048) в оригинале, TEXT для больших строк)

```

```

        service_id INT NOT NULL,                                -- FK к
'Services', услуга, к которой относится работа в портфолио

        description VARCHAR(500),                                -- Описание

        FOREIGN KEY (master_id) REFERENCES Masters(master_id) ON
DELETE CASCADE,

        FOREIGN KEY (service_id) REFERENCES Services(service_id) ON
DELETE CASCADE

    );

-- Создание таблицы 'Reviews' (Отзывы)
CREATE TABLE Reviews (

    review_id INT PRIMARY KEY AUTO_INCREMENT,                -- PK, Id отзыва

    client_id INT NOT NULL,                                    -- FK к 'Clients',
Кто оставил отзыв

    master_id INT NOT NULL,                                    -- FK к 'Masters',
Кому оставлен отзыв

    review_text VARCHAR(500),                                -- Текст отзыва

    rating INT CHECK (rating >= 1 AND rating <= 5), -- Баллы
(integer(2) в оригинале, CHECK для 1-5 звезд)

    FOREIGN KEY (client_id) REFERENCES Clients(client_id) ON
DELETE CASCADE,

    FOREIGN KEY (master_id) REFERENCES Masters(master_id) ON
DELETE CASCADE

);

-- Создание таблицы 'Schedule' (Расписание)
-- Этот компонент, судя по диаграмме, содержит доступные слоты.
-- 'Статус бронирования' указывает, свободен слот или занят.
CREATE TABLE Schedule (

    schedule_id INT PRIMARY KEY AUTO_INCREMENT, -- PK, Id
расписания

```

```

        master_id INT NOT NULL,                                -- FK к 'Masters',
к какому мастеру относится расписание

        schedule_date DATE NOT NULL,                          -- Дата (date(8) в
оригинале, DATE)

        start_time TIME NOT NULL,                              -- Время начала
слота

        is_booked BOOLEAN DEFAULT FALSE,                      -- Статус
бронирования (bool(1) в оригинале, BOOLEAN)

        FOREIGN KEY (master_id) REFERENCES Masters(master_id) ON
DELETE CASCADE

);

-- Создание таблицы 'Appointments' (Запись на прием)
-- Это центральная таблица для бронирования.
-- Она связывает Клиента, Мастера, Услугу и конкретный слот
Расписания.

CREATE TABLE Appointments (

    appointment_id INT PRIMARY KEY AUTO_INCREMENT, -- PK, Id
записи

    client_id INT NOT NULL,                                -- FK к
'Clients'

    master_id INT NOT NULL,                                -- FK к
'Masters'

    service_id INT NOT NULL,                                -- FK к
'Services'

    schedule_id INT NOT NULL,                                -- FK к
'Schedule', ссылка на забронированный слот

    start_time TIME NOT NULL,                                -- Время начала
(взято из Schedule, но также может быть здесь для удобства)

    end_time TIME NOT NULL,                                  -- Время
завершения (рассчитывается на основе start_time и
service_duration)

    appointment_date DATE NOT NULL,                          -- Дата записи
(взято из Schedule, но также может быть здесь для удобства)

```

```

        FOREIGN KEY (client_id) REFERENCES Clients(client_id) ON
DELETE CASCADE,

        FOREIGN KEY (master_id) REFERENCES Masters(master_id) ON
DELETE CASCADE,

        FOREIGN KEY (service_id) REFERENCES Services(service_id) ON
DELETE CASCADE,

        FOREIGN KEY (schedule_id) REFERENCES Schedule(schedule_id) ON
DELETE RESTRICT -- RESTRICT, чтобы слот нельзя было удалить, если
он забронирован

);

-- Создание таблицы 'VisitHistory' (История посещений)
-- Эта таблица является промежуточной для связи Many-to-Many между
Clients и Services,
-- а также может содержать дату посещения.
CREATE TABLE VisitHistory (

    visit_id INT PRIMARY KEY AUTO_INCREMENT,      -- PK, Id записи
                                                    истории посещения

    client_id INT NOT NULL,                        -- FK к 'Clients'

    appointment_id INT NOT NULL,                   -- FK к
'Appointments', ссылка на конкретную запись, которая произошла

    visit_date DATE NOT NULL,                      -- Дата посещения
(date(8) в оригинале, DATE)

    service_id INT NOT NULL,                       -- FK к 'Services'
(Услуга, которая была оказана)

    FOREIGN KEY (client_id) REFERENCES Clients(client_id) ON
DELETE CASCADE,

    FOREIGN KEY (appointment_id) REFERENCES
Appointments(appointment_id) ON DELETE CASCADE,

    FOREIGN KEY (service_id) REFERENCES Services(service_id) ON
DELETE CASCADE

);

```

-- Связующие таблицы для Many-to-Many отношений, которые подразумеваются диаграммой:

-- Мастер может оказывать несколько услуг, и услуга может быть оказана несколькими мастерами.

```
CREATE TABLE MasterServices (
    master_id INT NOT NULL,
    service_id INT NOT NULL,
    PRIMARY KEY (master_id, service_id),
    FOREIGN KEY (master_id) REFERENCES Masters(master_id) ON
DELETE CASCADE,
    FOREIGN KEY (service_id) REFERENCES Services(service_id) ON
DELETE CASCADE
);
```

-- Мастер может иметь несколько элементов портфолио (уже учтено в таблице Portfolio, добавлена FK master_id)

-- Мастер может иметь несколько отзывов (уже учтено в таблице Reviews, добавлена FK master_id)

-- Добавление уникального индекса для Schedule, чтобы предотвратить дублирование слотов для мастера

-- (Если мастер может работать в нескольких местах одновременно, это нужно учитывать, но здесь предполагается один слот в одно время)

```
ALTER TABLE Schedule ADD UNIQUE (master_id, schedule_date,
start_time);
```

Приложение В
(обязательное)
Исходный код серверной части

Листинг В.1 – Серверная часть

```

AccountService
Index.html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Аккаунт</title>
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.0/css/bootst
trap.min.css" rel="stylesheet" />

    <style>
        body {
            background: linear-gradient(120deg, #f7f7f7, #cbd3db);
            margin: 0;
            padding: 0;
            min-height: 100vh;
        }

        .header {
            display: flex;
            flex-direction: row;
            justify-content: space-between;
            align-items: center;
            background: #ffffffcc;
            padding: 20px;
            border-bottom: 1px solid #ddd;
        }

        .header_p {
            display: flex;
            flex-direction: row;
            align-items: center;
        }

        .header_p h3 {
            margin: 0;
            font-weight: 600;
            color: #333;
        }

        .header_p button {
            margin-left: 10px;
            margin-right: 10px;
        }

        .account-container {
            max-width: 600px;
            margin: 40px auto;
            background: #fff;
            border-radius: 6px;

```

```

        box-shadow: 0 2px 10px rgba(0, 0, 0, 0.15);
        padding: 20px 30px;
    }

    .account-container h4 {
        font-weight: 600;
        color: #333;
        margin-bottom: 20px;
    }

    form[name="accountForm"] {
        margin-top: 10px;
    }

    .panel-body {
        margin-top: 15px;
        text-align: center;
    }

    #edit_but {
        margin-right: 20px;
    }

    #delete_but {
        margin-left: 20px;
    }
</style>
</head>
<body>

    <div class="header">
        <div class="header_p">
            <h3>Аккаунт</h3>
        </div>
        <div class="header_p">
            <button type="button" class="btn btn-outline-primary"
id="rasp_Link">Расписание</button>
            <button type="button" class="btn btn-outline-primary"
onclick="window.location.href='https://localhost:7077/'">Выйти</bu
tton>
        </div>
    </div>

    <div class="account-container">
        <h4>Управление учётной записью</h4>
        <form name="accountForm">
            <input type="hidden" name="id" value="0" />

            <div class="form-row">
                <div class="form-group col-md-6">
                    <label for="email">Email:</label>
                    <input id="email" class="form-control"
name="Email" />

```

```

        </div>
        <div class="form-group col-md-6">
            <label for="password">Password:</label>
            <input id="password" class="form-control"
name="Password" />
        </div>
    </div>

    <div class="panel-body">
        <button type="button" id="edit_but" class="btn
btn-primary">Изменить</button>
        <button type="button" id="delete_but" class="btn
btn-danger">Удалить аккаунт</button>
    </div>
</form>
</div>

<script>
    let user_id = new
URLSearchParams(window.location.search).get("id_user");
    console.log(user_id);
    document.location.replace =
"https://localhost:7077/Account";

    var contactButton = document.getElementById('rasp_Link');
    contactButton.addEventListener('click', function () {
        window.location.href =
`https://localhost:7077/Rasp?id_User=${user_id}`;
    });

    // Загрузка данных пользователя
    async function GetUser(id_user) {
        const response = await
fetch(`/Account/Account?id_user=${id_user}`, {
            method: "GET",
            headers: { "Accept": "application/json" }
        });
        if (response.ok === true) {
            const user = await response.json();
            console.log(user);
            const form = document.forms["accountForm"];
            form.elements["id"].value = user.id_user;
            form.elements["email"].value = user.email;
            form.elements["password"].value = user.password;
        }
    }

    // Обновление данных пользователя
    async function EditUser(Id, Email, Password) {
        const response = await fetch("/Account/Account", {
            method: "PUT",
            headers: { "Accept": "application/json", "Content-
Type": "application/json" },

```

```

        body: JSON.stringify({
            id_User: parseInt(Id, 10),
            email: Email,
            password: Password
        })
    });
    if (response.ok === true) {
        const user = await response.json();
        console.log(user);
    }
}

// Удаление пользователя
async function DeleteUser(Id_user) {
    const response = await
fetch(`/Account/Account?id=${Id_user}`, {
    method: "DELETE",
    headers: { "Accept": "application/json" }
});
    if (response.ok === true) {
        const user = await response.json();
        console.log(user);
    }
}

const editButton = document.getElementById("edit_but");
const deleteButton =
document.getElementById("delete_but");

editButton.addEventListener('click', function () {
    const form = document.forms["accountForm"];
    const email = form.elements["email"].value;
    const password = form.elements["password"].value;
    console.log(email, password);
    if (window.confirm("Вы действительно хотите изменить
данные?")) {
        EditUser(user_id, email, password);
    }
});

deleteButton.addEventListener('click', function () {
    if (window.confirm("Вы уверены, что хотите удалить
аккаунт?")) {
        DeleteUser(user_id);
        window.location.href = `https://localhost:7077`;
    }
    document.location.replace =
"https://localhost:7077/Account";
});

// Инициализация: загрузка данных пользователя
GetUser(user_id);
</script>

```

```

</body>
</html>
Account.cs
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace AccountService.Models
{
    [Table("registration")]
    public class Account
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id_User { get; set; }
        [Required]
        public string Email { get; set; }
        [Required]
        public string Password { get; set; }
    }
}
UserRepository.cs
using AccountService.Models;
using Microsoft.EntityFrameworkCore;

namespace AccountService
{
    public class UserRepository : DbContext
    {
        public UserRepository(DbContextOptions options) :
base(options)
        {
        }

        public DbSet<Account> Users { get; set; }
    }
}
APIGatewayServices
Ocelot.json
{
    "Routes": [
        {
            "DownstreamPathTemplate": "/",
            "DownstreamScheme": "https",
            "DownstreamHostAndPorts": [
                {
                    "Host": "localhost",
                    "Port": 7026
                }
            ],
            "UpstreamPathTemplate": "/",
            "UpstreamHttpMethod": [ "GET", "POST" ]
        },
        {

```

```

    "DownstreamPathTemplate": "/Registration/Login",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
      {
        "Host": "localhost",
        "Port": 7026
      }
    ],
    "UpstreamPathTemplate": "/Registration/Login",
    "UpstreamHttpMethod": [ "POST" ]
  },
  {
    "DownstreamPathTemplate": "/Registration/Signup",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
      {
        "Host": "localhost",
        "Port": 7026
      }
    ],
    "UpstreamPathTemplate": "/Registration/Signup",
    "UpstreamHttpMethod": [ "POST" ]
  },
  {
    "DownstreamPathTemplate": "/Registration/id_user",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
      {
        "Host": "localhost",
        "Port": 7026
      }
    ],
    "UpstreamPathTemplate": "/Registration/id_user",
    "UpstreamHttpMethod": [ "GET" ]
  },
  {
    "DownstreamPathTemplate": "/",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
      {
        "Host": "localhost",
        "Port": 7082
      }
    ],
    "UpstreamPathTemplate": "/Rasp",
    "UpstreamHttpMethod": [ "GET" ]
  },
  {
    "DownstreamPathTemplate": "/Rasp/{everything}",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [

```

```

        {
            "Host": "localhost",
            "Port": 7082
        }
    ],
    "UpstreamPathTemplate": "/Rasp/{everything}",
    "UpstreamHttpMethod": [ "GET", "POST", "PUT", "DELETE" ]
},

{
    "DownstreamPathTemplate": "/Account/Account",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
        {
            "Host": "localhost",
            "Port": 7063
        }
    ],
    "UpstreamPathTemplate": "/Account/Account",
    "UpstreamHttpMethod": [ "GET", "PUT", "DELETE" ]
},
{
    "DownstreamPathTemplate": "/Account/SaveUser",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
        {
            "Host": "localhost",
            "Port": 7063
        }
    ],
    "UpstreamPathTemplate": "/Account/SaveUser",
    "UpstreamHttpMethod": [ "POST" ]
},
{
    "DownstreamPathTemplate": "/Account",
    "DownstreamScheme": "https",
    "DownstreamHostAndPorts": [
        {
            "Host": "localhost",
            "Port": 7063
        }
    ],
    "UpstreamPathTemplate": "/Account",
    "UpstreamHttpMethod": [ "GET" ]
}
],

"GlobalConfiguration": {
    "BaseUrl": "http://localhost:7077"
}
}
RaspService
Index.html

```

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width" />
  <title>Список Расписаний</title>

  <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.0/css/bootst
trap.min.css" rel="stylesheet" />

  <style>
    body {
      background: linear-gradient(120deg, #f7f7f7, #cbd3db);
      min-height: 100vh;
      margin: 0;
      padding: 0;
    }

    .header {
      display: flex;
      flex-direction: row;
      justify-content: space-between;
      align-items: center;
      background: #ffffffcc;
      padding: 20px;
      border-bottom: 1px solid #ddd;
    }

    .header_p {
      display: flex;
      flex-direction: row;
      align-items: center;
    }

    .header_p h3 {
      margin: 0;
      font-weight: 600;
      color: #333;
    }

    .header_p button {
      margin-left: 10px;
      margin-right: 10px;
    }

    .page-content {
      max-width: 1200px;
      margin: 30px auto;
      background: #fff;
      border-radius: 6px;
      box-shadow: 0 2px 10px rgba(0, 0, 0, 0.15);
      padding: 20px 30px;
    }
  </style>

```

```

    }

    .page-content h4 {
        font-weight: 600;
        margin-bottom: 20px;
        color: #333;
    }

    .spacer {
        margin-top: 30px;
        margin-bottom: 20px;
        border-top: 1px solid #eee;
    }

    .table {
        margin-top: 20px;
        background-color: #fff;
    }

    .table thead th {
        background: #f3f3f3;
    }

    a[style*="cursor:pointer"] {
        text-decoration: underline;
        font-weight: 500;
        color: #0056b3 !important;
    }
</style>
</head>

<body>

    <div class="header">
        <div class="header_p">
            <h3>Список Расписаний</h3>
        </div>
        <div class="header_p">
            <button type="button" class="btn btn-outline-primary"
id="account_Link">Аккаунт</button>
            <button type="button" class="btn btn-outline-primary"
onclick="window.location.href='https://localhost:7077/'">Выйти</bu
tton>
        </div>
    </div>

    <div class="page-content">

        <h4>Добавить или изменить Расписание</h4>

```

```

<form name="raspForm">
  <input type="hidden" name="id" value="0" />

  <div class="form-row">
    <div class="form-group col-md-6">
      <label for="fio">ФИО (мастера):</label>
      <input id="fio" class="form-control"
name="fio" />
    </div>
    <div class="form-group col-md-6">
      <label for="email">Эл. почта
(клиента):</label>
      <input id="email" class="form-control"
name="email" />
    </div>
  </div>

  <div class="form-row">
    <div class="form-group col-md-6">
      <label for="phone">Номер телефона
(клиента):</label>
      <input id="phone" class="form-control"
name="phone" />
    </div>
    <div class="form-group col-md-6">
      <label for="adress">Услуга:</label>
      <input id="adress" class="form-control"
name="adress" />
    </div>
  </div>

  <div class="form-row">
    <div class="form-group col-md-6">
      <label for="work">Дата:</label>
      <input id="work" class="form-control"
name="work" />
    </div>
    <div class="form-group col-md-6">
      <label for="another">Время:</label>
      <input id="another" class="form-control"
name="another" />
    </div>
  </div>

  <div class="form-row">
    <div class="form-group col-md-12">
      <label for="archiveReason">Дополнительная
информация:</label>
      <input id="archiveReason" class="form-control"
name="archiveReason" />
    </div>
  </div>

```

```

        <div class="panel-body">
            <button type="submit" id="submit" class="btn btn-
primary">Сохранить Расписание</button>
        </div>
    </form>

    <div class="spacer"></div>

    <table class="table table-striped table-hover col-md-6">
        <thead>
            <tr>
                <th>ID</th>
                <th>ФИО</th>
                <th>Почта</th>
                <th>Телефон</th>
                <th>Услуга</th>
                <th>Дата</th>
                <th>Время</th>
                <th>Дополнительная информация</th>
                <th></th>
            </tr>
        </thead>
        <tbody>
        </tbody>
    </table>
</div>

<script>
    let user_id = new
URLSearchParams(window.location.search).get("id_User");
    console.log(user_id);

    var accountButton =
document.getElementById('account_Link');
    accountButton.addEventListener('click', function () {
        window.location.href =
`https://localhost:7063/Account?id_user=${user_id}`;
    });

    // получить все данные расписания
    async function GetRaspData(userId) {
        const response = await
fetch(`/Rasp/GetAll?id_user=${userId}`, {
            method: "GET",
            headers: { "Accept": "application/json" }
        });
        if (response.ok) {
            const archives = await response.json();
            let rows = document.querySelector("tbody");
            rows.innerHTML = ""; // чистим таблицу
            archives.forEach(a => {
                rows.append(row(a));
            });
        }
    }

```

```

        });
    }
}

// получить конкретную запись расписания
async function GetRaspRecord(id) {
    const response = await fetch(`/Rasp/Get?id=${id}`, {
        method: "GET",
        headers: { "Accept": "application/json" }
    });
    if (response.ok) {
        const record = await response.json();
        const form = document.forms["raspForm"];
        form.elements["id"].value = record.id_rasp_data;
        form.elements["fio"].value = record.fio;
        form.elements["email"].value = record.email;
        form.elements["phone"].value = record.phone;
        form.elements["adress"].value = record.address;
        form.elements["work"].value = record.work;
        form.elements["another"].value = record.another;
        form.elements["archiveReason"].value =
record.archiveReason;
    }
}

// добавить новую запись расписания
async function CreateRaspRecord(fio, email, phone,
address, work, another, archiveReason, userId) {
    const response = await fetch("/Rasp/Create", {
        method: "POST",
        headers: { "Accept": "application/json", "Content-
Type": "application/json" },
        body: JSON.stringify({
            fio: fio,
            email: email,
            phone: phone,
            address: address,
            work: work,
            another: another,
            archiveReason: archiveReason,
            id_User: userId
        })
    });
    if (response.ok) {
        const record = await response.json();
        reset();

document.querySelector("tbody").append(row(record));
    }
}

// обновить запись расписания

```

```

    async function EditRaspRecord(id, fio, email, phone,
address, work, another, archiveReason, userId) {
        const response = await fetch("/Rasp/Update", {
            method: "PUT",
            headers: { "Accept": "application/json", "Content-
Type": "application/json" },
            body: JSON.stringify({
                id_rasp_data: parseInt(id),
                fio: fio,
                email: email,
                phone: phone,
                address: address,
                work: work,
                another: another,
                archiveReason: archiveReason,
                id_User: parseInt(userId)
            })
        });
        if (response.ok) {
            const record = await response.json();
            reset();
            document.querySelector("tr[data-rowid='" +
record.id_rasp_data + "']").replaceWith(row(record));
        }
    }

    // удалить запись расписания
    async function DeleteRaspRecord(id) {
        const response = await fetch(`/Rasp/Delete?id=${id}`,
{
            method: "DELETE",
            headers: { "Accept": "application/json" }
        });
        if (response.ok) {
            const record = await response.json();
            document.querySelector("tr[data-rowid='" +
record.id_rasp_data + "']").remove();
        }
    }

    // сброс формы
    function reset() {
        const form = document.forms["raspForm"];
        form.reset();
        form.elements["id"].value = 0;
    }

    // создание строки таблицы расписания
    function row(record) {
        const tr = document.createElement("tr");
        tr.setAttribute("data-rowid", record.id_rasp_data);

        const idTd = document.createElement("td");

```

```

idTd.append(record.id_rasp_data);
tr.append(idTd);

const fioTd = document.createElement("td");
fioTd.append(record.fio);
tr.append(fioTd);

const emailTd = document.createElement("td");
emailTd.append(record.email);
tr.append(emailTd);

const phoneTd = document.createElement("td");
phoneTd.append(record.phone);
tr.append(phoneTd);

const addressTd = document.createElement("td");
addressTd.append(record.address);
tr.append(addressTd);

const workTd = document.createElement("td");
workTd.append(record.work);
tr.append(workTd);

const anotherTd = document.createElement("td");
anotherTd.append(record.another);
tr.append(anotherTd);

const reasonTd = document.createElement("td");
reasonTd.append(record.archiveReason);
tr.append(reasonTd);

const linksTd = document.createElement("td");

const editLink = document.createElement("a");
editLink.setAttribute("style",
"cursor:pointer;padding:8px;color:DodgerBlue");
editLink.append("Изменить");
editLink.addEventListener("click", e => {
    e.preventDefault();
    GetRaspRecord(record.id_rasp_data);
});
linksTd.append(editLink);

const removeLink = document.createElement("a");
removeLink.setAttribute("style",
"cursor:pointer;padding:8px;color:DodgerBlue");
removeLink.append("Удалить");
removeLink.addEventListener("click", e => {
    e.preventDefault();
    DeleteRaspRecord(record.id_rasp_data);
});
linksTd.append(removeLink);

```

```

        tr.appendChild(linksTd);

        return tr;
    }

    // Обработчик отправки формы
    document.forms["raspForm"].addEventListener("submit", e =>
{
    e.preventDefault();
    const form = document.forms["raspForm"];
    const id = form.elements["id"].value;
    const fio = form.elements["fio"].value;
    const email = form.elements["email"].value;
    const phone = form.elements["phone"].value;
    const address = form.elements["adress"].value;
    const work = form.elements["work"].value;
    const another = form.elements["another"].value;
    const archiveReason =
form.elements["archiveReason"].value;

    if (id == 0) {
        CreateRaspRecord(fio, email, phone, address, work,
another, archiveReason, user_id);
    } else {
        EditRaspRecord(id, fio, email, phone, address,
work, another, archiveReason, user_id);
    }
});

    // Инициализация: загрузка данных расписания
    GetRaspData(user_id);
</script>
</body>
</html>
RaspController.cs
using RaspService.Models;
using Microsoft.AspNetCore.Mvc;
using System.Linq;

namespace RaspService.Controllers
{
    [Route("[controller]")]
    [ApiController]
    public class RaspController : ControllerBase
    {
        private readonly RaspRepository _repository;

        public RaspController(RaspRepository repository)
        {
            _repository = repository;
        }

        // GET /Rasp/GetAll?id_user=3

```

```

[HttpGet("GetAll")]
public IActionResult GetAll([FromQuery] int id_user)
{
    try
    {
        var rasps = _repository.RaspDatas
            .Where(a => a.Id_User == id_user)
            .ToList();
        return Ok(rasps);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}

[HttpPost("SaveUser")]
public IActionResult SaveUser([FromBody] int user)
{
    // Здесь должна быть логика сохранения пользователя
    return Ok();
}

// GET /Rasp/Get?id=5
[HttpGet("Get")]
public IActionResult Get(int id)
{
    try
    {
        var record = _repository.RaspDatas.Find(id);
        if (record == null)
        {
            return NotFound($"Запись с id={id} не
найдена");
        }
        return Ok(record);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}

// POST /Rasp/Create
[HttpPost("Create")]
public IActionResult Create([FromBody] RaspData request)
{
    try
    {
        if (request == null)
            return BadRequest("Неверные данные запроса");

        _repository.Add(request);
    }
}

```

```

        _repository.SaveChanges();
        return Ok(request);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}

// PUT /Rasp/Update
[HttpPut("Update")]
public IActionResult Update([FromBody] RaspData request)
{
    try
    {
        if (request == null)
            return BadRequest("Неверные данные для
обновления");

        var record =
_repository.RaspDatas.Find(request.Id_rasp_data);
        if (record == null)
            return NotFound($"Запись с ID
{request.Id_rasp_data} не найдена");

        record.FIO = request.FIO;
        record.Email = request.Email;
        record.Phone = request.Phone;
        record.Address = request.Address;
        record.Work = request.Work;
        record.Another = request.Another;
        record.ArchiveReason = request.ArchiveReason;

        _repository.SaveChanges();
        return Ok(request);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}

// DELETE /Rasp/Delete?id=5
[HttpDelete("Delete")]
public IActionResult Delete(int id)
{
    try
    {
        var record = _repository.RaspDatas.Find(id);
        if (record == null)
        {
            return NotFound($"Запись с ID {id} не
найдена");

```

```

        }
        _repository.Remove(record);
        _repository.SaveChanges();
        return Ok(record);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}

}

RaspData.cs
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace RaspService.Models
{
    [Table("rasp_data")]
    public class RaspData
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        [Column("Id_rasp_data")]
        public int Id_rasp_data { get; set; }

        [Required]
        [Column("FIO")]
        public string FIO { get; set; } = string.Empty;

        [Column("Email")]
        public string Email { get; set; } = string.Empty;

        [Column("Phone")]
        public string Phone { get; set; } = string.Empty;

        [Column("Address")]
        public string Address { get; set; } = string.Empty;

        [Column("Work")]
        public string Work { get; set; } = string.Empty;

        [Column("Another")]
        public string Another { get; set; } = string.Empty;

        [Column("Id_User")]
        public int Id_User { get; set; }

        [Column("ArchiveReason")]
        public string? ArchiveReason { get; set; }
    }
}

RaspRepository.cs

```

```

using RaspService.Models;
using Microsoft.EntityFrameworkCore;

namespace RaspService
{
    public class RaspRepository : DbContext
    {
        public RaspRepository(DbContextOptions<RaspRepository>
options)
            : base(options)
        {
        }

        public DbSet<RaspData> RaspDatas { get; set; }
    }
}

.field input:focus {
    border-color: #6A82FB;
}

.btn {
    height: 50px;
    width: 100%;
    border-radius: 4px;
    position: relative;
    overflow: hidden;
    margin-top: 25px;
}

.btn .btn-layer {
    height: 100%;
    width: 300%;
    position: absolute;
    left: -100%;
    background: linear-gradient(to right, #6A82FB,
#FC5C7D);
    transition: all 0.4s ease;
    border-radius: 4px;
}

.btn:hover .btn-layer {
    left: 0;
}

.btn input[type="submit"] {
    height: 100%;
    width: 100%;
    z-index: 1;
    position: relative;
    background: none;
    border: none;
    color: #fff;
    font-size: 18px;
}

```

```

        font-weight: 500;
        cursor: pointer;
    }

    #signup:checked ~ .form-container .form-inner {
        margin-left: -50%;
    }

    #message,
    #message_s {
        color: firebrick;
        min-height: 20px;
        margin-top: 5px;
        font-size: 14px;
    }
</style>
</head>

<body>
    <div class="wrapper">

        <div class="title-text">
            <div class="title login">Вход</div>
            <div class="title signup">Регистрация</div>
        </div>

        <div class="slide-controls">
            <input type="radio" name="slide" id="login" checked />
            <input type="radio" name="slide" id="signup" />
            <label for="login" class="slide login">Login</label>
            <label for="signup" class="slide
signup">Signup</label>
            <div class="slider-tab"></div>
        </div>

        <div class="form-container">
            <div class="form-inner">

                <form name="loginForm" class="login"
method="post">

                    <p id="message"></p>
                    <div class="field">
                        <input id="email1"
                            name="Email"
                            type="text"
                            placeholder="Email Address"
                            required />
                    </div>
                    <div class="field">
                        <input id="password1"
                            name="Password"
                            type="password"

```

```

placeholder="Password"
required />
</div>
<div class="field btn">
  <div class="btn-layer"></div>
  <input id="login_button" type="submit"
value="Login" />
</div>
</form>

<form name="signupForm" class="signup"
method="post">
  <p id="message_s"></p>
  <div class="field">
    <input id="email2"
      name="Email"
      type="text"
      placeholder="Email Address"
      required />
  </div>
  <div class="field">
    <input id="password2"
      name="Password"
      type="password"
      placeholder="Password"
      required />
  </div>
  <div class="field">
    <input id="cpassword2"
      type="password"
      placeholder="Confirm password"
      required />
  </div>
  <div class="field btn">
    <div class="btn-layer"></div>
    <input id="signup_button" type="submit"
value="Signup" />
  </div>
</form>
</div>
</div>
</div>
</div>

<script>
  const loginText = document.querySelector(".title-text
.login");
  const loginForm = document.querySelector("form.login");
  const loginBtn = document.querySelector("label.login");
  const signupBtn = document.querySelector("label.signup");
  signupBtn.onclick = () => {
    loginForm.style.marginLeft = "-50%";
    loginText.style.marginLeft = "-50%";
  }

```

```

};
loginBtn.onclick = () => {
    loginForm.style.marginLeft = "0%";
    loginText.style.marginLeft = "0%";
};

document
    .getElementById("login_button")
    .addEventListener("click", function (event) {
        event.preventDefault();
        let output = "";
        const email =
document.getElementById("email1").value;
        const password =
document.getElementById("password1").value;
        const re = /^[\\w-\\.]+@[\\w-]+\\. [a-z]{2,4}$/i;
        const valid = re.test(email);

        if (valid) {
            if (email !== "" && password !== "") {
                PostUserLogin();
            }
        } else {
            output = "Адрес электронной почты введен
неправильно!";
        }
        document.getElementById("message").innerHTML =
output;
    });

async function PostUserLogin() {
    const response = await fetch("/Registration/Login", {
        method: "POST",
        body: new FormData(document.forms.loginForm),
    });
    if (response.ok) {
        const user = await response.json();
        console.log(user);
        SendToRaspData(user);
    } else {
        alert("Пользователь не найден.
Зарегистрируйтесь!");
        const form = document.forms["loginForm"];
        form.reset();
        document.forms.loginForm.style.display = "block";
    }
}

async function SendToRaspData(user) {
    const response = await
fetch("https://localhost:7077/Rasp/SaveUser", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
    });

```

```

        body: JSON.stringify(user),
    });
    console.log(user);
    if (response.ok) {
        window.location.href =
`https://localhost:7077/Rasp?id_User=${user}`;
    }
}

document
    .getElementById("signup_button")
    .addEventListener("click", function (event) {
        event.preventDefault();
        let output = "";
        const email =
document.getElementById("email2").value;
        const password =
document.getElementById("password2").value;
        const cpassword =
document.getElementById("cpassword2").value;
        const re = /^[\\w-\\.]+@[\\w-]+\\. [a-z]{2,4}$/i;
        const valid = re.test(email);

        if (valid) {
            if (password === cpassword) {
                if (email !== "" && password !== "") {
                    PostUserSignup();
                }
            } else {
                alert("Пароли не совпадают!");
            }
        } else {
            output = "Адрес электронной почты введен
неправильно!";
        }
        document.getElementById("message_s").innerHTML =
output;
    });

async function PostUserSignup() {
    const response = await fetch("/Registration/Signup", {
        method: "POST",
        body: new FormData(document.forms.signupForm),
    });
    if (response.ok) {
        const user = await response.json();
        console.log(user);
    } else {
        alert("Пользователь создан. Авторизируйтесь!");
        const form = document.forms["loginForm"];
        form.reset();
        document.forms.loginForm.style.display = "block";
    }
}

```

```

    }
</script>
</body>
</html>
RegistrationController.cs
using LoginRegistrationService.Models;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Npgsql;
using System.Net;

namespace LoginRegistrationService.Controllers
{
    [Route("[controller]")]
    [ApiController]
    public class RegistrationController : ControllerBase
    {
        private readonly RegistrationRepository _userRepository;

        public RegistrationController(RegistrationRepository
userRepository)
        {
            _userRepository = userRepository;
        }
        [HttpPost("Signup")]
        public IActionResult Registration([FromForm]Registration
request)
        {
            var user = _userRepository.Users.FirstOrDefault(u =>
u.Email == request.Email && u.Password == request.Password);
            if (user == null)
            {
                _userRepository.Add(request);
                _userRepository.SaveChanges();
                return Ok(request.Id_User);
            }
            else
            {
                return NotFound("Этот пользователь уже существует.
Пожалуйста, войдите!");
            }
        }

        [HttpPost("Login")]
        public IActionResult LoginAsync([FromForm] Registration
request)
        {
            var user = _userRepository.Users.FirstOrDefault(u =>
u.Email == request.Email && u.Password == request.Password);
            if (user == null)
            {

```

```

        return NotFound("Пользователь не найден.
Зарегистрируйтесь!");
    }
    else
    {
        return Ok(user.Id_User);
    }
}
}
}
}
Registration.cs
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
namespace LoginRegistrationService.Models
{
    [Table("registration")]
    public class Registration
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id_User { get; set; }
        [Required]
        public string Email { get; set; }
        [Required]
        public string Password { get; set; }
    }
}
RegistrationRepository.cs
using LoginRegistrationService.Models;
using Microsoft.EntityFrameworkCore;

namespace LoginRegistrationService
{
    public class RegistrationRepository:DbContext
    {
        public RegistrationRepository(DbContextOptions options) :
base(options)
        {
        }

        public DbSet<Registration> Users { get; set; }
    }
}
Program.cs:
using kurs.Services;
using Microsoft.AspNetCore.Components;
using Microsoft.AspNetCore.Components.Web;

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddRazorPages();
builder.Services.AddServerSideBlazor();
builder.Services.AddSingleton<DataService>();

```

```

var dataService =
builder.Services.BuildServiceProvider().GetRequiredService<DataSer
vice>();
dataService.InitializeData();

var app = builder.Build();

if (!app.Environment.IsDevelopment())
{
    // The default HSTS value is 30 days. You may want to change
    this for production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.MapBlazorHub();
app.MapFallbackToPage("/_Host");

app.Run();

```

MainLayout.cs:

```

@using kurs.Services
@using kurs.Models
@inherits LayoutComponentBase
@inject NavigationManager NavigationManager
@inject DataService DataService

@Body

@code {
    private bool IsUserLoggedIn { get; set; }
    private bool _navigationInitiated = false;

    protected override void OnInitialized()
    {
        var users = DataService.ReadAllLines("users@SQL$TABLE");
        IsUserLoggedIn = users.Any(u => u.Split(",")[0] ==
"test");
    }

    protected override async Task OnAfterRenderAsync(bool
firstRender)
    {
        if (firstRender && !_navigationInitiated)
        {
            _navigationInitiated = true;

```

```

        // Всегда перенаправляем на /login, если пользователь
        не авторизован
        if (!IsUserLoggedIn)
        {
            NavigationManager.NavigateTo("/login", true);
        }
        // Если пользователь авторизован, и мы находимся на
        корневом пути, перенаправляем на /main
        else if (IsUserLoggedIn && NavigationManager.Uri ==
        NavigationManager.BaseUri)
        {
            NavigationManager.NavigateTo("/login", true);
        }
    }

    await base.OnAfterRenderAsync(firstRender);
}
}

```

App.razor:

```

<Router AppAssembly="@typeof(App).Assembly">
    <Found Context="routeData">
        <RouteView RouteData="@routeData"
        DefaultLayout="@typeof(MainLayout)" />
        <FocusOnNavigate RouteData="@routeData" Selector="h1" />
    </Found>
    <NotFound>
        <PageTitle>Not found</PageTitle>
        <LayoutView Layout="@typeof(MainLayout)">
            <p role="alert">Sorry, there's nothing at this
            address.</p>
        </LayoutView>
    </NotFound>
</Router>

```

Register.Page.cs:

```

@page "/register"
@using kurs.Services
@using kurs.Models
@inject NavigationManager NavigationManager
@inject DataService DataService
@inject IJSRuntime JSRuntime
@using System.Text.RegularExpressions

<h1 style="text-align: center;">Регистрация</h1>
<div class="form-group">
    <label>Логин</label>
    <input type="text" class="form-control" @bind="NewUser.Login"
    />
</div>
<div class="form-group">
    <label>Пароль</label>

```

```

        <input type="password" class="form-control"
@bind="NewUser.Password" id="passwordInput"
@oninput="ValidatePassword" />
        <div class="password-meter-container">
            <div class="password-meter">
                <div class="strength-meter"
id="passwordInputStrengthMeter"></div>
            </div>
            <ul class="requirements-list">
                @foreach (var r in Requirements)
                {
                    <li style="color: red;">@r</li>
                }
            </ul>
        </div>
</div>
<div class="form-group">
    <label>Подтверждение пароля</label>
    <input type="password" class="form-control"
id="confirmPasswordInput" @bind="ConfirmPassword"
@oninput="ValidateConfirmPassword" />
    @if (!string.IsNullOrEmpty(ConfirmPasswordError))
    {
        <p class="error">@ConfirmPasswordError</p>
    }
</div>
<div class="form-group">
    <label>Имя</label>
    <input type="text" class="form-control" @bind="NewUser.Name"
/>
</div>
<div class="form-group">
    <label>Номер телефона</label>
    <input type="text" class="form-control"
@bind="NewUser.PhoneNumber" />
</div>
<button class="btn" @onclick="HandleRegister"
disabled="@(!IsPassValid)">Зарегистрироваться</button>
<br />
<a href="/login">Уже есть аккаунт?</a>
@if (!string.IsNullOrEmpty(ErrorMessage))
{
    <div class="alert alert-danger">@ErrorMessage</div>
}

<style>
    .password-meter-container {
        width: 100%;
        margin-top: 5px;
        margin-bottom: 10px;
        position: relative;
        height: 130px;
        overflow: hidden;

```

```

    }

    .password-meter {
        width: 100%;
        border: 1px solid #ddd;
        height: 10px;
        background-color: #eee;
        position: relative;
    }

    .strength-meter {
        height: 100%;
        width: 0%;
        transition: width 0.3s ease;
        position: absolute;
        top: 0;
        left: 0;
    }

    .requirements-list {
        padding-left: 20px;
        margin-top: 5px;
        margin-bottom: 0;
        position: absolute;
        top: 10px;
        z-index: 1;
    }

    .requirements-list li {
        margin-bottom: 2px;
    }

    .error {
        color: red;
        font-size: 0.9rem;
        margin-top: 5px;
        margin-bottom: 0px;
    }
</style>

@code {
    private User NewUser { get; set; } = new User();
    private string? ErrorMessage { get; set; }
    private string? ConfirmPassword { get; set; }
    private string? ConfirmPasswordError { get; set; }
    private DotNetObjectReference<RegisterPage>? dotNetHelper;
    private List<string> Requirements { get; set; } = new
List<string>();
    private bool IsPassValid
    {
        get
        {
            if (NewUser.Password == null) return false;

```

```

        return NewUser.Password.Length >= 6 &&
            Regex.IsMatch(NewUser.Password, "[a-z]") &&
            Regex.IsMatch(NewUser.Password, "[A-Z]") &&
            Regex.IsMatch(NewUser.Password, "[0-9]") &&
            string.IsNullOrEmpty(ConfirmPasswordError);
    }
}

protected override void OnInitialized()
{
    dotNetHelper = DotNetObjectReference.Create(this);
}

protected override async Task OnAfterRenderAsync(bool
firstRender)
{
    if (firstRender)
    {
        await JSRuntime.InvokeVoidAsync("validatePassword",
"passwordInput", NewUser.Password, dotNetHelper);
        await
JSRuntime.InvokeVoidAsync("validateConfirmPassword",
"passwordInput", "confirmPasswordInput", dotNetHelper);
    }
}

private async Task ValidatePassword()
{
    await JSRuntime.InvokeVoidAsync("validatePassword",
"passwordInput", NewUser.Password, dotNetHelper);
    StateHasChanged();
}

private async Task ValidateConfirmPassword()
{
    await JSRuntime.InvokeVoidAsync("validateConfirmPassword",
"passwordInput", "confirmPasswordInput", dotNetHelper);
    StateHasChanged();
}

[JSInvokable]
public void SetConfirmPasswordError(string? error)
{
    ConfirmPasswordError = error;
    StateHasChanged();
}

[JSInvokable]
public void SetRequirements(List<string> requirements)
{
    Requirements = requirements;
    StateHasChanged();
}

```

```

    }

    private void HandleRegister()
    {
        if (string.IsNullOrEmpty(NewUser.Login) ||
            string.IsNullOrEmpty(NewUser.Password) ||
                string.IsNullOrEmpty(NewUser.Name) ||
            string.IsNullOrEmpty(NewUser.PhoneNumber) || !IsPassValid)
        {
            ErrorMessage = "Заполните все поля корректно";
            return;
        }
        if (NewUser.Password != ConfirmPassword)
        {
            ErrorMessage = "Пароли не совпадают";
            return;
        }

        var users = DataService.ReadAllLines("users@SQL$TABLE");
        if (users.Any(u => u.Split(",")[0] == NewUser.Login))
        {
            ErrorMessage = "Пользователь с таким логином уже существует";
            return;
        }

        users.Add($"{NewUser.Login},{NewUser.Password},{NewUser.Name},{New
        User.PhoneNumber}");
        DataService.WriteAllLines("users@SQL$TABLE", users);
        NavigationManager.NavigateTo("/main");
    }
}

```

MainPage.cs:

@page "/main"

@using kurs.Components

```

<div class="main-container">
    <div class="sidebar">
        <div class="menu-item" @onclick="() =>
            ShowComponent(ComponentType.Profile)">Профиль</div>
        <div class="menu-item" @onclick="() =>
            ShowComponent(ComponentType.Masters)">Мастера</div>
        <div class="menu-item" @onclick="() =>
            ShowComponent(ComponentType.Services)">Услуги</div>
        <div class="menu-item" @onclick="() =>
            ShowComponent(ComponentType.Appointment)">Записаться на
            прием</div>
    </div>

    <div class="content">
        @switch (CurrentComponent)

```

```

        {
            case ComponentType.Profile:
                <UserProfile />
                break;
            case ComponentType.Masters:
                <MasterList />
                break;
            case ComponentType.Services:
                <ServiceList />
                break;
            case ComponentType.Appointment:
                <AppointmentForm />
                break;
            default:
                <div>Выберите пункт меню</div>
                break;
        }
    </div>
</div>

<style>
    .main-container {
        display: flex;
        min-height: 100vh;
    }

    .sidebar {
        width: 250px;
        background: linear-gradient(to bottom, #2c3e50, #3498db);
        /* Градиент темно-синего и синего */
        color: white;
        display: flex;
        flex-direction: column;
    }

    .menu-item {
        padding: 18px;
        cursor: pointer;
        transition: background-color 0.3s ease;
        font-size: 1.1rem;
    }

    .menu-item:hover {
        background-color: rgba(255, 255, 255, 0.1);
    }

    .content {
        flex: 1;
        padding: 30px;
    }
</style>

@code {

```

```

        private enum ComponentType { Profile, Masters, Services,
Appointment }
        private ComponentType CurrentComponent { get; set; }

        private void ShowComponent(ComponentType componentType)
        {
            CurrentComponent = componentType;
        }
    }
}

```

```

LoginPage.cs:
@page "/login"
@using kurs.Services
@using kurs.Models
@Inject NavigationManager NavigationManager
@Inject DataService DataService

<h1 style="text-align: center;">Авторизация</h1>

<div class="form-group">
    <label>Логин</label>
    <input type="text" class="form-control" @bind="Login" />
</div>
<div class="form-group">
    <label>Пароль</label>
    <input type="password" class="form-control" @bind="Password"
/>
</div>

<button class="btn" @onclick="HandleLogin">Войти</button>
<br />
<a href="/register">Если нет аккаунта, то зарегистрироваться</a>

@if (!string.IsNullOrEmpty(ErrorMessage))
{
    <div class="alert alert-danger">@ErrorMessage</div>
}

@code {
    private string? Login { get; set; }
    private string? Password { get; set; }
    private string? ErrorMessage { get; set; }

    private void HandleLogin()
    {
        var users =
DataService.ReadAllLines("users@SQL$TABLE").Select(u => new
User(u)).ToList();
        var user = users.FirstOrDefault(u => u.Login == Login &&
u.Password == Password);

        if (user != null)
        {

```

```

        NavigationManager.NavigateTo("/main");
    }
    else
    {
        ErrorMessage = "Неверный логин или пароль";
    }
}
}

```

User.cs:

```

namespace kurs.Models
{
    public class User
    {
        public string? Login { get; set; }
        public string? Password { get; set; }
        public string? Name { get; set; }
        public string? PhoneNumber { get; set; }

        public User(string line)
        {
            string[] parts = line.Split(',');
            if (parts.Length == 4)
            {
                Login = parts[0];
                Password = parts[1];
                Name = parts[2];
                PhoneNumber = parts[3];
            }
        }

        public User() { }
    }
}

```

Service.cs:

```

namespace kurs.Models
{
    public class Service
    {
        public string? Name { get; set; }
        public string? Type { get; set; }
        public string? Description { get; set; }
        public decimal Price { get; set; }

        public Service(string line)
        {
            string[] parts = line.Split(',');
            if (parts.Length == 4)
            {
                Name = parts[0];
                Type = parts[1];
            }
        }
    }
}

```

```

        Description = parts[2];
        Price = decimal.Parse(parts[3]);
    }
}

public Service() { }
}
}

```

Review.cs:

```

namespace kurs.Models
{
    public class Review
    {
        public string? Client { get; set; }
        public string? Master { get; set; }
        public string? Service { get; set; }
        public int Rating { get; set; }
        public string? Text { get; set; }
        public int Value { get; set; }

        public Review(string line)
        {
            string[] parts = line.Split(',');
            if (parts.Length == 6)
            {
                Client = parts[0];
                Master = parts[1];
                Service = parts[2];
                Rating = int.Parse(parts[3]);
                Text = parts[4];
                Value = int.Parse(parts[5]);
            }
        }

        public Review() { }
    }
}

```

Portfolio.cs:

```

namespace kurs.Models
{
    public class Portfolio
    {
        public string? Name { get; set; }
        public string? Text { get; set; }
        public string? Master { get; set; }
        public string? Service { get; set; }

        public Portfolio(string line)
        {

```

```

        string[] parts = line.Split(',');
        if (parts.Length == 4)
        {
            Name = parts[0];
            Text = parts[1];
            Master = parts[2];
            Service = parts[3];
        }
    }

    public Portfolio() { }
}

```

Master.cs:

```

namespace kurs.Models
{
    public class Master
    {
        public string? FirstName { get; set; }
        public string? LastName { get; set; }
        public int Experience { get; set; }
        public string? Specialization { get; set; }

        public Master(string line)
        {
            string[] parts = line.Split(',');
            if (parts.Length == 4)
            {
                FirstName = parts[0];
                LastName = parts[1];
                Experience = int.Parse(parts[2]);
                Specialization = parts[3];
            }
        }

        public Master() { }
    }
}

```

App.cs:

```

using System;

namespace kurs.Models
{
    public class Appointment
    {
        public DateTime Date { get; set; }
        public string? Time { get; set; }
        public string? Service { get; set; }
        public string? Master { get; set; }
    }
}

```

```

        public string? Client { get; set; }
        public string? Status { get; set; } // Добавляем свойство
        Status

        public Appointment(string line)
        {
            string[] parts = line.Split(',');
            if (parts.Length == 6)
            {
                Date = DateTime.Parse(parts[0]);
                Time = parts[1];
                Service = parts[2];
                Master = parts[3];
                Client = parts[4];
                Status = parts[5]; // Инициализируем Status
            }
        }

        public Appointment() { }
    }
}

```

UserProdile.cs:

```

@using kurs.Services
@using kurs.Models
@Inject DataService DataService

<h3 style="text-align: center;">Профиль</h3>
<div class="form-group">
    <label>Логин</label>
    <input type="text" class="form-control"
@bind="CurrentUser.Login" />
</div>
<div class="form-group">
    <label>Пароль</label>
    <input type="text" class="form-control"
@bind="CurrentUser.Password" />
</div>
<div class="form-group">
    <label>Имя</label>
    <input type="text" class="form-control"
@bind="CurrentUser.Name" />
</div>
<div class="form-group">
    <label>Номер телефона</label>
    <input type="text" class="form-control"
@bind="CurrentUser.PhoneNumber" />
</div>
<button class="btn" @onclick="SaveProfile">Сохранить</button>

<h3 style="text-align: center;">Активные записи</h3>
@if (ActiveAppointments != null && ActiveAppointments.Count > 0)
{

```

```

<table class="table">
  <thead>
    <tr>
      <th>Дата</th>
      <th>Время</th>
      <th>Услуга</th>
      <th>Мастер</th>
      <th>Статус</th>
    </tr>
  </thead>
  <tbody>
    @foreach (var appointment in ActiveAppointments)
    {
      <tr>
        <td>@appointment.Date.ToShortDateString()</td>
        <td>@appointment.Time</td>
        <td>@appointment.Service</td>
        <td>@appointment.Master</td>
        <td>@appointment.Status</td>
      </tr>
    }
  </tbody>
</table>
}
else
{
  <p>Нет активных записей</p>
}

<h3 style="text-align: center;">Ожидают утверждения</h3>
@if (PendingAppointments != null && PendingAppointments.Count > 0)
{
  <table class="table">
    <thead>
      <tr>
        <th>Дата</th>
        <th>Время</th>
        <th>Услуга</th>
        <th>Мастер</th>
        <th>Статус</th>
      </tr>
    </thead>
    <tbody>
      @foreach (var appointment in PendingAppointments)
      {
        <tr>
          <td>@appointment.Date.ToShortDateString()</td>
          <td>@appointment.Time</td>
          <td>@appointment.Service</td>
          <td>@appointment.Master</td>
          <td>@appointment.Status</td>
        </tr>
      }
    }
  </tbody>
</table>
}

```

```

        </tbody>
    </table>
}
else
{
    <p>Нет записей, ожидающих утверждения</p>
}

<h3 style="text-align: center;">Отмененные записи</h3>
@if (CanceledAppointments != null && CanceledAppointments.Count >
0)
{
    <table class="table">
        <thead>
            <tr>
                <th>Дата</th>
                <th>Время</th>
                <th>Услуга</th>
                <th>Мастер</th>
                <th>Статус</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var appointment in CanceledAppointments)
            {
                <tr>
                    <td>@appointment.Date.ToShortDateString()</td>
                    <td>@appointment.Time</td>
                    <td>@appointment.Service</td>
                    <td>@appointment.Master</td>
                    <td>@appointment.Status</td>
                </tr>
            }
        </tbody>
    </table>
}
else
{
    <p>Нет отмененных записей</p>
}

<h3 style="text-align: center;">История записей</h3>
@if (HistoryAppointments != null && HistoryAppointments.Count > 0)
{
    <table class="table">
        <thead>
            <tr>
                <th>Дата</th>
                <th>Время</th>
                <th>Услуга</th>
                <th>Мастер</th>
                <th></th>
            </tr>

```

```

</thead>
<tbody>
    @foreach (var appointment in HistoryAppointments)
    {
        <tr>
            <td>@appointment.Date.ToShortDateString()</td>
            <td>@appointment.Time</td>
            <td>@appointment.Service</td>
            <td>@appointment.Master</td>
            <td>
                @if (appointment != null && !Reviews.Any(r
=> r.Client == CurrentUser.Login && r.Master == appointment.Master
&& r.Service == appointment.Service))
                {
                    <button class="btn btn-sm btn-
secondary" @onclick="() => ShowReviewForm(appointment)">Оставить
отзыв</button>
                }
            </td>
        </tr>
    }
</tbody>
</table>
}
else
{
    <p>Нет истории записей</p>
}

@if (ShowReviewModal)
{
    <div class="modal fade show" style="display: block;"
role="dialog">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title">Отзыв</h5>
                    <button type="button" class="close"
@onclick="CloseReviewForm">
                        <span aria-hidden="true">×</span>
                    </button>
                </div>
                <div class="modal-body">
                    <div class="form-group">
                        <label>Рейтинг</label>
                        <select class="form-control"
@bind="ReviewValue">
                            <option value="1">1</option>
                            <option value="2">2</option>
                            <option value="3">3</option>
                            <option value="4">4</option>
                            <option value="5">5</option>
                        </select>

```

```

        </div>
        <div class="form-group">
            <label>Текст отзыва</label>
            <textarea class="form-control"
@bind="ReviewText"></textarea>
        </div>
    </div>
    <div class="modal-footer">
        <button class="btn"
@onclick="AddReview">Отправить отзыв</button>
    </div>
</div>
</div>
</div>
}

@code {
    private User CurrentUser { get; set; } = new User();
    private string? ReviewText { get; set; }
    private int ReviewValue { get; set; }
    private bool ShowReviewModal { get; set; }
    private Appointment? SelectedAppointment { get; set; }
    private List<Appointment>? ActiveAppointments { get; set; }
    private List<Appointment>? PendingAppointments { get; set; }
    private List<Appointment>? CanceledAppointments { get; set; }
    private List<Appointment>? HistoryAppointments { get; set; }
    private List<Review>? Reviews { get; set; }

    protected override void OnInitialized()
    {
        var users =
DataService.ReadAllLines("users@SQL$TABLE").Select(u => new
User(u)).ToList();
        //имитация логина, так как логина пока нет
        CurrentUser = users.FirstOrDefault(u => u.Login == "test")
?? new User();
        if (CurrentUser == null)
            return;
        var appointments =
DataService.ReadAllLines("rasp@SQL$TABLE").Select(u => new
Appointment(u)).ToList();
        ActiveAppointments = appointments.Where(a => a.Client ==
CurrentUser.Login && a.Date >= DateTime.Today && a.Status ==
"Утверждена").ToList();
        PendingAppointments = appointments.Where(a => a.Client ==
CurrentUser.Login && a.Date >= DateTime.Today && a.Status ==
"Ожидает утверждения").ToList();
        CanceledAppointments = appointments.Where(a => a.Client ==
CurrentUser.Login && a.Date >= DateTime.Today && a.Status ==
"Отменена").ToList();
        HistoryAppointments = appointments.Where(a => a.Client ==
CurrentUser.Login && a.Date < DateTime.Today).ToList();
    }
}

```

```

        Reviews =
DataService.ReadAllLines("reviews@SQL$TABLE").Select(r => new
Review(r)).ToList();
    }

    private void ShowReviewForm(Appointment appointment)
    {
        SelectedAppointment = appointment;
        ShowReviewModal = true;
    }

    private void CloseReviewForm()
    {
        ShowReviewModal = false;
    }

    private void AddReview()
    {
        var reviews =
DataService.ReadAllLines("reviews@SQL$TABLE");
        if (SelectedAppointment != null)

reviews.Add($"{CurrentUser.Login},{SelectedAppointment.Master},{Se
lectedAppointment.Service},{ReviewValue},{ReviewText},{ReviewValue
}");
        DataService.WriteAllLines("reviews@SQL$TABLE", reviews);
        ShowReviewModal = false;
        Reviews = reviews.Select(r => new Review(r)).ToList();
        StateHasChanged();
    }

    private void SaveProfile()
    {
        var users = DataService.ReadAllLines("users@SQL$TABLE");
        var currentLine = users.FirstOrDefault(u =>
u.Split(',')[0] == CurrentUser.Login);
        if (currentLine != null)
            users.Remove(currentLine);

users.Add($"{CurrentUser.Login},{CurrentUser.Password},{CurrentUse
r.Name},{CurrentUser.PhoneNumber}");
        DataService.WriteAllLines("users@SQL$TABLE", users);
    }
}

ServiceList.cs:
@using kurs.Services
@using kurs.Models
@Inject DataService DataService

<h3 style="text-align: center;">Список услуг</h3>
<div class="card-container">
    @foreach (var service in Services)

```



```

        <th>Рейтинг</th>
        <th>Отзыв</th>
    </tr>
</thead>
<tbody>
    @foreach (var review in Reviews)
    {
        <tr>
            <td>@review.Client</td>
            <td>@review.Master</td>
            <td>@review.Value</td>
            <td>@review.Text</td>
        </tr>
    }
</tbody>
</table>
    }
    else
    {
        <p>Нет отзывов</p>
    }
</div>
</div>
}

<style>
    .card-container {
        display: flex;
        flex-wrap: wrap;
        gap: 20px;
        justify-content: center;
    }

    .card {
        width: 280px;
        padding: 20px;
        border: 1px solid #e0e0e0;
        border-radius: 8px;
        cursor: pointer;
        background-color: white;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
        transition: transform 0.2s ease;
    }

    .card:hover {
        transform: translateY(-5px);
    }

    .detail-modal {
        position: fixed;
        top: 0;
        left: 0;
        width: 100%;

```

```

        height: 100%;
        background-color: rgba(0, 0, 0, 0.5);
        display: flex;
        align-items: center;
        justify-content: center;
        z-index: 1000;
    }
</style>

@code {
    private List<Service>? Services { get; set; }
    private Service? SelectedService { get; set; }
    private List<Review>? Reviews { get; set; }
    private bool ShowServiceDetail { get; set; }
    private Dictionary<string, List<string>> MasterServices { get;
set; } = new Dictionary<string, List<string>>();

    protected override void OnInitialized()
    {
        Services =
DataService.ReadAllLines("uslugi@SQL$TABLE").Select(s => new
Service(s)).ToList();
        MasterServices = DataService.GetMasterServices();
    }

    private void ShowServiceDetails(Service service)
    {
        SelectedService = service;
        Reviews =
DataService.ReadAllLines("reviews@SQL$TABLE").Select(r => new
Review(r)).Where(r => r.Service == service.Name).ToList();
        ShowServiceDetail = true;
    }

    private void CloseServiceDetails()
    {
        ShowServiceDetail = false;
    }
}

MasterList.cs:
@using kurs.Services
@using kurs.Models
@Inject DataService DataService

<h3 style="text-align: center;">Список мастеров</h3>
<div class="card-container">
    @foreach (var master in Masters)
    {
        <div class="card"
@onclick="()=>ShowMasterDetails(master)">
            <h4>@master.FirstName @master.LastName</h4>
            <p>Специализация: @master.Specialization</p>
            <p>Услуги:</p>

```

```

        <ul>
            @if
            (MasterServices.ContainsKey($"{master.FirstName}
            {master.LastName}"))
            {
                @foreach (var service in
                MasterServices[$"{master.FirstName} {master.LastName}"])
                {
                    <li>@service</li>
                }
            }
        </ul>
    </div>
}
</div>

@if (ShowMasterDetail)
{
    <div class="detail-modal">
        <div class="modal-content">
            <button class="close-button"
            @onclick="CloseMasterDetails">Назад</button>
            <h2>Информация о мастере: @SelectedMaster?.FirstName
            @SelectedMaster?.LastName</h2>
            <p>Стаж: @SelectedMaster?.Experience лет</p>
            <p>Специализация: @SelectedMaster?.Specialization</p>
            <p>Услуги:</p>
            <ul>
                @if
                (MasterServices.ContainsKey($"{SelectedMaster?.FirstName}
                {SelectedMaster?.LastName}"))
                {
                    @foreach (var service in
                    MasterServices[$"{SelectedMaster?.FirstName}
                    {SelectedMaster?.LastName}"])
                    {
                        <li>@service</li>
                    }
                }
            </ul>

            <h3>Портфолио</h3>
            @if (Portfolio != null && Portfolio.Count > 0)
            {
                <table class="table">
                    <thead>
                        <tr>
                            <th>Название</th>
                            <th>Описание</th>
                            <th>Услуга</th>
                        </tr>
                    </thead>
                    <tbody>

```

```

        @foreach (var item in Portfolio)
        {
            <tr>
                <td>@item.Name</td>
                <td>@item.Text</td>
                <td>@item.Service</td>
            </tr>
        }
    </tbody>
</table>
}
else
{
    <p>Нет портфолио</p>
}

<h3>ОТЗЫВЫ</h3>
@if (Reviews != null && Reviews.Count > 0)
{
    <table class="table">
        <thead>
            <tr>
                <th>Клиент</th>
                <th>Услуга</th>
                <th>Рейтинг</th>
                <th>ОТЗЫВ</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var review in Reviews)
            {
                <tr>
                    <td>@review.Client</td>
                    <td>@review.Service</td>
                    <td>@review.Value</td>
                    <td>@review.Text</td>
                </tr>
            }
        </tbody>
    </table>
}
else
{
    <p>Нет отзывов</p>
}
</div>
</div>
}

<style>
    .card-container {
        display: flex;
        flex-wrap: wrap;
    }

```

```

        gap: 20px;
        justify-content: center;
    }

    .card {
        width: 280px;
        padding: 20px;
        border: 1px solid #e0e0e0;
        border-radius: 8px;
        cursor: pointer;
        background-color: white;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
        transition: transform 0.2s ease;
    }

    .card:hover {
        transform: translateY(-5px);
    }

    .detail-modal {
        position: fixed;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        background-color: rgba(0, 0, 0, 0.5);
        display: flex;
        align-items: center;
        justify-content: center;
        z-index: 1000;
    }
}
</style>

@code {
    private List<Master>? Masters { get; set; }
    private Master? SelectedMaster { get; set; }
    private List<Portfolio>? Portfolio { get; set; }
    private List<Review>? Reviews { get; set; }
    private bool ShowMasterDetail { get; set; }
    private Dictionary<string, List<string>> MasterServices { get;
set; } = new Dictionary<string, List<string>>();

    protected override void OnInitialized()
    {
        Masters =
DataService.ReadAllLines("master@SQL$TABLE").Select(m => new
Master(m)).ToList();
        MasterServices = DataService.GetMasterServices();
    }

    private void ShowMasterDetails(Master master)
    {
        SelectedMaster = master;
    }
}

```



```

        {
            <option
value="@service.Name">@service.Name</option>
        }
    </select>
</div>
    <button class="btn" @onclick="() => Progress = 2"
disabled="@ (string.IsNullOrEmpty(SelectedServiceName)) ">Далее</but
ton>
}
else if (Progress == 2)
{
    <h4>Выберите мастера</h4>
    <div class="form-group">
        <select class="form-control"
@bind="SelectedMasterName" id="masterSelect">
            <option value="">Выберите мастера</option>
            @foreach (var master in AvailableMasters)
            {
                <option value="@($"{master.FirstName}
{master.LastName})" ">@master.FirstName @master.LastName</option>
            }
        </select>
    </div>
    <div class="button-group">
        <button class="btn" @onclick="() => Progress =
1">Назад</button>
        <button class="btn" @onclick="() => Progress = 3"
disabled="@ (string.IsNullOrEmpty(SelectedMasterName)) ">Далее</butt
on>
    </div>
}
else if (Progress == 3)
{
    <h4>Выберите дату</h4>
    <div class="form-group">
        <input type="date" class="form-control"
@bind="SelectedDate" @bind:event="onchange" id="dateInput" />
    </div>
    <div class="button-group">
        <button class="btn" @onclick="() => Progress =
2">Назад</button>
        <button class="btn" @onclick="() => Progress = 4"
disabled="@ (SelectedDate == null) ">Далее</button>
    </div>
}
else if (Progress == 4)
{
    <h4>Выберите время</h4>
    <div class="form-group">
        <select class="form-control" @bind="SelectedTime">
            <option value="">Выберите время</option>
            @foreach (var time in AvailableTimes)

```

```

        {
            <option value="@time">@time</option>
        }
    </select>
</div>
<div class="button-group">
    <button class="btn" @onclick="() => Progress =
3">Назад</button>
    <button class="btn" @onclick="MakeAppointment"
disabled="@ (string.IsNullOrEmpty(SelectedTime)) ">Записаться</butto
n>
</div>
}
</div>
</div>

@if (!string.IsNullOrEmpty(Message))
{
    <div class="alert alert-success">@Message</div>
}

<style>
    .progress-container {
        width: 100%;
        height: 30px;
        background-color: #f0f0f0;
        border-radius: 15px;
        margin-bottom: 20px;
        position: relative;
        overflow: hidden;
    }

    .progress-bar {
        height: 100%;
        background: linear-gradient(to right, #3498db, #9b59b6);
        transition: width 0.3s ease;
    }

    .progress-steps {
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        display: flex;
        justify-content: space-between;
        width: 80%;
    }

    .progress-steps span {
        width: 30px;
        height: 30px;
        border-radius: 50%;
        background-color: #fff;
    }

```

```

        border: 2px solid #ddd;
        display: flex;
        align-items: center;
        justify-content: center;
        color: #555;
        font-weight: bold;
        transition: background-color 0.3s ease, color 0.3s ease;
    }

    .progress-steps span.active {
        background-color: #3498db;
        color: white;
        border-color: #3498db;
    }

    .card-container {
        display: flex;
        justify-content: center;
    }

    .appointment-card {
        width: 400px;
        padding: 30px;
        border: 1px solid #e0e0e0;
        border-radius: 8px;
        background-color: white;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    }

    .button-group {
        display: flex;
        justify-content: space-between;
        margin-top: 20px;
    }
</style>

@code {
    private List<Master>? Masters { get; set; }
    private List<Service>? Services { get; set; }
    private DateTime SelectedDate { get; set; } = DateTime.Today;
    private string? SelectedTime { get; set; }
    private string? SelectedMasterName { get; set; }
    private string? SelectedServiceName { get; set; }
    private string? Message { get; set; }
    private List<string> AvailableTimes { get; set; } = new
List<string>
    {
        "10:00", "11:00", "12:00", "13:00", "14:00", "15:00", "16:00", "17:00", "1
8:00"
    };
    private List<Master> AvailableMasters { get; set; } = new
List<Master>();

```

```

        private Dictionary<string, List<string>> MasterServices { get;
set; } = new Dictionary<string, List<string>>();
        private List<Appointment> Appointments { get; set; } = new
List<Appointment>();
        private int Progress { get; set; } = 1;
        private DotNetObjectReference<AppointmentForm>? dotNetHelper;

        protected override void OnInitialized()
        {
            Masters =
DataService.ReadAllLines("master@SQL$TABLE").Select(m => new
Master(m)).ToList();
            Services =
DataService.ReadAllLines("uslugi@SQL$TABLE").Select(s => new
Service(s)).ToList();
            MasterServices = DataService.GetMasterServices();
            Appointments =
DataService.ReadAllLines("rasp@SQL$TABLE").Select(a => new
Appointment(a)).ToList();
            dotNetHelper = DotNetObjectReference.Create(this);
            UpdateAvailableMasters();
        }

        protected override async Task OnAfterRenderAsync(bool
firstRender)
        {
            if(firstRender)
            {
                await
JSRuntime.InvokeVoidAsync("updateAvailableMasters","serviceSelect"
, dotNetHelper);
                await
JSRuntime.InvokeVoidAsync("updateAvailableTimes","masterSelect",
dotNetHelper);
            }
        }

        [JSInvokable]
        public void UpdateAvailableMastersJS()
        {
            UpdateAvailableMasters();
            StateHasChanged();
        }

        [JSInvokable]
        public void UpdateAvailableTimesJS()
        {
            UpdateAvailableTimes();
            StateHasChanged();
        }

```

```

private void UpdateAvailableMasters()
{
    if (!string.IsNullOrEmpty(SelectedServiceName))
    {
        AvailableMasters = Masters.Where(m =>
MasterServices.ContainsKey($"{m.FirstName} {m.LastName}") &&
MasterServices[$"{m.FirstName}
{m.LastName}"].Contains(SelectedServiceName)).ToList();
    }
    else
    {
        AvailableMasters = new List<Master>();
    }
    SelectedMasterName = null;
    UpdateAvailableTimes();
}
private void UpdateAvailableTimes()
{
    if (!string.IsNullOrEmpty(SelectedMasterName) &&
SelectedDate != null)
    {
        var bookedTimes = Appointments
            .Where(a => a.Master == SelectedMasterName &&
a.Date.Date == SelectedDate.Date)
            .Select(a => a.Time)
            .ToList();

        AvailableTimes =
AvailableTimes.Except(bookedTimes).ToList();
    }
    else
    {
        AvailableTimes = new List<string>
        {
            "10:00", "11:00", "12:00", "13:00", "14:00", "15:00", "16:00", "17:00", "1
8:00"
        };
    }
    SelectedTime = null;
}

private void MakeAppointment()
{
    if (string.IsNullOrEmpty(SelectedMasterName) ||
string.IsNullOrEmpty(SelectedTime) ||
string.IsNullOrEmpty(SelectedServiceName))
    {
        Message = "Выберите мастера, услугу и время";
        return;
    }
}

```

```

        var appointments =
DataService.ReadAllLines("rasp@SQL$TABLE");
        if (appointments.Any(a =>
a.Contains($"{SelectedDate.ToShortDateString()}, {SelectedTime}, {Se
lectedServiceName}, {SelectedMasterName}")))
        {
            Message = "Это время занято, выберите другое";
            return;
        }
        var users =
DataService.ReadAllLines("users@SQL$TABLE").Select(u => new
User(u)).ToList();
        var user = users.FirstOrDefault(u => u.Login == "test");
        if (user == null)
            return;

appointments.Add($"{SelectedDate.ToShortDateString()}, {SelectedTim
e}, {SelectedServiceName}, {SelectedMasterName}, {user.Login}, Ожидает
утверждения");
        DataService.WriteAllLines("rasp@SQL$TABLE", appointments);
        Message = "Запись прошла успешно!";
        Progress = 1;
        SelectedServiceName = null;
        SelectedMasterName = null;
        SelectedDate = DateTime.Today;
        SelectedTime = null;
    }
}
App.js:
function updateAvailableMasters(elementId, dotNetHelper) {
    const serviceSelect =
document.getElementById('serviceSelect');
    if (serviceSelect) {
        const selectedService = serviceSelect.value;
        dotNetHelper.invokeMethodAsync('UpdateAvailableMastersJS',
selectedService);
    }
}

function updateAvailableTimes(elementId, dotNetHelper) {
    const masterSelect = document.getElementById('masterSelect');
    const dateInput = document.getElementById('dateInput');
    if (masterSelect && dateInput) {
        const selectedMaster = masterSelect.value;
        const selectedDate = dateInput.value;
        dotNetHelper.invokeMethodAsync('UpdateAvailableTimesJS',
selectedMaster, selectedDate);
    }
}

function validatePassword(elementId, password2, dotNetHelper) {
    const strengthMeter = document.getElementById(elementId +
'StrengthMeter');

```

```

let strength = 0;
let requirements = [];

const password = document.getElementById(elementId).value;

if (!password) {
    strengthMeter.style.width = '0%';
    strengthMeter.style.backgroundColor = 'red';
    dotNetHelper.invokeMethodAsync("SetRequirements", []);
    return;
}
if (password.length >= 6) {
    strength += 25;
} else {
    requirements.push("Пароль от 6 символов");
}

if (/[a-z]/.test(password)) {
    strength += 25;
} else {
    requirements.push("Должна быть хоть одна маленькая
буква");
}

if (/[A-Z]/.test(password)) {
    strength += 25;
} else {
    requirements.push("Должна быть хоть одна прописная
буква");
}

if (/[0-9]/.test(password)) {
    strength += 25;
} else {
    requirements.push("Должна быть хоть одна цифра");
}

strength = Math.min(strength, 100);

let color = 'red';
if (strength > 40 && strength < 100) {
    color = 'orange';
} else if (strength === 100)
    color = 'green';

strengthMeter.style.width = strength + '%';
strengthMeter.style.backgroundColor = color;
dotNetHelper.invokeMethodAsync("SetRequirements",
requirements);
}

```

```
function validateConfirmPassword(passwordId, confirmPasswordId,
dotNetHelper) {
    const password = document.getElementById(passwordId).value;
    const confirmPassword =
document.getElementById(confirmPasswordId).value;
    if (password !== confirmPassword)
        dotNetHelper.invokeMethodAsync("SetConfirmPasswordError",
"Пароли не совпадают");
    else
        dotNetHelper.invokeMethodAsync("SetConfirmPasswordError",
null);
}
```

App.css:

```
/* Общие стили для всего приложения */
body {
    font-family: 'Roboto', sans-serif; /* Используем шрифт Roboto */
    margin: 0;
    padding: 0;
    background-color: #f5f5f5; /* Светло-серый фон */
    color: #333; /* Темно-серый цвет текста */
    line-height: 1.6;
}

h1 {
    font-size: 2.5rem; /* 40px */
    margin-bottom: 20px;
    text-align: center;
    color: #2c3e50; /* Темно-синий цвет заголовка */
}

h2 {
    font-size: 2rem; /* 32px */
    margin-bottom: 15px;
    color: #2c3e50;
}

h3 {
    font-size: 1.5rem; /* 24px */
    margin-bottom: 10px;
    color: #2c3e50;
}

p, label, a, li {
    font-size: 1rem; /* 16px */
    line-height: 1.6;
}

/* Стили для таблиц */
.table {
    width: 100%;
    border-collapse: collapse;
```

```

margin-bottom: 20px;
border: 1px solid #e0e0e0; /* Светло-серый цвет границ */
box-shadow: 0 2px 4px rgba(0, 0, 0, 0.05); /* Легкая тень */
background-color: white;
}

.table th, .table td {
    padding: 12px;
    border: 1px solid #e0e0e0;
    text-align: left;
}

.table th {
    background-color: #f0f0f0; /* Светло-серый цвет фона
заголовка */
    font-weight: 500; /* Средняя жирность */
    color: #555; /* Темно-серый цвет текста заголовка */
}

/* Стили для кнопок */
.btn {
    padding: 12px 24px;
    border: none;
    border-radius: 4px;
    color: white;
    cursor: pointer;
    background: linear-gradient(to right, #3498db, #9b59b6); /*
Градиент синего и фиолетового */
    transition: background 0.3s ease;
    font-weight: 500;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.btn:hover {
    background: linear-gradient(to right, #2980b9, #8e44ad);
/* Более темный градиент при наведении */
}

/* Стили для ссылок */
a {
    color: #3498db; /* Синий цвет ссылок */
    text-decoration: none;
    transition: color 0.3s ease;
}

a:hover {
    text-decoration: underline;
    color: #2980b9; /* Более темный синий при наведении */
}

/* Стили для форм */
.form-group {
    margin-bottom: 15px;

```

```

}

.form-control {
  width: 100%;
  padding: 12px;
  border: 1px solid #e0e0e0;
  border-radius: 4px;
  box-sizing: border-box;
  transition: border-color 0.3s ease;
}

.form-control:focus {
  border-color: #3498db; /* Синий цвет при фокусе */
  outline: none;
  box-shadow: 0 0 5px rgba(52, 152, 219, 0.3);
}

/* Стили для модальных окон */
.detail-modal {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.5);
  display: flex;
  align-items: center;
  justify-content: center;
  z-index: 1000;
}

.modal-content {
  background-color: white;
  padding: 30px;
  border-radius: 8px;
  position: relative;
  max-width: 80%;
  max-height: 80%;
  overflow-y: auto;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.close-button {
  position: absolute;
  top: 15px;
  left: 15px;
  cursor: pointer;
  background: none;
  border: none;
  font-size: 1.2rem;
  color: #555;
  transition: color 0.3s ease;
}

```

```

        .close-button:hover {
            color: #333;
        }

/* Стили для карточек */
.card-container {
    display: flex;
    flex-wrap: wrap;
    gap: 20px;
    justify-content: center;
}

.card {
    width: 280px;
    padding: 20px;
    border: 1px solid #e0e0e0;
    border-radius: 8px;
    cursor: pointer;
    background-color: white;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    transition: transform 0.2s ease;
}

.card:hover {
    transform: translateY(-5px);
}

#blazor-error-ui {
    background: lightyellow;
    bottom: 0;
    box-shadow: 0 -1px 2px rgba(0, 0, 0, 0.2);
    display: none;
    left: 0;
    padding: 0.6rem 1.25rem 0.7rem 1.25rem;
    position: fixed;
    width: 100%;
    z-index: 1000;
}

#blazor-error-ui .dismiss {
    cursor: pointer;
    position: absolute;
    right: 3.5rem;
    top: 0.5rem;
}

.blazor-error-boundary {

```



```

        {
            case ComponentType.Profile:
                <UserProfile />
                break;
            case ComponentType.Masters:
                <MasterList />
                break;
            case ComponentType.Services:
                <ServiceList />
                break;
            case ComponentType.Appointment:
                <AppointmentForm />
                break;
            default:
                <div>Выберите пункт меню</div>
                break;
        }
    </div>
</div>

<style>
    .main-container {
        display: flex;
        min-height: 100vh;
    }

    .sidebar {
        width: 250px;
        background: linear-gradient(to bottom, #2c3e50, #3498db);
        /* Градиент темно-синего и синего */
        color: white;
        display: flex;
        flex-direction: column;
    }

    .menu-item {
        padding: 18px;
        cursor: pointer;
        transition: background-color 0.3s ease;
        font-size: 1.1rem;
    }

    .menu-item:hover {
        background-color: rgba(255, 255, 255, 0.1);
    }

    .content {
        flex: 1;
        padding: 30px;
    }
</style>

@code {

```

```

        private enum ComponentType { Profile, Masters, Services,
Appointment }
        private ComponentType CurrentComponent { get; set; }

        private void ShowComponent(ComponentType componentType)
        {
            CurrentComponent = componentType;
        }
    }
}

```

LoginPage.cs:

```

@page "/login"
@using kurs.Services
@using kurs.Models
@Inject NavigationManager NavigationManager
@Inject DataService DataService

<h1 style="text-align: center;">Авторизация</h1>

<div class="form-group">
    <label>Логин</label>
    <input type="text" class="form-control" @bind="Login" />
</div>
<div class="form-group">
    <label>Пароль</label>
    <input type="password" class="form-control" @bind="Password"
/>
</div>

<button class="btn" @onclick="HandleLogin">Войти</button>
<br />
<a href="/register">Если нет аккаунта, то зарегистрироваться</a>

@if (!string.IsNullOrEmpty(ErrorMessage))
{
    <div class="alert alert-danger">@ErrorMessage</div>
}

@code {
    private string? Login { get; set; }
    private string? Password { get; set; }
    private string? ErrorMessage { get; set; }

    private void HandleLogin()
    {
        var users =
DataService.ReadAllLines("users@SQL$TABLE").Select(u => new
User(u)).ToList();
        var user = users.FirstOrDefault(u => u.Login == Login &&
u.Password == Password);

        if (user != null)
        {

```

```

        NavigationManager.NavigateTo("/main");
    }
    else
    {
        ErrorMessage = "Неверный логин или пароль";
    }
}
}

```

User.cs:

```

namespace kurs.Models
{
    public class User
    {
        public string? Login { get; set; }
        public string? Password { get; set; }
        public string? Name { get; set; }
        public string? PhoneNumber { get; set; }

        public User(string line)
        {
            string[] parts = line.Split(',');
            if (parts.Length == 4)
            {
                Login = parts[0];
                Password = parts[1];
                Name = parts[2];
                PhoneNumber = parts[3];
            }
        }

        public User() { }
    }
}

```

Service.cs:

```

namespace kurs.Models
{
    public class Service
    {
        public string? Name { get; set; }
        public string? Type { get; set; }
        public string? Description { get; set; }
        public decimal Price { get; set; }

        public Service(string line)
        {
            string[] parts = line.Split(',');
            if (parts.Length == 4)
            {
                Name = parts[0];
                Type = parts[1];
            }
        }
    }
}

```

```

        Description = parts[2];
        Price = decimal.Parse(parts[3]);
    }
}

public Service() { }
}
}

```

Review.cs:

```

namespace kurs.Models
{
    public class Review
    {
        public string? Client { get; set; }
        public string? Master { get; set; }
        public string? Service { get; set; }
        public int Rating { get; set; }
        public string? Text { get; set; }
        public int Value { get; set; }

        public Review(string line)
        {
            string[] parts = line.Split(',');
            if (parts.Length == 6)
            {
                Client = parts[0];
                Master = parts[1];
                Service = parts[2];
                Rating = int.Parse(parts[3]);
                Text = parts[4];
                Value = int.Parse(parts[5]);
            }
        }

        public Review() { }
    }
}

```

Portfolio.cs:

```

namespace kurs.Models
{
    public class Portfolio
    {
        public string? Name { get; set; }
        public string? Text { get; set; }
        public string? Master { get; set; }
        public string? Service { get; set; }

        public Portfolio(string line)
        {

```

```

        string[] parts = line.Split(',');
        if (parts.Length == 4)
        {
            Name = parts[0];
            Text = parts[1];
            Master = parts[2];
            Service = parts[3];
        }
    }

    public Portfolio() { }
}

Master.cs:
namespace kurs.Models
{
    public class Master
    {
        public string? FirstName { get; set; }
        public string? LastName { get; set; }
        public int Experience { get; set; }
        public string? Specialization { get; set; }

        public Master(string line)
        {
            string[] parts = line.Split(',');
            if (parts.Length == 4)
            {
                FirstName = parts[0];
                LastName = parts[1];
                Experience = int.Parse(parts[2]);
                Specialization = parts[3];
            }
        }

        public Master() { }
    }
}

```

App.cs:

```

using System;

namespace kurs.Models
{
    public class Appointment
    {
        public DateTime Date { get; set; }
        public string? Time { get; set; }
        public string? Service { get; set; }
        public string? Master { get; set; }
    }
}

```

```

        public string? Client { get; set; }
        public string? Status { get; set; } // Добавляем свойство
        Status

        public Appointment(string line)
        {
            string[] parts = line.Split(',');
            if (parts.Length == 6)
            {
                Date = DateTime.Parse(parts[0]);
                Time = parts[1];
                Service = parts[2];
                Master = parts[3];
                Client = parts[4];
                Status = parts[5]; // Инициализируем Status
            }
        }

        public Appointment() { }
    }
}

```

UserProdile.cs:

```

@using kurs.Services
@using kurs.Models
@Inject DataService DataService

<h3 style="text-align: center;">Профиль</h3>
<div class="form-group">
    <label>Логин</label>
    <input type="text" class="form-control"
@bind="CurrentUser.Login" />
</div>
<div class="form-group">
    <label>Пароль</label>
    <input type="text" class="form-control"
@bind="CurrentUser.Password" />
</div>
<div class="form-group">
    <label>Имя</label>
    <input type="text" class="form-control"
@bind="CurrentUser.Name" />
</div>
<div class="form-group">
    <label>Номер телефона</label>
    <input type="text" class="form-control"
@bind="CurrentUser.PhoneNumber" />
</div>
<button class="btn" @onclick="SaveProfile">Сохранить</button>

<h3 style="text-align: center;">Активные записи</h3>
@if (ActiveAppointments != null && ActiveAppointments.Count > 0)
{

```

```

<table class="table">
  <thead>
    <tr>
      <th>Дата</th>
      <th>Время</th>
      <th>Услуга</th>
      <th>Мастер</th>
      <th>Статус</th>
    </tr>
  </thead>
  <tbody>
    @foreach (var appointment in ActiveAppointments)
    {
      <tr>
        <td>@appointment.Date.ToShortDateString()</td>
        <td>@appointment.Time</td>
        <td>@appointment.Service</td>
        <td>@appointment.Master</td>
        <td>@appointment.Status</td>
      </tr>
    }
  </tbody>
</table>
}
else
{
  <p>Нет активных записей</p>
}

<h3 style="text-align: center;">Ожидают утверждения</h3>
@if (PendingAppointments != null && PendingAppointments.Count > 0)
{
  <table class="table">
    <thead>
      <tr>
        <th>Дата</th>
        <th>Время</th>
        <th>Услуга</th>
        <th>Мастер</th>
        <th>Статус</th>
      </tr>
    </thead>
    <tbody>
      @foreach (var appointment in PendingAppointments)
      {
        <tr>
          <td>@appointment.Date.ToShortDateString()</td>
          <td>@appointment.Time</td>
          <td>@appointment.Service</td>
          <td>@appointment.Master</td>
          <td>@appointment.Status</td>
        </tr>
      }
    }
  }
}

```

```

        </tbody>
    </table>
}
else
{
    <p>Нет записей, ожидающих утверждения</p>
}

<h3 style="text-align: center;">Отмененные записи</h3>
@if (CanceledAppointments != null && CanceledAppointments.Count >
0)
{
    <table class="table">
        <thead>
            <tr>
                <th>Дата</th>
                <th>Время</th>
                <th>Услуга</th>
                <th>Мастер</th>
                <th>Статус</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var appointment in CanceledAppointments)
            {
                <tr>
                    <td>@appointment.Date.ToShortDateString()</td>
                    <td>@appointment.Time</td>
                    <td>@appointment.Service</td>
                    <td>@appointment.Master</td>
                    <td>@appointment.Status</td>
                </tr>
            }
        </tbody>
    </table>
}
else
{
    <p>Нет отмененных записей</p>
}

<h3 style="text-align: center;">История записей</h3>
@if (HistoryAppointments != null && HistoryAppointments.Count > 0)
{
    <table class="table">
        <thead>
            <tr>
                <th>Дата</th>
                <th>Время</th>
                <th>Услуга</th>
                <th>Мастер</th>
                <th></th>
            </tr>

```

```

</thead>
<tbody>
    @foreach (var appointment in HistoryAppointments)
    {
        <tr>
            <td>@appointment.Date.ToShortDateString()</td>
            <td>@appointment.Time</td>
            <td>@appointment.Service</td>
            <td>@appointment.Master</td>
            <td>
                @if (appointment != null && !Reviews.Any(r
=> r.Client == CurrentUser.Login && r.Master == appointment.Master
&& r.Service == appointment.Service))
                {
                    <button class="btn btn-sm btn-
secondary" @onclick="() => ShowReviewForm(appointment)">Оставить
отзыв</button>
                }
            </td>
        </tr>
    }
</tbody>
</table>
}
else
{
    <p>Нет истории записей</p>
}

@if (ShowReviewModal)
{
    <div class="modal fade show" style="display: block;"
role="dialog">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title">Отзыв</h5>
                    <button type="button" class="close"
@onclick="CloseReviewForm">
                        <span aria-hidden="true">x</span>
                    </button>
                </div>
                <div class="modal-body">
                    <div class="form-group">
                        <label>Рейтинг</label>
                        <select class="form-control"
@bind="ReviewValue">
                            <option value="1">1</option>
                            <option value="2">2</option>
                            <option value="3">3</option>
                            <option value="4">4</option>
                            <option value="5">5</option>
                        </select>

```

```

        </div>
        <div class="form-group">
            <label>Текст отзыва</label>
            <textarea class="form-control"
@bind="ReviewText"></textarea>
        </div>
    </div>
    <div class="modal-footer">
        <button class="btn"
@onclick="AddReview">Отправить отзыв</button>
    </div>
</div>
</div>
</div>
}

@code {
    private User CurrentUser { get; set; } = new User();
    private string? ReviewText { get; set; }
    private int ReviewValue { get; set; }
    private bool ShowReviewModal { get; set; }
    private Appointment? SelectedAppointment { get; set; }
    private List<Appointment>? ActiveAppointments { get; set; }
    private List<Appointment>? PendingAppointments { get; set; }
    private List<Appointment>? CanceledAppointments { get; set; }
    private List<Appointment>? HistoryAppointments { get; set; }
    private List<Review>? Reviews { get; set; }

    protected override void OnInitialized()
    {
        var users =
DataService.ReadAllLines("users@SQL$TABLE").Select(u => new
User(u)).ToList();
        //имитация логина, так как логина пока нет
        CurrentUser = users.FirstOrDefault(u => u.Login == "test")
?? new User();
        if (CurrentUser == null)
            return;
        var appointments =
DataService.ReadAllLines("rasp@SQL$TABLE").Select(u => new
Appointment(u)).ToList();
        ActiveAppointments = appointments.Where(a => a.Client ==
CurrentUser.Login && a.Date >= DateTime.Today && a.Status ==
"Утверждена").ToList();
        PendingAppointments = appointments.Where(a => a.Client ==
CurrentUser.Login && a.Date >= DateTime.Today && a.Status ==
"Ожидает утверждения").ToList();
        CanceledAppointments = appointments.Where(a => a.Client ==
CurrentUser.Login && a.Date >= DateTime.Today && a.Status ==
"Отменена").ToList();
        HistoryAppointments = appointments.Where(a => a.Client ==
CurrentUser.Login && a.Date < DateTime.Today).ToList();
    }
}

```

```

        Reviews =
DataService.ReadAllLines("reviews@SQL$TABLE").Select(r => new
Review(r)).ToList();
    }

    private void ShowReviewForm(Appointment appointment)
    {
        SelectedAppointment = appointment;
        ShowReviewModal = true;
    }

    private void CloseReviewForm()
    {
        ShowReviewModal = false;
    }

    private void AddReview()
    {
        var reviews =
DataService.ReadAllLines("reviews@SQL$TABLE");
        if (SelectedAppointment != null)

reviews.Add($"{CurrentUser.Login},{SelectedAppointment.Master},{Se
lectedAppointment.Service},{ReviewValue},{ReviewText},{ReviewValue
}");
        DataService.WriteAllLines("reviews@SQL$TABLE", reviews);
        ShowReviewModal = false;
        Reviews = reviews.Select(r => new Review(r)).ToList();
        StateHasChanged();
    }

    private void SaveProfile()
    {
        var users = DataService.ReadAllLines("users@SQL$TABLE");
        var currentLine = users.FirstOrDefault(u =>
u.Split(',')[0] == CurrentUser.Login);
        if (currentLine != null)
            users.Remove(currentLine);

users.Add($"{CurrentUser.Login},{CurrentUser.Password},{CurrentUse
r.Name},{CurrentUser.PhoneNumber}");
        DataService.WriteAllLines("users@SQL$TABLE", users);
    }
}

ServiceList.cs:
@using kurs.Services
@using kurs.Models
@Inject DataService DataService

<h3 style="text-align: center;">Список услуг</h3>
<div class="card-container">
    @foreach (var service in Services)

```



```

        <th>Рейтинг</th>
        <th>Отзыв</th>
    </tr>
</thead>
<tbody>
    @foreach (var review in Reviews)
    {
        <tr>
            <td>@review.Client</td>
            <td>@review.Master</td>
            <td>@review.Value</td>
            <td>@review.Text</td>
        </tr>
    }
</tbody>
</table>
    }
    else
    {
        <p>Нет отзывов</p>
    }
</div>
</div>
}

<style>
    .card-container {
        display: flex;
        flex-wrap: wrap;
        gap: 20px;
        justify-content: center;
    }

    .card {
        width: 280px;
        padding: 20px;
        border: 1px solid #e0e0e0;
        border-radius: 8px;
        cursor: pointer;
        background-color: white;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
        transition: transform 0.2s ease;
    }

    .card:hover {
        transform: translateY(-5px);
    }

    .detail-modal {
        position: fixed;
        top: 0;
        left: 0;
        width: 100%;

```

```

        height: 100%;
        background-color: rgba(0, 0, 0, 0.5);
        display: flex;
        align-items: center;
        justify-content: center;
        z-index: 1000;
    }
</style>

@code {
    private List<Service>? Services { get; set; }
    private Service? SelectedService { get; set; }
    private List<Review>? Reviews { get; set; }
    private bool ShowServiceDetail { get; set; }
    private Dictionary<string, List<string>> MasterServices { get;
set; } = new Dictionary<string, List<string>>();

    protected override void OnInitialized()
    {
        Services =
DataService.ReadAllLines("uslugi@SQL$TABLE").Select(s => new
Service(s)).ToList();
        MasterServices = DataService.GetMasterServices();
    }

    private void ShowServiceDetails(Service service)
    {
        SelectedService = service;
        Reviews =
DataService.ReadAllLines("reviews@SQL$TABLE").Select(r => new
Review(r)).Where(r => r.Service == service.Name).ToList();
        ShowServiceDetail = true;
    }

    private void CloseServiceDetails()
    {
        ShowServiceDetail = false;
    }
}
MasterList.cs:
@using kurs.Services
@using kurs.Models
@Inject DataService DataService

<h3 style="text-align: center;">Список мастеров</h3>
<div class="card-container">
    @foreach (var master in Masters)
    {
        <div class="card"
@onclick="()=>ShowMasterDetails(master)">
            <h4>@master.FirstName @master.LastName</h4>
            <p>Специализация: @master.Specialization</p>
            <p>Услуги:</p>

```

```

        <ul>
            @if
            (MasterServices.ContainsKey($"{master.FirstName}
            {master.LastName}"))
            {
                @foreach (var service in
                MasterServices[$"{master.FirstName} {master.LastName}"])
                {
                    <li>@service</li>
                }
            }
        </ul>
    </div>
}
</div>

@if (ShowMasterDetail)
{
    <div class="detail-modal">
        <div class="modal-content">
            <button class="close-button"
            @onclick="CloseMasterDetails">Назад</button>
            <h2>Информация о мастере: @SelectedMaster?.FirstName
            @SelectedMaster?.LastName</h2>
            <p>Стаж: @SelectedMaster?.Experience лет</p>
            <p>Специализация: @SelectedMaster?.Specialization</p>
            <p>Услуги:</p>
            <ul>
                @if
                (MasterServices.ContainsKey($"{SelectedMaster?.FirstName}
                {SelectedMaster?.LastName}"))
                {
                    @foreach (var service in
                    MasterServices[$"{SelectedMaster?.FirstName}
                    {SelectedMaster?.LastName}"])
                    {
                        <li>@service</li>
                    }
                }
            </ul>

            <h3>Портфолио</h3>
            @if (Portfolio != null && Portfolio.Count > 0)
            {
                <table class="table">
                    <thead>
                        <tr>
                            <th>Название</th>
                            <th>Описание</th>
                            <th>Услуга</th>
                        </tr>
                    </thead>
                    <tbody>

```

```

        @foreach (var item in Portfolio)
        {
            <tr>
                <td>@item.Name</td>
                <td>@item.Text</td>
                <td>@item.Service</td>
            </tr>
        }
    </tbody>
</table>
}
else
{
    <p>Нет портфолио</p>
}

<h3>ОТЗЫВЫ</h3>
@if (Reviews != null && Reviews.Count > 0)
{
    <table class="table">
        <thead>
            <tr>
                <th>Клиент</th>
                <th>Услуга</th>
                <th>Рейтинг</th>
                <th>ОТЗЫВ</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var review in Reviews)
            {
                <tr>
                    <td>@review.Client</td>
                    <td>@review.Service</td>
                    <td>@review.Value</td>
                    <td>@review.Text</td>
                </tr>
            }
        </tbody>
    </table>
}
else
{
    <p>Нет отзывов</p>
}
</div>
</div>
}

<style>
    .card-container {
        display: flex;
        flex-wrap: wrap;
    }

```

```

        gap: 20px;
        justify-content: center;
    }

    .card {
        width: 280px;
        padding: 20px;
        border: 1px solid #e0e0e0;
        border-radius: 8px;
        cursor: pointer;
        background-color: white;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
        transition: transform 0.2s ease;
    }

    .card:hover {
        transform: translateY(-5px);
    }

    .detail-modal {
        position: fixed;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        background-color: rgba(0, 0, 0, 0.5);
        display: flex;
        align-items: center;
        justify-content: center;
        z-index: 1000;
    }
}
</style>

@code {
    private List<Master>? Masters { get; set; }
    private Master? SelectedMaster { get; set; }
    private List<Portfolio>? Portfolio { get; set; }
    private List<Review>? Reviews { get; set; }
    private bool ShowMasterDetail { get; set; }
    private Dictionary<string, List<string>> MasterServices { get;
set; } = new Dictionary<string, List<string>>();

    protected override void OnInitialized()
    {
        Masters =
DataService.ReadAllLines("master@SQL$TABLE").Select(m => new
Master(m)).ToList();
        MasterServices = DataService.GetMasterServices();
    }

    private void ShowMasterDetails(Master master)
    {
        SelectedMaster = master;
    }
}

```



```

        {
            <option
value="@service.Name">@service.Name</option>
        }
    </select>
</div>
    <button class="btn" @onclick="() => Progress = 2"
disabled="@ (string.IsNullOrEmpty(SelectedServiceName)) ">Далее</but
ton>
}
else if (Progress == 2)
{
    <h4>Выберите мастера</h4>
    <div class="form-group">
        <select class="form-control"
@bind="SelectedMasterName" id="masterSelect">
            <option value="">Выберите мастера</option>
            @foreach (var master in AvailableMasters)
            {
                <option value="@($"{master.FirstName}
{master.LastName})" ">@master.FirstName @master.LastName</option>
            }
        </select>
    </div>
    <div class="button-group">
        <button class="btn" @onclick="() => Progress =
1">Назад</button>
        <button class="btn" @onclick="() => Progress = 3"
disabled="@ (string.IsNullOrEmpty(SelectedMasterName)) ">Далее</butt
on>
    </div>
}
else if (Progress == 3)
{
    <h4>Выберите дату</h4>
    <div class="form-group">
        <input type="date" class="form-control"
@bind="SelectedDate" @bind:event="onchange" id="dateInput" />
    </div>
    <div class="button-group">
        <button class="btn" @onclick="() => Progress =
2">Назад</button>
        <button class="btn" @onclick="() => Progress = 4"
disabled="@ (SelectedDate == null) ">Далее</button>
    </div>
}
else if (Progress == 4)
{
    <h4>Выберите время</h4>
    <div class="form-group">
        <select class="form-control" @bind="SelectedTime">
            <option value="">Выберите время</option>
            @foreach (var time in AvailableTimes)

```

```

        {
            <option value="@time">@time</option>
        }
    </select>
</div>
<div class="button-group">
    <button class="btn" @onclick="() => Progress =
3">Назад</button>
    <button class="btn" @onclick="MakeAppointment"
disabled="@ (string.IsNullOrEmpty(SelectedTime)) ">Записаться</butto
n>
</div>
}
</div>
</div>

@if (!string.IsNullOrEmpty(Message))
{
    <div class="alert alert-success">@Message</div>
}

<style>
    .progress-container {
        width: 100%;
        height: 30px;
        background-color: #f0f0f0;
        border-radius: 15px;
        margin-bottom: 20px;
        position: relative;
        overflow: hidden;
    }

    .progress-bar {
        height: 100%;
        background: linear-gradient(to right, #3498db, #9b59b6);
        transition: width 0.3s ease;
    }

    .progress-steps {
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        display: flex;
        justify-content: space-between;
        width: 80%;
    }

    .progress-steps span {
        width: 30px;
        height: 30px;
        border-radius: 50%;
        background-color: #fff;
    }

```

```

        border: 2px solid #ddd;
        display: flex;
        align-items: center;
        justify-content: center;
        color: #555;
        font-weight: bold;
        transition: background-color 0.3s ease, color 0.3s ease;
    }

    .progress-steps span.active {
        background-color: #3498db;
        color: white;
        border-color: #3498db;
    }

    .card-container {
        display: flex;
        justify-content: center;
    }

    .appointment-card {
        width: 400px;
        padding: 30px;
        border: 1px solid #e0e0e0;
        border-radius: 8px;
        background-color: white;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    }

    .button-group {
        display: flex;
        justify-content: space-between;
        margin-top: 20px;
    }
</style>

@code {
    private List<Master>? Masters { get; set; }
    private List<Service>? Services { get; set; }
    private DateTime SelectedDate { get; set; } = DateTime.Today;
    private string? SelectedTime { get; set; }
    private string? SelectedMasterName { get; set; }
    private string? SelectedServiceName { get; set; }
    private string? Message { get; set; }
    private List<string> AvailableTimes { get; set; } = new
List<string>
    {
        "10:00", "11:00", "12:00", "13:00", "14:00", "15:00", "16:00", "17:00", "1
8:00"
    };
    private List<Master> AvailableMasters { get; set; } = new
List<Master>();

```

```

        private Dictionary<string, List<string>> MasterServices { get;
set; } = new Dictionary<string, List<string>>();
        private List<Appointment> Appointments { get; set; } = new
List<Appointment>();
        private int Progress { get; set; } = 1;
        private DotNetObjectReference<AppointmentForm>? dotNetHelper;

        protected override void OnInitialized()
        {
            Masters =
DataService.ReadAllLines("master@SQL$TABLE").Select(m => new
Master(m)).ToList();
            Services =
DataService.ReadAllLines("uslugi@SQL$TABLE").Select(s => new
Service(s)).ToList();
            MasterServices = DataService.GetMasterServices();
            Appointments =
DataService.ReadAllLines("rasp@SQL$TABLE").Select(a => new
Appointment(a)).ToList();
            dotNetHelper = DotNetObjectReference.Create(this);
            UpdateAvailableMasters();
        }

        protected override async Task OnAfterRenderAsync(bool
firstRender)
        {
            if(firstRender)
            {
                await
JSRuntime.InvokeVoidAsync("updateAvailableMasters","serviceSelect"
, dotNetHelper);
                await
JSRuntime.InvokeVoidAsync("updateAvailableTimes","masterSelect",
dotNetHelper);
            }
        }

        [JSInvokable]
        public void UpdateAvailableMastersJS()
        {
            UpdateAvailableMasters();
            StateHasChanged();
        }

        [JSInvokable]
        public void UpdateAvailableTimesJS()
        {
            UpdateAvailableTimes();
            StateHasChanged();
        }

```

```

private void UpdateAvailableMasters()
{
    if (!string.IsNullOrEmpty(SelectedServiceName))
    {
        AvailableMasters = Masters.Where(m =>
MasterServices.ContainsKey($"{m.FirstName} {m.LastName}") &&
MasterServices[$"{m.FirstName}
{m.LastName}"].Contains(SelectedServiceName)).ToList();
    }
    else
    {
        AvailableMasters = new List<Master>();
    }
    SelectedMasterName = null;
    UpdateAvailableTimes();
}
private void UpdateAvailableTimes()
{
    if (!string.IsNullOrEmpty(SelectedMasterName) &&
SelectedDate != null)
    {
        var bookedTimes = Appointments
            .Where(a => a.Master == SelectedMasterName &&
a.Date.Date == SelectedDate.Date)
            .Select(a => a.Time)
            .ToList();

        AvailableTimes =
AvailableTimes.Except(bookedTimes).ToList();
    }
    else
    {
        AvailableTimes = new List<string>
        {
            "10:00", "11:00", "12:00", "13:00", "14:00", "15:00", "16:00", "17:00", "1
8:00"
        };
    }
    SelectedTime = null;
}

private void MakeAppointment()
{
    if (string.IsNullOrEmpty(SelectedMasterName) ||
string.IsNullOrEmpty(SelectedTime) ||
string.IsNullOrEmpty(SelectedServiceName))
    {
        Message = "Выберите мастера, услугу и время";
        return;
    }
}

```

```

        var appointments =
DataService.ReadAllLines("rasp@SQL$TABLE");
        if (appointments.Any(a =>
a.Contains($"{SelectedDate.ToShortDateString()}, {SelectedTime}, {Se
lectedServiceName}, {SelectedMasterName}")))
        {
            Message = "Это время занято, выберите другое";
            return;
        }
        var users =
DataService.ReadAllLines("users@SQL$TABLE").Select(u => new
User(u)).ToList();
        var user = users.FirstOrDefault(u => u.Login == "test");
        if (user == null)
            return;

appointments.Add($"{SelectedDate.ToShortDateString()}, {SelectedTim
e}, {SelectedServiceName}, {SelectedMasterName}, {user.Login}, Ожидает
утверждения");
        DataService.WriteAllLines("rasp@SQL$TABLE", appointments);
        Message = "Запись прошла успешно!";
        Progress = 1;
        SelectedServiceName = null;
        SelectedMasterName = null;
        SelectedDate = DateTime.Today;
        SelectedTime = null;
    }
}
App.js:
function updateAvailableMasters(elementId, dotNetHelper) {
    const serviceSelect =
document.getElementById('serviceSelect');
    if (serviceSelect) {
        const selectedService = serviceSelect.value;
        dotNetHelper.invokeMethodAsync('UpdateAvailableMastersJS',
selectedService);
    }
}

function updateAvailableTimes(elementId, dotNetHelper) {
    const masterSelect = document.getElementById('masterSelect');
    const dateInput = document.getElementById('dateInput');
    if (masterSelect && dateInput) {
        const selectedMaster = masterSelect.value;
        const selectedDate = dateInput.value;
        dotNetHelper.invokeMethodAsync('UpdateAvailableTimesJS',
selectedMaster, selectedDate);
    }
}

function validatePassword(elementId, password2, dotNetHelper) {
    const strengthMeter = document.getElementById(elementId +
'StrengthMeter');

```

```

let strength = 0;
let requirements = [];

const password = document.getElementById(elementId).value;

if (!password) {
    strengthMeter.style.width = '0%';
    strengthMeter.style.backgroundColor = 'red';
    dotNetHelper.invokeMethodAsync("SetRequirements", []);
    return;
}
if (password.length >= 6) {
    strength += 25;
} else {
    requirements.push("Пароль от 6 символов");
}

if (/[a-z]/.test(password)) {
    strength += 25;
} else {
    requirements.push("Должна быть хоть одна маленькая
буква");
}

if (/[A-Z]/.test(password)) {
    strength += 25;
} else {
    requirements.push("Должна быть хоть одна прописная
буква");
}

if (/[0-9]/.test(password)) {
    strength += 25;
} else {
    requirements.push("Должна быть хоть одна цифра");
}

strength = Math.min(strength, 100);

let color = 'red';
if (strength > 40 && strength < 100) {
    color = 'orange';
} else if (strength === 100)
    color = 'green';

strengthMeter.style.width = strength + '%';
strengthMeter.style.backgroundColor = color;
dotNetHelper.invokeMethodAsync("SetRequirements",
requirements);
}

```

```

function validateConfirmPassword(passwordId, confirmPasswordId,
dotNetHelper) {
    const password = document.getElementById(passwordId).value;
    const confirmPassword =
document.getElementById(confirmPasswordId).value;
    if (password != confirmPassword)
        dotNetHelper.invokeMethodAsync("SetConfirmPasswordError",
"Пароли не совпадают");
    else
        dotNetHelper.invokeMethodAsync("SetConfirmPasswordError",
null);
}
MainPage.cs:
@page "/main"
@using kurs.Components

<div class="main-container">
    <div class="sidebar">
        <div class="menu-item" @onclick="() =>
ShowComponent(ComponentType.Profile)">Профиль</div>
        <div class="menu-item" @onclick="() =>
ShowComponent(ComponentType.Masters)">Мастера</div>
        <div class="menu-item" @onclick="() =>
ShowComponent(ComponentType.Services)">Услуги</div>
        <div class="menu-item" @onclick="() =>
ShowComponent(ComponentType.Appointment)">Записаться на
прием</div>
    </div>

    <div class="content">
        @switch (CurrentComponent)
        {
            case ComponentType.Profile:
                <UserProfile />
                break;
            case ComponentType.Masters:
                <MasterList />
                break;
            case ComponentType.Services:
                <ServiceList />
                break;
            case ComponentType.Appointment:
                <AppointmentForm />
                break;
            default:
                <div>Выберите пункт меню</div>
                break;
        }
    </div>
</div>

<style>
    .main-container {

```

```

        display: flex;
        min-height: 100vh;
    }

    .sidebar {
        width: 250px;
        background: linear-gradient(to bottom, #2c3e50, #3498db);
        /* Градиент темно-синего и синего */
        color: white;
        display: flex;
        flex-direction: column;
    }

    .menu-item {
        padding: 18px;
        cursor: pointer;
        transition: background-color 0.3s ease;
        font-size: 1.1rem;
    }

    .menu-item:hover {
        background-color: rgba(255, 255, 255, 0.1);
    }

    .content {
        flex: 1;
        padding: 30px;
    }
}
</style>

```

```

@code {
    private enum ComponentType { Profile, Masters, Services,
Appointment }
    private ComponentType CurrentComponent { get; set; }

    private void ShowComponent(ComponentType componentType)
    {
        CurrentComponent = componentType;
    }
}

```

```

LoginPage.cs:
@page "/login"
@using kurs.Services
@using kurs.Models
@Inject NavigationManager NavigationManager
@Inject DataService DataService

<h1 style="text-align: center;">Авторизация</h1>

<div class="form-group">
    <label>Логин</label>
    <input type="text" class="form-control" @bind="Login" />

```

```

</div>
<div class="form-group">
    <label>Пароль</label>
    <input type="password" class="form-control" @bind="Password"
/>
</div>

```

```

<button class="btn" @onclick="HandleLogin">Войти</button>
<br />
<a href="/register">Если нет аккаунта, то зарегистрироваться</a>

```

```

@if (!string.IsNullOrEmpty(ErrorMessage))
{
    <div class="alert alert-danger">@ErrorMessage</div>
}

```

```

@code {
    private string? Login { get; set; }
    private string? Password { get; set; }
    private string? ErrorMessage { get; set; }

    private void HandleLogin()
    {
        var users =
DataService.ReadAllLines("users@SQL$TABLE").Select(u => new
User(u)).ToList();
        var user = users.FirstOrDefault(u => u.Login == Login &&
u.Password == Password);

        if (user != null)
        {
            NavigationManager.NavigateTo("/main");
        }
        else
        {
            ErrorMessage = "Неверный логин или пароль";
        }
    }
}

```

User.cs:

```

namespace kurs.Models
{
    public class User
    {
        public string? Login { get; set; }
        public string? Password { get; set; }
        public string? Name { get; set; }
        public string? PhoneNumber { get; set; }

        public User(string line)
        {

```

```

        string[] parts = line.Split(',');
        if (parts.Length == 4)
        {
            Login = parts[0];
            Password = parts[1];
            Name = parts[2];
            PhoneNumber = parts[3];
        }
    }

    public User() { }
}

```

Service.cs:

```

namespace kurs.Models
{
    public class Service
    {
        public string? Name { get; set; }
        public string? Type { get; set; }
        public string? Description { get; set; }
        public decimal Price { get; set; }

        public Service(string line)
        {
            string[] parts = line.Split(',');
            if (parts.Length == 4)
            {
                Name = parts[0];
                Type = parts[1];
                Description = parts[2];
                Price = decimal.Parse(parts[3]);
            }
        }

        public Service() { }
    }
}

```

Review.cs:

```

namespace kurs.Models
{
    public class Review
    {
        public string? Client { get; set; }
        public string? Master { get; set; }
        public string? Service { get; set; }
        public int Rating { get; set; }
        public string? Text { get; set; }
        public int Value { get; set; }
    }
}

```

```

        public Review(string line)
        {
            string[] parts = line.Split(',');
            if (parts.Length == 6)
            {
                Client = parts[0];
                Master = parts[1];
                Service = parts[2];
                Rating = int.Parse(parts[3]);
                Text = parts[4];
                Value = int.Parse(parts[5]);
            }
        }

        public Review() { }
    }
}

```

```

Portfolio.cs:
namespace kurs.Models
{
    public class Portfolio
    {
        public string? Name { get; set; }
        public string? Text { get; set; }
        public string? Master { get; set; }
        public string? Service { get; set; }

        public Portfolio(string line)
        {
            string[] parts = line.Split(',');
            if (parts.Length == 4)
            {
                Name = parts[0];
                Text = parts[1];
                Master = parts[2];
                Service = parts[3];
            }
        }

        public Portfolio() { }
    }
}

```

```

Master.cs:
namespace kurs.Models
{
    public class Master
    {
        public string? FirstName { get; set; }
        public string? LastName { get; set; }
        public int Experience { get; set; }
    }
}

```

```

        public string? Specialization { get; set; }

        public Master(string line)
        {
            string[] parts = line.Split(',');
            if (parts.Length == 4)
            {
                FirstName = parts[0];
                LastName = parts[1];
                Experience = int.Parse(parts[2]);
                Specialization = parts[3];
            }
        }

        public Master() { }
    }
}

App.cs:

using System;

namespace kurs.Models
{
    public class Appointment
    {
        public DateTime Date { get; set; }
        public string? Time { get; set; }
        public string? Service { get; set; }
        public string? Master { get; set; }
        public string? Client { get; set; }
        public string? Status { get; set; } // Добавляем свойство
        Status

        public Appointment(string line)
        {
            string[] parts = line.Split(',');
            if (parts.Length == 6)
            {
                Date = DateTime.Parse(parts[0]);
                Time = parts[1];
                Service = parts[2];
                Master = parts[3];
                Client = parts[4];
                Status = parts[5]; // Инициализируем Status
            }
        }

        public Appointment() { }
    }
}

```

```

UserProdile.cs:
@using kurs.Services
@using kurs.Models
@Inject DataService DataService

<h3 style="text-align: center;">Профиль</h3>
<div class="form-group">
    <label>Логин</label>
    <input type="text" class="form-control"
@bind="CurrentUser.Login" />
</div>
<div class="form-group">
    <label>Пароль</label>
    <input type="text" class="form-control"
@bind="CurrentUser.Password" />
</div>
<div class="form-group">
    <label>Имя</label>
    <input type="text" class="form-control"
@bind="CurrentUser.Name" />
</div>
<div class="form-group">
    <label>Номер телефона</label>
    <input type="text" class="form-control"
@bind="CurrentUser.PhoneNumber" />
</div>
<button class="btn" @onclick="SaveProfile">Сохранить</button>

<h3 style="text-align: center;">Активные записи</h3>
@if (ActiveAppointments != null && ActiveAppointments.Count > 0)
{
    <table class="table">
        <thead>
            <tr>
                <th>Дата</th>
                <th>Время</th>
                <th>Услуга</th>
                <th>Мастер</th>
                <th>Статус</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var appointment in ActiveAppointments)
            {
                <tr>
                    <td>@appointment.Date.ToShortDateString()</td>
                    <td>@appointment.Time</td>
                    <td>@appointment.Service</td>
                    <td>@appointment.Master</td>
                    <td>@appointment.Status</td>
                </tr>
            }
        </tbody>
    </table>
}

```

```

        </table>
    }
    else
    {
        <p>Нет активных записей</p>
    }

    <h3 style="text-align: center;">Ожидают утверждения</h3>
    @if (PendingAppointments != null && PendingAppointments.Count > 0)
    {
        <table class="table">
            <thead>
                <tr>
                    <th>Дата</th>
                    <th>Время</th>
                    <th>Услуга</th>
                    <th>Мастер</th>
                    <th>Статус</th>
                </tr>
            </thead>
            <tbody>
                @foreach (var appointment in PendingAppointments)
                {
                    <tr>
                        <td>@appointment.Date.ToShortDateString()</td>
                        <td>@appointment.Time</td>
                        <td>@appointment.Service</td>
                        <td>@appointment.Master</td>
                        <td>@appointment.Status</td>
                    </tr>
                }
            </tbody>
        </table>
    }
    else
    {
        <p>Нет записей, ожидающих утверждения</p>
    }

    <h3 style="text-align: center;">Отмененные записи</h3>
    @if (CanceledAppointments != null && CanceledAppointments.Count >
0)
    {
        <table class="table">
            <thead>
                <tr>
                    <th>Дата</th>
                    <th>Время</th>
                    <th>Услуга</th>
                    <th>Мастер</th>
                    <th>Статус</th>
                </tr>
            </thead>

```

```

        <tbody>
            @foreach (var appointment in CanceledAppointments)
            {
                <tr>
                    <td>@appointment.Date.ToShortDateString()</td>
                    <td>@appointment.Time</td>
                    <td>@appointment.Service</td>
                    <td>@appointment.Master</td>
                    <td>@appointment.Status</td>
                </tr>
            }
        </tbody>
    </table>
}
else
{
    <p>Нет отмененных записей</p>
}

<h3 style="text-align: center;">История записей</h3>
@if (HistoryAppointments != null && HistoryAppointments.Count > 0)
{
    <table class="table">
        <thead>
            <tr>
                <th>Дата</th>
                <th>Время</th>
                <th>Услуга</th>
                <th>Мастер</th>
                <th></th>
            </tr>
        </thead>
        <tbody>
            @foreach (var appointment in HistoryAppointments)
            {
                <tr>
                    <td>@appointment.Date.ToShortDateString()</td>
                    <td>@appointment.Time</td>
                    <td>@appointment.Service</td>
                    <td>@appointment.Master</td>
                    <td>
                        @if (appointment != null && !Reviews.Any(r
=> r.Client == CurrentUser.Login && r.Master == appointment.Master
&& r.Service == appointment.Service))
                        {
                            <button class="btn btn-sm btn-
secondary" @onclick="() => ShowReviewForm(appointment)">Оставить
отзыв</button>
                        }
                    </td>
                </tr>
            }
        </tbody>
    }
}

```

```

        </table>
    }
    else
    {
        <p>Нет истории записей</p>
    }

    @if (ShowReviewModal)
    {
        <div class="modal fade show" style="display: block;"
        role="dialog">
            <div class="modal-dialog">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title">Отзыв</h5>
                        <button type="button" class="close"
@onclick="CloseReviewForm">
                            <span aria-hidden="true">×</span>
                        </button>
                    </div>
                    <div class="modal-body">
                        <div class="form-group">
                            <label>Рейтинг</label>
                            <select class="form-control"
@bind="ReviewValue">
                                <option value="1">1</option>
                                <option value="2">2</option>
                                <option value="3">3</option>
                                <option value="4">4</option>
                                <option value="5">5</option>
                            </select>
                        </div>
                        <div class="form-group">
                            <label>Текст отзыва</label>
                            <textarea class="form-control"
@bind="ReviewText"></textarea>
                        </div>
                    </div>
                    <div class="modal-footer">
                        <button class="btn"
@onclick="AddReview">Отправить отзыв</button>
                    </div>
                </div>
            </div>
        </div>
    }

```

Приложение Г
(обязательное)

Исходный код мобильного приложения

Листинг Г.1 – Клиентское приложение

```

Sss
com\example\marmar\MainActivity.kt
-----
package com.example.marmar

import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.ImageView
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.GravityCompat
import androidx.drawerlayout.widget.DrawerLayout
import androidx.lifecycle.LifecycleScope
import androidx.navigation.NavController
import androidx.navigation.fragment.NavHostFragment
import androidx.navigation.ui.AppBarConfiguration
import androidx.navigation.ui.navigateUp
import androidx.navigation.ui.setupActionBarWithNavController
import androidx.navigation.ui.setupWithNavController
import com.example.marmar.data.model.UserRole
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.ActivityMainBinding
import kotlinx.coroutines.launch
import java.io.File

class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding
    private lateinit var navController: NavController
    private lateinit var appBarConfiguration: AppBarConfiguration

    override fun onCreate(savedInstanceState: Bundle?) {

AppCompatActivity.setDefaultNightMode(AppCompatActivity.MODE_NIGHT
_NO)

        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        lifecycleScope.launch {
            DataRepository.loadData(applicationContext)
        }

        setupNavigation()
        setupUIChangesOnNavigation()
    }

    private fun setupNavigation() {

```

```

        setSupportActionBar(binding.toolbar)

        val navHostFragment = supportFragmentManager
            .findFragmentById(R.id.nav_host_fragment) as
NavHostFragment

        navController = navHostFragment.navController

        val navGraph =
navController.navInflater.inflate(R.navigation.nav_graph)
        navGraph.setStartDestination(R.id.loginFragment)
        navController.graph = navGraph

        appBarConfiguration = AppBarConfiguration(
            setOf(
                R.id.clientHomeFragment, R.id.masterHomeFragment,
R.id.adminHomeFragment
            ),
            binding.drawerLayout
        )

        setupActionBarWithNavController(navController,
appBarConfiguration)
        binding.navView.setupWithNavController(navController)

        binding.navView.setNavigationItemSelectedListener {
menuItem ->
            when (menuItem.itemId) {
                R.id.nav_logout -> {
                    logout()
                    true
                }
                R.id.nav_master_cabinet -> {
                    DataRepository.currentUser?.id?.let { masterId
->
                        val action =
NavGraphDirections.actionGlobalNavMasterCabinet(masterId)
                        navController.navigate(action)
                    }
                }
            }
            binding.drawerLayout.closeDrawer(GravityCompat.START)
            true
        }
        else -> {
            navController.navigate(menuItem.itemId)
        }
        binding.drawerLayout.closeDrawer(GravityCompat.START)
        true
    }
}
}
}

```

```

        private fun setupUIChangesOnNavigation() {
            navController.addOnDestinationChangedListener { _,
destination, _ ->
                when (destination.id) {
                    R.id.loginFragment, R.id.registerFragment -> {
                        binding.toolbar.visibility = View.GONE

binding.drawerLayout.setDrawerLockMode(DrawerLayout.LOCK_MODE_LOCK
ED_CLOSED)
                    }
                    else -> {
                        binding.toolbar.visibility = View.VISIBLE

binding.drawerLayout.setDrawerLockMode(DrawerLayout.LOCK_MODE_UNLO
CKED)

                        updateDrawerBasedOnUserRole()
                        updateNavHeader()
                    }
                }
            }
        }

        private fun updateDrawerBasedOnUserRole() {
            val user = DataRepository.currentUser
            binding.navView.menu.clear()
            if (user != null) {
                when (user.role) {
                    UserRole.CLIENT ->
binding.navView.inflateMenu(R.menu.client_menu)
                    UserRole.MASTER ->
binding.navView.inflateMenu(R.menu.master_menu)
                    UserRole.ADMIN ->
binding.navView.inflateMenu(R.menu.admin_menu)
                }
            }
            binding.navView.menu.findItem(R.id.nav_logout)?.isVisible
= user != null
        }

        private fun updateNavHeader() {
            val headerView: View = binding.navView.getHeaderView(0)
            val userImageView: ImageView =
headerView.findViewById(R.id.nav_header_imageView)
            val userInfoTextView: TextView =
headerView.findViewById(R.id.nav_header_login_id)
            val headerClickableArea: View =
headerView.findViewById(R.id.header_clickable_area)

            DataRepository.currentUser?.let { user ->
                userInfoTextView.text = "${user.login} (ID:
${user.id.take(6)}...)"

                if (!user.photoPath.isNullOrEmpty()) {

```

```

        val photoFile = File(user.photoPath!!)
        if (photoFile.exists()) {

userImageView.setImageURI(android.net.Uri.fromFile(photoFile))
        } else {

userImageView.setImageResource(R.drawable.ic_default_avatar)
        }
        } else {

userImageView.setImageResource(R.drawable.ic_default_avatar)
        }

        // --- ИЗМЕНЕНИЯ ЗДЕСЬ ---
        headerClickableArea.setOnClickListener {
            if (navController.currentDestination?.id !=
R.id.profileEditFragment) {
                // Используем глобальный action и передаем ID
ТЕКУЩЕГО пользователя
                val action =
NavGraphDirections.actionGlobalProfileEditFragment(user.id)
                navController.navigate(action)
            }

binding.drawerLayout.closeDrawer(GravityCompat.START)
        }
        // --- КОНЕЦ ИЗМЕНЕНИЙ ---
    }

    private fun logout() {
        DataRepository.currentUser = null
        navController.navigate(R.id.action_global_logout)
        binding.drawerLayout.closeDrawer(GravityCompat.START)
        Log.i("MainActivity", "Пользователь вышел из системы.")
    }

    override fun onSupportNavigateUp(): Boolean {
        return navController.navigateUp(appBarConfiguration) ||
super.onSupportNavigateUp()
    }

    override fun onStop() {
        super.onStop()
        lifecycleScope.launch {
            DataRepository.saveData(applicationContext)
            Log.i("MainActivity", "Данные сохранены при выходе из
приложения.")
        }
    }
}

```

=====

```

com\example\marmar\adapter\AdminAccountAdapter.kt
-----
package com.example.marmar.adapter

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.marmar.data.model.User
import com.example.marmar.databinding.ItemAdminAccountBinding

class AdminAccountAdapter(
    private var users: List<User>,
    private val listener: AccountInteractionListener
) : RecyclerView.Adapter<AdminAccountAdapter.AccountViewHolder>()
{

    interface AccountInteractionListener {
        fun onEditClicked(user: User)
        fun onChangeRoleClicked(user: User)
        fun onDeleteClicked(user: User)
    }

    inner class AccountViewHolder(val binding:
ItemAdminAccountBinding) : RecyclerView.ViewHolder(binding.root)

        override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): AccountViewHolder {
            val binding =
ItemAdminAccountBinding.inflate(LayoutInflater.from(parent.context
), parent, false)
            return AccountViewHolder(binding)
        }

        override fun onBindViewHolder(holder: AccountViewHolder,
position: Int) {
            val user = users[position]
            with(holder.binding) {
                userLogin.text = user.login
                userFullName.text = user.fullName ?: "Профиль не
заполнен"
                userRole.text = user.role.name

                editButton.setOnClickListener {
listener.onEditClicked(user) }
                changeRoleButton.setOnClickListener {
listener.onChangeRoleClicked(user) }
                deleteButton.setOnClickListener {
listener.onDeleteClicked(user) }
            }
        }

        override fun getItemCount() = users.size

```

```

        fun updateData(newUsers: List<User>) {
            this.users = newUsers
            notifyDataSetChanged()
        }
    }
}

=====

com\example\marmar\adapter\AdminAppointmentAdapter.kt
-----
package com.example.marmar.adapter

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.marmar.R
import com.example.marmar.data.model.Appointment
import com.example.marmar.data.model.AppointmentStatus
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.ItemAdminAppointmentBinding

class AdminAppointmentAdapter(
    private var appointments: List<Appointment>,
    private val listener: AppointmentInteractionListener
) :
    RecyclerView.Adapter<AdminAppointmentAdapter.AppointmentViewHolder>() {

    interface AppointmentInteractionListener {
        fun onEditClicked(appointment: Appointment)
        fun onDeleteClicked(appointment: Appointment)
    }

    class AppointmentViewHolder(val binding:
        ItemAdminAppointmentBinding) :
        RecyclerView.ViewHolder(binding.root)

        override fun onCreateViewHolder(parent: ViewGroup, viewType:
            Int): AppointmentViewHolder {
            val binding =
                ItemAdminAppointmentBinding.inflate(LayoutInflater.from(parent.con
                    text), parent, false)
            return AppointmentViewHolder(binding)
        }

        override fun onBindViewHolder(holder: AppointmentViewHolder,
            position: Int) {
            val appointment = appointments[position]
            val context = holder.itemView.context
            with(holder.binding) {

```

```

        serviceName.text =
DataRepository.getServiceById(appointment.serviceId)?.name ?:
"Услуга удалена"
        clientName.text = "Клиент:
${DataRepository.getUserById(appointment.clientId)?.fullName ?:
"Клиент удален"}"
        masterName.text = "Мастер:
${DataRepository.getUserById(appointment.masterId)?.fullName ?:
"Мастер удален"}"
        dateTimeText.text = "${appointment.getFormattedDate()}
в ${appointment.getFormattedTime()}"

        // Устанавливаем и текст, и цвет на основе статуса
        val (statusText, statusColor) =
when(appointment.status) {
    AppointmentStatus.PENDING_CONFIRMATION ->
Pair(context.getString(R.string.status_pending),
R.color.purple_accent)
    AppointmentStatus.AWAITING_PAYMENT ->
Pair(context.getString(R.string.status_awaiting_payment),
R.color.pink)
    AppointmentStatus.CONFIRMED ->
Pair(context.getString(R.string.status_confirmed),
R.color.success_green)
    AppointmentStatus.COMPLETED ->
Pair(context.getString(R.string.status_completed),
R.color.divider)
    AppointmentStatus.CANCELLED ->
Pair(context.getString(R.string.status_cancelled),
R.color.error_red)
}
        statusChip.text = statusText
        statusChip.setChipBackgroundColorResource(statusColor)

        editButton.setOnClickListener {
listener.onEditClicked(appointment) }
        deleteButton.setOnClickListener {
listener.onDeleteClicked(appointment) }
    }

    override fun getItemCount(): Int = appointments.size

    fun updateData(newAppointments: List<Appointment>) {
        this.appointments = newAppointments
        notifyDataSetChanged()
    }
}

=====

com\example\marmar\adapter\AdminMasterAdapter.kt
-----

```

```

package com.example.marmar.adapter

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.marmar.data.model.User
import com.example.marmar.databinding.ItemAdminMasterBinding

class AdminMasterAdapter(
    private var masters: List<User>,
    private val listener: MasterInteractionListener
) : RecyclerView.Adapter<AdminMasterAdapter.MasterViewHolder>() {

    interface MasterInteractionListener {
        fun onEditClicked(master: User)
    }

    class MasterViewHolder(val binding: ItemAdminMasterBinding) :
        RecyclerView.ViewHolder(binding.root)

    override fun onCreateViewHolder(parent: ViewGroup, viewType:
        Int): MasterViewHolder {
        val binding =
            ItemAdminMasterBinding.inflate(LayoutInflater.from(parent.context)
            , parent, false)
        return MasterViewHolder(binding)
    }

    override fun onBindViewHolder(holder: MasterViewHolder,
        position: Int) {
        val master = masters[position]
        with(holder.binding) {
            masterName.text = master.fullName
            masterSpecialization.text = master.specialization
            editButton.setOnClickListener {
                listener.onEditClicked(master)
            }
        }
    }

    override fun getItemCount(): Int = masters.size

    fun updateData(newMasters: List<User>) {
        this.masters = newMasters
        notifyDataSetChanged()
    }
}

=====

com\example\marmar\adapter\AdminReviewAdapter.kt
-----
package com.example.marmar.adapter

```

```

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.marmar.R
import com.example.marmar.data.model.Review
import com.example.marmar.data.model.ReviewTargetType
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.ItemAdminReviewBinding

class AdminReviewAdapter(
    private var reviews: List<Review>,
    private val listener: ReviewInteractionListener
) : RecyclerView.Adapter<AdminReviewAdapter.ReviewViewHolder>() {

    interface ReviewInteractionListener {
        fun onDeleteClicked(review: Review)
    }

    class ReviewViewHolder(val binding: ItemAdminReviewBinding) :
        RecyclerView.ViewHolder(binding.root)

    override fun onCreateViewHolder(parent: ViewGroup, viewType:
        Int): ReviewViewHolder {
        val binding =
            ItemAdminReviewBinding.inflate(LayoutInflater.from(parent.context)
            , parent, false)
        return ReviewViewHolder(binding)
    }

    override fun onBindViewHolder(holder: ReviewViewHolder,
        position: Int) {
        val review = reviews[position]
        val context = holder.itemView.context
        with(holder.binding) {
            reviewRatingBar.rating = review.rating.toFloat()
            reviewText.text = review.text
            reviewAuthor.text = "Автор:
            ${DataRepository.getUserById(review.authorId)?.login ?:
            "Неизвестен"}"

            val targetName = when(review.targetType) {
                ReviewTargetType.MASTER -> {
                    val master =
                        DataRepository.getUserById(review.targetId)
                    context.getString(R.string.review_on_master,
                        master?.fullName ?: "Неизвестный мастер")
                }
                ReviewTargetType.SERVICE -> {
                    val service =
                        DataRepository.getServiceById(review.targetId)
                    context.getString(R.string.review_on_service,
                        service?.name ?: "Неизвестная услуга")
                }
            }
        }
    }
}

```

```

        }
    }
    reviewTarget.text = targetName

    if (!review.reply.isNullOrBlank()) {
        replyLayout.visibility = View.VISIBLE
        replyText.text = review.reply
    } else {
        replyLayout.visibility = View.GONE
    }

    deleteButton.setOnClickListener {
listener.onDeleteClicked(review) }
    }

    override fun getItemCount(): Int = reviews.size

    fun updateData(newReviews: List<Review>) {
        this.reviews = newReviews
        notifyDataSetChanged()
    }
}

=====

com\example\marmar\adapter\AdminServiceAdapter.kt
-----
package com.example.marmar.adapter

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.marmar.data.model.Service
import com.example.marmar.databinding.ItemAdminServiceBinding

class AdminServiceAdapter(
    private var services: List<Service>,
    private val listener: ServiceInteractionListener
) : RecyclerView.Adapter<AdminServiceAdapter.ServiceViewHolder>()
{

    interface ServiceInteractionListener {
        fun onEditClicked(service: Service)
        fun onDeleteClicked(service: Service)
    }

    inner class ServiceViewHolder(val binding:
ItemAdminServiceBinding) : RecyclerView.ViewHolder(binding.root)

        override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): ServiceViewHolder {

```

```

        val binding =
            ItemAdminServiceBinding.inflate(LayoutInflater.from(parent.context
            ), parent, false)
        return ServiceViewHolder(binding)
    }

    override fun onBindViewHolder(holder: ServiceViewHolder,
    position: Int) {
        val service = services[position]
        with(holder.binding) {
            serviceName.text = service.name
            servicePrice.text = service.price

            editButton.setOnClickListener {
            listener.onEditClicked(service) }
            deleteButton.setOnClickListener {
            listener.onDeleteClicked(service) }
        }
    }

    override fun getItemCount() = services.size

    fun updateData(newServices: List<Service>) {
        this.services = newServices
        notifyDataSetChanged()
    }
}

```

=====

com\example\marmar\adapter\AppointmentAdapter.kt

package com.example.marmar.adapter

```

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.core.content.ContextCompat
import androidx.recyclerview.widget.RecyclerView
import com.example.marmar.R
import com.example.marmar.data.model.Appointment
import com.example.marmar.data.model.AppointmentStatus
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.ItemAppointmentBinding

```

```

class AppointmentAdapter(
    private var appointments: List<Appointment>,
    private val listener: AppointmentInteractionListener
) :
    RecyclerView.Adapter<AppointmentAdapter.AppointmentViewHolder>() {

    interface AppointmentInteractionListener {
        fun onPayClicked(appointment: Appointment)
    }
}

```



```

        buttonPay.setOnClickListener {
listener.onPayClicked(appointment) }

statusContainer.setCardBackgroundColor(ContextCompat.getColor(cont
ext, R.color.pink)) // Заменяли цвет
    }
    // --- КОНЕЦ ИЗМЕНЕНИЙ ---
    AppointmentStatus.CONFIRMED -> {
        appointmentStatus.text =
context.getString(R.string.status_confirmed)

statusContainer.setCardBackgroundColor(ContextCompat.getColor(cont
ext, R.color.success_green))
    }
    AppointmentStatus.COMPLETED -> {
        appointmentStatus.text =
context.getString(R.string.status_completed)
        val hasReview = DataRepository.reviews.any {
it.appointmentId == appointment.id }
        if (!hasReview) {
            buttonLeaveReview.visibility =
View.VISIBLE
            buttonLeaveReview.setOnClickListener {
listener.onLeaveReviewClicked(appointment) }
        }
    }
    AppointmentStatus.CANCELLED -> {
        appointmentStatus.text =
context.getString(R.string.status_cancelled)
        cancellationReasonText.visibility =
View.VISIBLE
        cancellationReasonText.text =
context.getString(R.string.cancellation_reason,
appointment.cancellationReason ?: "Причина не указана")

statusContainer.setCardBackgroundColor(ContextCompat.getColor(cont
ext, R.color.error_red))
    }
    }
}

override fun getItemCount() = appointments.size

fun updateData(newAppointments: List<Appointment>) {
    this.appointments = newAppointments
    notifyDataSetChanged()
}

}

=====

com\example\marmar\adapter\HomeTileAdapter.kt

```

```

-----
package com.example.marmar.adapter

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.marmar.databinding.ItemHomeTileBinding

data class HomeTile(val title: String, val iconResId: Int, val
action: () -> Unit)

class HomeTileAdapter(private val tiles: List<HomeTile>) :
RecyclerView.Adapter<HomeTileAdapter.TileViewHolder>() {

    inner class TileViewHolder(val binding: ItemHomeTileBinding) :
RecyclerView.ViewHolder(binding.root)

        override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): TileViewHolder {
            val binding =
ItemHomeTileBinding.inflate(LayoutInflater.from(parent.context),
parent, false)
            return TileViewHolder(binding)
        }

        override fun onBindViewHolder(holder: TileViewHolder,
position: Int) {
            val tile = tiles[position]
            holder.binding.tileIcon.setImageResource(tile.iconResId)
            holder.binding.tileTitle.text = tile.title
            holder.itemView.setOnClickListener { tile.action() }
        }

        override fun getItemCount() = tiles.size
    }
}

```

```

=====

com\example\marmar\adapter\MasterAdapter.kt
-----
package com.example.marmar.adapter

import android.net.Uri
import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.marmar.R
import com.example.marmar.data.model.User
import com.example.marmar.databinding.ItemMasterCardBinding
import java.io.File

class MasterAdapter(
    private var masters: List<User>,

```

```

        private val onItemClick: (User) -> Unit
    ) : RecyclerView.Adapter<MasterAdapter.MasterViewHolder>() {

        inner class MasterViewHolder(val binding:
ItemMasterCardBinding) : RecyclerView.ViewHolder(binding.root)

            override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): MasterViewHolder {
                val binding =
ItemMasterCardBinding.inflate(LayoutInflater.from(parent.context),
parent, false)
                return MasterViewHolder(binding)
            }

            override fun onBindViewHolder(holder: MasterViewHolder,
position: Int) {
                val master = masters[position]
                with(holder.binding) {
                    masterName.text = master.fullName
                    masterSpecialization.text = master.specialization
                    masterExperience.text =
holder.itemView.context.getString(R.string.master_detail_experienc
e, master.experienceYears ?: 0)

                    master.photoPath?.let { path ->
                        val file = File(path)
                        if (file.exists()) {
                            masterPhoto.setImageURI(Uri.fromFile(file))
                        } else {
masterPhoto.setImageResource(R.drawable.ic_default_avatar)
                        }
                    } ?:
masterPhoto.setImageResource(R.drawable.ic_default_avatar)

                    root.setOnClickListener {
                        onItemClick(master)
                    }
                }
            }

            override fun getItemCount() = masters.size

            fun updateData(newMasters: List<User>) {
                masters = newMasters
                notifyDataSetChanged()
            }
    }

```

```
=====
```

```
com\example\marmar\adapter\MasterRequestAdapter.kt
```

```
-----
```

```

package com.example.marmar.adapter

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.marmar.data.model.Appointment
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.ItemMasterRequestBinding

class MasterRequestAdapter(
    private var appointments: List<Appointment>,
    private val listener: RequestInteractionListener
) : RecyclerView.Adapter<MasterRequestAdapter.RequestViewHolder>()
{

    interface RequestInteractionListener {
        fun onConfirmClicked(appointment: Appointment)
        fun onCancelClicked(appointment: Appointment)
    }

    inner class RequestViewHolder(val binding:
ItemMasterRequestBinding) : RecyclerView.ViewHolder(binding.root)

        override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): RequestViewHolder {
            val binding =
ItemMasterRequestBinding.inflate(LayoutInflater.from(parent.context), parent, false)
            return RequestViewHolder(binding)
        }

        override fun onBindViewHolder(holder: RequestViewHolder,
position: Int) {
            val appointment = appointments[position]
            with(holder.binding) {
                val client =
DataRepository.getUserById(appointment.clientId)
                val service =
DataRepository.getServiceById(appointment.serviceId)

                clientName.text = client?.fullName ?: "Клиент не
найден"
                serviceName.text = service?.name ?: "Услуга не
найдена"
                appointmentDate.text = appointment.getFormattedDate()
                appointmentTime.text = appointment.getFormattedTime()

                confirmButton.setOnClickListener {
listener.onConfirmClicked(appointment) }
                cancelButton.setOnClickListener {
listener.onCancelClicked(appointment) }
            }
        }
    }
}

```

```

        override fun getItemCount() = appointments.size

        fun updateData(newAppointments: List<Appointment>) {
            this.appointments = newAppointments
            notifyDataSetChanged()
        }
    }
}

=====

com\example\marmar\adapter\MasterReviewAdapter.kt
-----
package com.example.marmar.adapter

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.marmar.data.model.Review
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.ItemMasterReviewBinding

class MasterReviewAdapter(
    private var reviews: List<Review>,
    private val onReplyClicked: (Review) -> Unit
) : RecyclerView.Adapter<MasterReviewAdapter.ReviewViewHolder>() {

    inner class ReviewViewHolder(val binding:
    ItemMasterReviewBinding) : RecyclerView.ViewHolder(binding.root)

        override fun onCreateViewHolder(parent: ViewGroup, viewType:
    Int): ReviewViewHolder {
            val binding =
    ItemMasterReviewBinding.inflate(LayoutInflater.from(parent.context
    ), parent, false)
            return ReviewViewHolder(binding)
        }

        override fun onBindViewHolder(holder: ReviewViewHolder,
    position: Int) {
            val review = reviews[position]
            with(holder.binding) {
                val author =
    DataRepository.getUserById(review.authorId)
                reviewAuthor.text = author?.fullName ?: "Анонимный
клиент"

                reviewRatingBar.rating = review.rating.toFloat()
                reviewText.text = review.text

                if (review.reply.isNullOrBlank()) {
                    replyLayout.visibility = View.GONE
                    replyButton.visibility = View.VISIBLE
                }
            }
        }
    }
}

```

```

        replyButton.setOnClickListener {
onReplyClicked(review) }
        } else {
            replyLayout.visibility = View.VISIBLE
            replyButton.visibility = View.GONE
            replyText.text = review.reply
        }
    }
}

override fun getItemCount() = reviews.size

fun updateData(newReviews: List<Review>) {
    this.reviews = newReviews
    notifyDataSetChanged()
}
}

=====

com\example\marmar\adapter\ReviewAdapter.kt
-----
package com.example.marmar.adapter

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.marmar.R
import com.example.marmar.data.model.Review
import com.example.marmar.data.model.ReviewTargetType
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.ItemReviewBinding

class ReviewAdapter(
    private val reviews: List<Review>
) : RecyclerView.Adapter<ReviewAdapter.ReviewViewHolder>() {

    inner class ReviewViewHolder(val binding: ItemReviewBinding) :
        RecyclerView.ViewHolder(binding.root)

    override fun onCreateViewHolder(parent: ViewGroup, viewType:
        Int): ReviewViewHolder {
        val binding =
            ItemReviewBinding.inflate(LayoutInflater.from(parent.context),
            parent, false)
        return ReviewViewHolder(binding)
    }

    override fun onBindViewHolder(holder: ReviewViewHolder,
        position: Int) {
        val review = reviews[position]
        val context = holder.itemView.context

```

```

        with(holder.binding) {
            reviewRatingBar.rating = review.rating.toFloat()
            reviewText.text = review.text

            // Определяем, на что оставлен отзыв
            val targetName = when(review.targetType) {
                ReviewTargetType.MASTER -> {
                    val master =
                        DataRepository.getUserById(review.targetId)
                        context.getString(R.string.review_on_master,
                            master?.fullName ?: "Неизвестный мастер")
                }
                ReviewTargetType.SERVICE -> {
                    val service =
                        DataRepository.getServiceById(review.targetId)
                        context.getString(R.string.review_on_service,
                            service?.name ?: "Неизвестная услуга")
                }
            }
            reviewTarget.text = targetName

            // Показываем ответ, если он есть
            if (!review.reply.isNullOrBlank()) {
                replyLayout.visibility = View.VISIBLE
                replyText.text = review.reply
            } else {
                replyLayout.visibility = View.GONE
            }
        }
    }

    override fun getItemCount() = reviews.size
}

=====

com\example\marmar\adapter\ServiceAdapter.kt
-----
package com.example.marmar.adapter

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.marmar.data.model.Service
import com.example.marmar.databinding.ItemServiceBinding

class ServiceAdapter(
    private var services: List<Service>,
    private val onBookClick: (Service) -> Unit
) : RecyclerView.Adapter<ServiceAdapter.ServiceViewHolder>() {

    inner class ServiceViewHolder(val binding: ItemServiceBinding)
    : RecyclerView.ViewHolder(binding.root)

```

```

        override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): ServiceViewHolder {
            val binding =
ItemServiceBinding.inflate(LayoutInflater.from(parent.context),
parent, false)
            return ServiceViewHolder(binding)
        }

        override fun onBindViewHolder(holder: ServiceViewHolder,
position: Int) {
            val service = services[position]
            with(holder.binding) {
                serviceName.text = service.name
                servicePrice.text = service.price
                serviceDescription.text = service.description
                bookButton.setOnClickListener {
                    onBookClick(service)
                }
            }
        }

        override fun getItemCount() = services.size
    }

```

=====

com\example\marmar\adapter\TimeSlotAdapter.kt

package com.example.marmar.adapter

```

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.marmar.databinding.ItemTimeSlotBinding

```

```

class TimeSlotAdapter(
    private var timeSlots: List<String>,
    private val onTimeSelected: (String) -> Unit
) : RecyclerView.Adapter<TimeSlotAdapter.TimeSlotViewHolder>() {

    private var selectedPosition = RecyclerView.NO_POSITION

    inner class TimeSlotViewHolder(val binding:
ItemTimeSlotBinding) : RecyclerView.ViewHolder(binding.root)

        override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): TimeSlotViewHolder {
            val binding =
ItemTimeSlotBinding.inflate(LayoutInflater.from(parent.context),
parent, false)
            return TimeSlotViewHolder(binding)
        }
    }

```

```

        override fun onBindViewHolder(holder: TimeSlotViewHolder,
position: Int) {
            val time = timeSlots[position]
            holder.binding.timeChip.text = time
            holder.binding.timeChip.isChecked = position ==
selectedPosition

            holder.binding.timeChip.setOnClickListener {
                notifyItemChanged(selectedPosition) // Снимаем
выделение со старого
                selectedPosition = holder.adapterPosition
                notifyItemChanged(selectedPosition) // Выделяем новый
                onTimeSelected(time)
            }
        }

        override fun getItemCount() = timeSlots.size

        fun updateData(newTimeSlots: List<String>) {
            timeSlots = newTimeSlots
            selectedPosition = RecyclerView.NO_POSITION
            notifyDataSetChanged()
        }
    }
}

```

```

=====

```

```

com\example\marmar\data\model\Appointment.kt

```

```

-----

```

```

package com.example.marmar.data.model

```

```

import java.io.Serializable

```

```

import java.util.Date

```

```

enum class AppointmentStatus {
    PENDING_CONFIRMATION, // Ожидает подтверждения
    AWAITING_PAYMENT,     // Ожидает оплаты
    CONFIRMED,             // Подтверждено и оплачено
    COMPLETED,            // Завершено
    CANCELLED              // Отменено
}

```

```

data class Appointment(
    val id: String,
    var clientId: String,
    var masterId: String,
    var serviceId: String,
    var dateTime: Long, // Храним в виде timestamp (ms) для
удобной сортировки и работы
    var status: AppointmentStatus,
    var cancellationReason: String? = null
) : Serializable {

```

```

        fun getFormattedDate(): String {
            val sdf = java.text.SimpleDateFormat("dd.MM.yyyy",
java.util.Locale.getDefault())
            return sdf.format(Date(dateTime))
        }
        fun getFormattedTime(): String {
            val sdf = java.text.SimpleDateFormat("HH:mm",
java.util.Locale.getDefault())
            return sdf.format(Date(dateTime))
        }
    }
}

```

=====

com\example\marmar\data\model\Review.kt

package com.example.marmar.data.model

import java.io.Serializable

```

enum class ReviewTargetType {
    MASTER, SERVICE
}

```

```

data class Review(
    val id: String,
    val authorId: String,
    val targetId: String, // ID мастера или услуги
    val targetType: ReviewTargetType,
    val rating: Int, // от 1 до 5
    val text: String,
    var reply: String? = null,
    val appointmentId: String // Ссылка на запись, по которой
оставлен отзыв
) : Serializable

```

=====

com\example\marmar\data\model\Service.kt

package com.example.marmar.data.model

import java.io.Serializable

```

data class Service(
    val id: String,
    var name: String,
    var price: String, // Тип String, так как может быть
диапазоном "4500 - 7000" или содержать "руб."
    var description: String
) : Serializable

```

=====

```

com\example\marmar\data\model\User.kt
-----
package com.example.marmar.data.model

import java.io.Serializable

enum class UserRole {
    CLIENT, MASTER, ADMIN
}

data class User(
    val id: String,
    var login: String,
    var password: String, // В реальном проекте здесь должен быть
хэш
    var role: UserRole,
    var fullName: String? = null,
    var phone: String? = null,
    var age: Int? = null,
    var email: String? = null,
    var photoPath: String? = null, // Путь к фото в хранилище
устройства

    // Данные специфичные для мастера
    var specialization: String? = null,
    var experienceYears: Int? = null,
    var providedServiceIds: MutableList<String>? = null,
    var workDays: MutableMap<String, Boolean>? = null, // e.g.
{"ПН": true, "БТ": true ...}
    var workHoursStart: Int? = null, // e.g. 9
    var workHoursEnd: Int? = null, // e.g. 18
) : Serializable // Serializable для передачи объектов между
фрагментами

=====

com\example\marmar\data\repository\DataRepository.kt
-----
package com.example.marmar.data.repository

import android.content.Context
import android.util.Log
import com.example.marmar.data.model.*
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.withContext
import java.io.File
import java.io.IOException

// Синглтон для хранения данных в памяти и работы с файлами
object DataRepository {

    private const val USERS_FILE = "marmar_users.txt"

```

```

        private const val SERVICES_FILE = "marmar_services.txt"
        private const val APPOINTMENTS_FILE =
"marmar_appointments.txt"
        private const val REVIEWS_FILE = "marmar_reviews.txt"
        private const val TAG = "DataRepository"

        // Кэш данных в оперативной памяти
        val users = mutableListOf<User>()
        val services = mutableListOf<Service>()
        val appointments = mutableListOf<Appointment>()
        val reviews = mutableListOf<Review>()

        var currentUser: User? = null

        // Функция для первоначальной загрузки данных из файлов
        suspend fun loadData(context: Context) {
            withContext(Dispatchers.IO) {
                // Проверяем, существуют ли файлы данных. Если нет -
копируем из assets.
                checkAndCopyInitialData(context)

                // Очищаем кэш перед загрузкой, чтобы избежать
дублирования
                users.clear()
                services.clear()
                appointments.clear()
                reviews.clear()

                Log.d(TAG, "Начало загрузки данных из файлов...")

                loadUsers(context)
                loadServices(context)
                loadAppointments(context)
                loadReviews(context)

                Log.d(TAG, "Загрузка данных завершена. Пользователей:
${users.size}, Услуг: ${services.size}, Записей:
${appointments.size}, Отзывов: ${reviews.size}")
            }
        }

        private fun checkAndCopyInitialData(context: Context) {
            val filesDir = context.filesDir
            val filesToCopy = listOf(USERS_FILE, SERVICES_FILE,
APPOINTMENTS_FILE, REVIEWS_FILE)

            for (filename in filesToCopy) {
                val destinationFile = File(filesDir, filename)
                // Копируем, только если файла не существует (первый
запуск)
                if (!destinationFile.exists()) {
                    try {

```



```

        age = parts[6].toIntOrNull(),
        email = parts[7].ifEmpty { null },
        photoPath = parts[8].ifEmpty { null },
        specialization = parts[9].ifEmpty { null
    },
        experienceYears = parts[10].toIntOrNull(),
        providedServiceIds =
parts[11].split(",").filter { it.isNotBlank() }.toMutableList(),
        workDays = parts[12].ifEmpty { null }?.let
{ parseWorkDays(it) },
        workHoursStart = parts[13].toIntOrNull(),
        workHoursEnd = parts[14].toIntOrNull()
    )
    users.add(user)
}
}
} catch (e: Exception) {
    Log.e(TAG, "Ошибка при чтении файла пользователей:
${e.message}")
}
}

private fun loadServices(context: Context) {
    val file = File(context.filesDir, SERVICES_FILE)
    if (!file.exists()) {
        Log.w(TAG, "Файл услуг '$SERVICES_FILE' не найден.")
        return
    }
    try {
        file.forEachLine { line ->
            if (line.isBlank()) return@forEachLine
            val parts = line.split(";")
            if (parts.size == 4) {
                services.add(Service(id = parts[0], name =
parts[1], price = parts[2], description = parts[3]))
            }
        }
    } catch (e: Exception) {
        Log.e(TAG, "Ошибка при чтении файла услуг:
${e.message}")
    }
}

private fun loadAppointments(context: Context) {
    val file = File(context.filesDir, APPOINTMENTS_FILE)
    if (!file.exists()) {
        Log.w(TAG, "Файл записей '$APPOINTMENTS_FILE' не
найден.")
        return
    }
    try {
        file.forEachLine { line ->
            if (line.isBlank()) return@forEachLine

```

```

        val parts = line.split(";")
        if (parts.size >= 6) {
            appointments.add(Appointment(
                id = parts[0],
                clientId = parts[1],
                masterId = parts[2],
                serviceId = parts[3],
                dateTime = parts[4].toLong(),
                status =
AppointmentStatus.valueOf(parts[5]),
                cancellationReason =
parts.getOrNull(6)?.ifEmpty { null }
            ))
        }
    }
} catch (e: Exception) {
    Log.e(TAG, "Ошибка при чтении файла записей:
${e.message}")
}

private fun loadReviews(context: Context) {
    val file = File(context.filesDir, REVIEWS_FILE)
    if (!file.exists()) {
        Log.w(TAG, "Файл отзывов '$REVIEWS_FILE' не найден.")
        return
    }
    try {
        file.forEachLine { line ->
            if (line.isBlank()) return@forEachLine
            val parts = line.split(";")
            if (parts.size >= 7) {
                reviews.add(Review(
                    id = parts[0],
                    authorId = parts[1],
                    targetId = parts[2],
                    targetType =
ReviewTargetType.valueOf(parts[3]),
                    rating = parts[4].toInt(),
                    text = parts[5],
                    reply = parts.getOrNull(6)?.ifEmpty { null
},
                    appointmentId = parts[7]
                ))
            }
        }
    } catch (e: Exception) {
        Log.e(TAG, "Ошибка при чтении файла отзывов:
${e.message}")
    }
}

```

```

private fun saveUsers(context: Context) {
    val file = File(context.filesDir, USERS_FILE)
    try {
        file.printWriter().use { out ->
            users.forEach { user ->
                val line = with(user) {
                    "$id;$login;$password;$role;${fullName ? :
""};${phone ? : ""};${age ? : ""};${email ? : ""};${photoPath ? :
""};${specialization ? : ""};${experienceYears ? :
""};${providedServiceIds?.joinToString(",") ? :
""};${formatWorkDays(workDays)};${workHoursStart ? :
""};${workHoursEnd ? : ""}"
                }
                out.println(line)
            }
        }
    } catch (e: IOException) {
        Log.e(TAG, "Ошибка при записи файла пользователей:
${e.message}")
    }
}

private fun saveServices(context: Context) {
    val file = File(context.filesDir, SERVICES_FILE)
    try {
        file.printWriter().use { out ->
            services.forEach { service ->

out.println("${service.id};${service.name};${service.price};${serv
ice.description}")
            }
        }
    } catch (e: IOException) {
        Log.e(TAG, "Ошибка при записи файла услуг:
${e.message}")
    }
}

private fun saveAppointments(context: Context) {
    val file = File(context.filesDir, APPOINTMENTS_FILE)
    try {
        file.printWriter().use { out ->
            appointments.forEach { appointment ->

out.println("${appointment.id};${appointment.clientId};${appointme
nt.masterId};${appointment.serviceId};${appointment.dateTime};${ap
pointment.status};${appointment.cancellationReason ? : ""}")
            }
        }
    } catch (e: IOException) {
        Log.e(TAG, "Ошибка при записи файла записей:
${e.message}")
    }
}

```

```

    }

    private fun saveReviews(context: Context) {
        val file = File(context.filesDir, REVIEWS_FILE)
        try {
            file.printWriter().use { out ->
                reviews.forEach { review ->

out.println("${review.id};${review.authorId};${review.targetId};${
review.targetType};${review.rating};${review.text};${review.reply
?: ""};${review.appointmentId}")
                }
            }
        } catch (e: IOException) {
            Log.e(TAG, "Ошибка при записи файла отзывов:
${e.message}")
        }
    }

    // Вспомогательные функции для парсинга/форматирования
    private fun parseWorkDays(data: String): MutableMap<String,
Boolean> {
        val map = mutableMapOf<String, Boolean>()
        data.split(",").forEach {
            val pair = it.split(":")
            if (pair.size == 2) {
                map[pair[0]] = pair[1].toBoolean()
            }
        }
        return map
    }

    private fun formatWorkDays(map: Map<String, Boolean>?): String
{
        return map?.map { "${it.key}:${it.value}"
}?.joinToString(",") ?: ""
    }

    // --- Функции для доступа к данным ---

    fun findUserByLogin(login: String): User? {
        return users.firstOrNull { it.login.equals(login,
ignoreCase = true) }
    }

    fun getUserById(id: String): User? {
        return users.firstOrNull { it.id == id }
    }

    fun getServiceById(id: String): Service? {
        return services.firstOrNull { it.id == id }
    }

```

```

    fun getAppointmentById(id: String): Appointment? {
        return appointments.firstOrNull { it.id == id }
    }

    fun getMasters(): List<User> {
        return users.filter { it.role == UserRole.MASTER }
    }

    fun getAppointmentsForClient(clientId: String):
List<Appointment> {
        return appointments.filter { it.clientId == clientId
    }.sortedByDescending { it.dateTime }
    }

    fun getReviewsByAuthor(authorId: String): List<Review> {
        return reviews.filter { it.authorId == authorId }
    }
}

=====

com\example\marmar\fragment\admin\AdminAccountsFragment.kt
-----
package com.example.marmar.fragment.admin

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ArrayAdapter
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.fragment.app.Fragment
import androidx.lifecycle.LifecycleScope
import androidx.navigation.fragment.findNavController
import com.example.marmar.R
import com.example.marmar.adapter.AdminAccountAdapter
import com.example.marmar.data.model.User
import com.example.marmar.data.model.UserRole
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.DialogAddAccountBinding
import com.example.marmar.databinding.DialogConfirmPasswordBinding
import com.example.marmar.databinding.FragmentAdminAccountsBinding
import kotlinx.coroutines.launch
import java.util.UUID

class AdminAccountsFragment : Fragment(),
AdminAccountAdapter.AccountInteractionListener {

    private var _binding: FragmentAdminAccountsBinding? = null
    private val binding get() = _binding!!
    private lateinit var adapter: AdminAccountAdapter

```

```

        override fun onCreateView(inflater: LayoutInflater, container:
ViewGroup?, savedInstanceState: Bundle?): View {
            _binding = FragmentAdminAccountsBinding.inflate(inflater,
container, false)
            return binding.root
        }

        override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
            super.onViewCreated(view, savedInstanceState)
            setupRecyclerView()
            binding.fabAddAccount.setOnClickListener {
                showAddAccountDialog()
            }
        }

        private fun setupRecyclerView() {
            adapter = AdminAccountAdapter(DataRepository.users, this)
            binding.accountsRecyclerView.adapter = adapter
        }

        override fun onEditClicked(user: User) {
            // Передаем ID пользователя для редактирования
            val action =
AdminAccountsFragmentDirections.actionAdminAccountsFragmentToProfi
leEditFragment(user.id)
            findNavController().navigate(action)
        }

        override fun onChangeRoleClicked(user: User) {
            val roles = UserRole.values().map { it.name
            }.toTypedArray()
            val currentRoleIndex =
UserRole.values().indexOf(user.role)

            AlertDialog.Builder(requireContext())
                .setTitle(getString(R.string.dialog_change_role_title,
user.login))
                .setSingleChoiceItems(roles, currentRoleIndex) {
dialog, which ->
                    val selectedRole = UserRole.values()[which]

                    if (selectedRole == UserRole.ADMIN && user.role !=
UserRole.ADMIN) {
                        showPasswordConfirmationDialog {
                            user.role = selectedRole
                            saveAndReload()
                            dialog.dismiss()
                        }
                    } else {
                        user.role = selectedRole
                        saveAndReload()
                        dialog.dismiss()
                    }
                }
        }

```

```

        }
    }
    .setNegativeButton(R.string.dialog_button_cancel,
null)
    .show()
}

override fun onDeleteClicked(user: User) {
    if (user.id == DataRepository.currentUser?.id) {
        Toast.makeText(context, "Вы не можете удалить свой
собственный аккаунт", Toast.LENGTH_SHORT).show()
        return
    }

    AlertDialog.Builder(requireContext())
        .setTitle(R.string.dialog_delete_confirmation_title)

    .setMessage(getString(R.string.dialog_delete_user_prompt,
user.login))
        .setIcon(R.drawable.ic_delete)
        .setPositiveButton(R.string.button_delete) { _, _ ->
            DataRepository.users.remove(user)
            saveAndReload()
        }
        .setNegativeButton(R.string.dialog_button_cancel,
null)
        .show()
    }

    private fun showAddAccountDialog() {
        val dialogBinding =
DialogAddAccountBinding.inflate(layoutInflater)
        val roles = UserRole.values().map { it.name }
        val spinnerAdapter = ArrayAdapter(requireContext(),
android.R.layout.simple_spinner_dropdown_item, roles)
        dialogBinding.roleSpinner.adapter = spinnerAdapter

        AlertDialog.Builder(requireContext())
            .setView(dialogBinding.root)
            .setPositiveButton("Добавить") { _, _ ->
                val login =
dialogBinding.loginEditText.text.toString().trim()
                val password =
dialogBinding.passwordEditText.text.toString().trim()
                if (login.isEmpty() || password.isEmpty()) {
                    Toast.makeText(context, "Логин и пароль не
могут быть пустыми", Toast.LENGTH_SHORT).show()
                    return@setPositiveButton
                }
                if (DataRepository.findUserByLogin(login) != null)
{
                    Toast.makeText(context, "Этот логин уже
занят", Toast.LENGTH_SHORT).show()

```

```

        return@setPositiveButton
    }
    val selectedRole =
UserRole.valueOf(dialogBinding.roleSpinner.selectedItem.toString()
)

        val newUser = User(
            id = UUID.randomUUID().toString(),
            login = login,
            password = password,
            role = selectedRole
        )
        DataRepository.users.add(newUser)
        saveAndReload()
    }
    .setNegativeButton(R.string.dialog_button_cancel,
null)
        .show()
    }

    private fun showPasswordConfirmationDialog(onSuccess: () ->
Unit) {
        val dialogBinding =
DialogConfirmPasswordBinding.inflate(layoutInflater)
        AlertDialog.Builder(requireContext())
            .setView(dialogBinding.root)
            .setPositiveButton("Подтвердить") { _, _ ->
                val password =
dialogBinding.passwordEditText.text.toString()
                if (password ==
DataRepository.currentUser?.password) {
                    onSuccess()
                } else {
                    Toast.makeText(context,
R.string.error_login_password_incorrect,
Toast.LENGTH_SHORT).show()
                }
            }
            .setNegativeButton(R.string.dialog_button_cancel,
null)
                .show()
        }

    private fun saveAndReload() {
        lifecycleScope.launch {
            DataRepository.saveData(requireContext())
            adapter.updateData(DataRepository.users.toList()) //
Пересоздаем список, чтобы адаптер обновился
        }
    }

    override fun onDestroyView() {
        super.onDestroyView()
    }

```

```

        _binding = null
    }
}

=====

com\example\marmar\fragment\admin\AdminAppointmentsFragment.kt
-----
package com.example.marmar.fragment.admin

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.AdapterView
import android.widget.AdapterView.Adapter
import android.widget.AdapterView.OnItemClickListener
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.fragment.app.Fragment
import androidx.lifecycle.Lifecycle
import androidx.lifecycle.LifecycleScope
import com.example.marmar.R
import com.example.marmar.adapter.AdminAppointmentAdapter
import com.example.marmar.data.model.Appointment
import com.example.marmar.data.model.AppointmentStatus
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.DialogEditAppointmentBinding
import
com.example.marmar.databinding.FragmentAdminAppointmentsBinding
import kotlinx.coroutines.launch

class AdminAppointmentsFragment : Fragment(),
AdminAppointmentAdapter.AppointmentInteractionListener {

    private var _binding: FragmentAdminAppointmentsBinding? = null
    private val binding get() = _binding!!

    private lateinit var adapter: AdminAppointmentAdapter

    override fun onCreateView(inflater: LayoutInflater, container:
ViewGroup?, savedInstanceState: Bundle?): View {
        _binding =
FragmentAdminAppointmentsBinding.inflate(inflater, container,
false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        setupRecyclerView()
    }

    private fun setupRecyclerView() {

```

```

        val sortedAppointments =
DataRepository.appointments.sortedByDescending { it.dateTime }
        adapter = AdminAppointmentAdapter(sortedAppointments,
this)
        binding.adminAppointmentsRecyclerView.adapter = adapter
    }

    // Эта функция теперь будет вызывать диалог редактирования
    override fun onEditClicked(appointment: Appointment) {
        showEditAppointmentDialog(appointment)
    }

    override fun onDeleteClicked(appointment: Appointment) {
        AlertDialog.Builder(requireContext())
            .setTitle(R.string.dialog_delete_confirmation_title)
            .setMessage(R.string.dialog_delete_appointment_prompt)
            .setIcon(R.drawable.ic_delete)
            .setPositiveButton(R.string.button_delete) { _, _ ->
                DataRepository.appointments.remove(appointment)
                saveAndReload()
            }
            .setNegativeButton(R.string.dialog_button_cancel,
null)
            .show()
    }

    private fun showEditAppointmentDialog(appointment:
Appointment) {
        val dialogBinding =
DialogEditAppointmentBinding.inflate(layoutInflater)
        val context = requireContext()

        // Заполняем информацию о записи
        val client =
DataRepository.getUserById(appointment.clientId)
        val master =
DataRepository.getUserById(appointment.masterId)
        val infoText = "Запись #${appointment.id.take(6)}\nКлиент:
${client?.fullName}\nМастер: ${master?.fullName}"
        dialogBinding.appointmentInfoText.text = infoText

        // Настраиваем Spinner со статусами
        val statuses = AppointmentStatus.values()
        // Создаем массив с русскими названиями статусов
        val statusNames = statuses.map { getRussianStatusName(it)
}

        val spinnerAdapter = ArrayAdapter(context,
android.R.layout.simple_spinner_dropdown_item, statusNames)
        dialogBinding.statusSpinner.adapter = spinnerAdapter

        // Устанавливаем текущий статус записи в Spinner
        val currentStatusIndex =
statuses.indexOf(appointment.status)

```

```

dialogBinding.statusSpinner.setSelection(currentStatusIndex)

    // Показываем/скрываем поле причины отмены в зависимости
    от выбора
    dialogBinding.statusSpinner.onItemSelectedListener =
    object : AdapterView.OnItemSelectedListener {
        override fun onItemSelected(parent: AdapterView<*>?,
        view: View?, position: Int, id: Long) {
            if (statuses[position] ==
            AppointmentStatus.CANCELLED) {

dialogBinding.cancellationReasonLayout.visibility = View.VISIBLE
                } else {

dialogBinding.cancellationReasonLayout.visibility = View.GONE
                }
            }
        override fun onNothingSelected(parent:
        AdapterView<*>?) {}
    }

    AlertDialog.Builder(context)
        .setView(dialogBinding.root)
        .setPositiveButton("Сохранить") { _, _ ->
            val selectedStatus =
            statuses[dialogBinding.statusSpinner.selectedItemPosition]

            if (selectedStatus == AppointmentStatus.CANCELLED)
        {
            val reason =
            dialogBinding.cancellationReasonEditText.text.toString().trim()
            if (reason.isEmpty()) {
                Toast.makeText(context,
                R.string.error_reason_required, Toast.LENGTH_SHORT).show()
                return@setPositiveButton // Не закрываем
                диалог, если причина не указана
            }
            appointment.cancellationReason = reason
        }

            appointment.status = selectedStatus
            Toast.makeText(context, "Статус записи обновлен",
            Toast.LENGTH_SHORT).show()
            saveAndReload()
        }
        .setNegativeButton(R.string.dialog_button_cancel,
        null)
        .show()
    }

    // Вспомогательная функция для перевода Enum в русскую строку

```

```

        private fun getRussianStatusName(status: AppointmentStatus):
String {
            return when (status) {
                AppointmentStatus.PENDING_CONFIRMATION ->
getString(R.string.status_pending)
                AppointmentStatus.AWAITING_PAYMENT ->
getString(R.string.status_awaiting_payment)
                AppointmentStatus.CONFIRMED ->
getString(R.string.status_confirmed)
                AppointmentStatus.COMPLETED ->
getString(R.string.status_completed)
                AppointmentStatus.CANCELLED ->
getString(R.string.status_cancelled)
            }
        }

        private fun saveAndReload() {
            lifecycleScope.launch {
                DataRepository.saveData(requireContext())
            }

            adapter.updateData(DataRepository.appointments.sortedByDescending
{ it.dateTime })
        }

        override fun onDestroyView() {
            super.onDestroyView()
            _binding = null
        }
    }
}

```

=====

com\example\marmar\fragment\admin\AdminHomeFragment.kt

package com.example.marmar.fragment.admin

```

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.findNavController
import com.example.marmar.R
import com.example.marmar.adapter.HomeTile
import com.example.marmar.adapter.HomeTileAdapter
import com.example.marmar.databinding.FragmentAdminHomeBinding

```

```

class AdminHomeFragment : Fragment() {

```

```

    private var _binding: FragmentAdminHomeBinding? = null
    private val binding get() = _binding!!

```

```

        override fun onCreateView(
            inflater: LayoutInflater, container: ViewGroup?,
            savedInstanceState: Bundle?
        ): View {
            _binding = FragmentAdminHomeBinding.inflate(inflater,
container, false)
            return binding.root
        }

        override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
            super.onViewCreated(view, savedInstanceState)
            setupNavigationTiles()
        }

        private fun setupNavigationTiles() {
            val tiles = listOf(
                HomeTile("Статистика", R.drawable.ic_stats) {
findNavController().navigate(R.id.nav_admin_stats) },
                HomeTile("Аккаунты", R.drawable.ic_accounts) {
findNavController().navigate(R.id.nav_admin_accounts) },
                HomeTile("Мастера", R.drawable.ic_masters) {
findNavController().navigate(R.id.nav_admin_masters) },
                HomeTile("Услуги", R.drawable.ic_services) {
findNavController().navigate(R.id.nav_admin_services) },
                HomeTile("Отзывы", R.drawable.ic_reviews) {
findNavController().navigate(R.id.nav_admin_reviews) },
                HomeTile("Записи", R.drawable.ic_appointments) {
findNavController().navigate(R.id.nav_admin_appointments) },
            )
            binding.adminHomeTilesRecycler.adapter =
HomeTileAdapter(tiles)
        }

        override fun onDestroyView() {
            super.onDestroyView()
            _binding = null
        }
    }
}

```

=====

com\example\marmar\fragment\admin\AdminMastersFragment.kt

package com.example.marmar.fragment.admin

```

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.findNavController
import com.example.marmar.adapter.AdminMasterAdapter

```

```

import com.example.marmar.data.model.User
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.FragmentAdminMastersBinding

class AdminMastersFragment : Fragment(),
AdminMasterAdapter.MasterInteractionListener {

    private var _binding: FragmentAdminMastersBinding? = null
    private val binding get() = _binding!!
    private lateinit var adapter: AdminMasterAdapter

    override fun onCreateView(inflater: LayoutInflater, container:
ViewGroup?, savedInstanceState: Bundle?): View {
        _binding = FragmentAdminMastersBinding.inflate(inflater,
container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        setupRecyclerView()
    }

    private fun setupRecyclerView() {
        adapter = AdminMasterAdapter(DataRepository.getMasters(),
this)
        binding.adminMastersRecyclerView.adapter = adapter
    }

    override fun onEditClicked(master: User) {
        // Передаем ID мастера для редактирования его рабочего
кабинета
        val action =
AdminMastersFragmentDirections.actionAdminMastersFragmentToNavMast
erCabinet(master.id)
        findNavController().navigate(action)
    }

    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}

=====

com\example\marmar\fragment\admin\AdminReviewsFragment.kt
-----
package com.example.marmar.fragment.admin

import android.os.Bundle
import android.view.LayoutInflater

```

```

import android.view.View
import android.view.ViewGroup
import androidx.appcompat.app.AlertDialog
import androidx.fragment.app.Fragment
import androidx.lifecycle.LifecycleScope
import com.example.marmar.R
import com.example.marmar.adapter.AdminReviewAdapter
import com.example.marmar.data.model.Review
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.FragmentAdminReviewsBinding
import kotlinx.coroutines.launch

class AdminReviewsFragment : Fragment(),
AdminReviewAdapter.ReviewInteractionListener {

    private var _binding: FragmentAdminReviewsBinding? = null
    private val binding get() = _binding!!
    private lateinit var adapter: AdminReviewAdapter

    override fun onCreateView(inflater: LayoutInflater, container:
ViewGroup?, savedInstanceState: Bundle?): View {
        _binding = FragmentAdminReviewsBinding.inflate(inflater,
container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        setupRecyclerView()
    }

    private fun setupRecyclerView() {
        adapter = AdminReviewAdapter(DataRepository.reviews, this)
        binding.adminReviewsRecyclerView.adapter = adapter
    }

    override fun onDeleteClicked(review: Review) {
        AlertDialog.Builder(requireContext())
            .setTitle(R.string.dialog_delete_confirmation_title)
            .setMessage(R.string.dialog_delete_review_prompt)
            .setIcon(R.drawable.ic_delete)
            .setPositiveButton(R.string.button_delete) { _, _ ->
                DataRepository.reviews.remove(review)
                saveAndReload()
            }
            .setNegativeButton(R.string.dialog_button_cancel,
null)
            .show()
    }

    private fun saveAndReload() {
        lifecycleScope.launch {

```

```

        DataRepository.saveData(requireContext())
        adapter.updateData(DataRepository.reviews)
    }
}

override fun onDestroyView() {
    super.onDestroyView()
    _binding = null
}
}

=====

com\example\marmar\fragment\admin\AdminServicesFragment.kt
-----
package com.example.marmar.fragment.admin

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.fragment.app.Fragment
import androidx.lifecycle.LifecycleScope
import com.example.marmar.R
import com.example.marmar.adapter.AdminServiceAdapter
import com.example.marmar.data.model.Service
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.DialogAddEditServiceBinding
import com.example.marmar.databinding.FragmentAdminServicesBinding
import kotlinx.coroutines.launch
import java.util.UUID

class AdminServicesFragment : Fragment(),
AdminServiceAdapter.ServiceInteractionListener {

    private var _binding: FragmentAdminServicesBinding? = null
    private val binding get() = _binding!!
    private lateinit var adapter: AdminServiceAdapter

    override fun onCreateView(inflater: LayoutInflater, container:
ViewGroup?, savedInstanceState: Bundle?): View {
        _binding = FragmentAdminServicesBinding.inflate(inflater,
container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        setupRecyclerView()
        binding.fabAddService.setOnClickListener {

```

```

        showServiceDialog(null)
    }
}

private fun setupRecyclerView() {
    adapter = AdminServiceAdapter(DataRepository.services,
this)
    binding.servicesRecyclerView.adapter = adapter
}

override fun onEditClicked(service: Service) {
    showServiceDialog(service)
}

override fun onDeleteClicked(service: Service) {
    AlertDialog.Builder(requireContext())
        .setTitle(R.string.dialog_delete_confirmation_title)
        .setMessage(getString(R.string.dialog_delete_service_prompt,
service.name))
        .setIcon(R.drawable.ic_delete)
        .setPositiveButton(R.string.button_delete) { _, _ ->
            DataRepository.services.remove(service)
            saveAndReload()
        }
        .setNegativeButton(R.string.dialog_button_cancel,
null)
        .show()
    }

    private fun showServiceDialog(service: Service?) {
        val isEditing = service != null
        val dialogBinding =
DialogAddEditServiceBinding.inflate(layoutInflater)

        dialogBinding.dialogTitle.text = if (isEditing)
getString(R.string.dialog_service_edit_title) else
getString(R.string.dialog_service_add_title)
        if (isEditing) {

dialogBinding.serviceNameEditText.setText(service!!.name)

dialogBinding.servicePriceEditText.setText(service.price)

dialogBinding.serviceDescriptionEditText.setText(service.descripti
on)

        }

        AlertDialog.Builder(requireContext())
            .setView(dialogBinding.root)
            .setPositiveButton("Сохранить") { _, _ ->
                val name =
dialogBinding.serviceNameEditText.text.toString().trim()

```

```

        val price =
dialogBinding.servicePriceEditText.text.toString().trim()
        val description =
dialogBinding.serviceDescriptionEditText.text.toString().trim()

        if (name.isEmpty() || price.isEmpty() ||
description.isEmpty()) {
            Toast.makeText(context, "Все поля должны быть
заполнены", Toast.LENGTH_SHORT).show()
            return@setPositiveButton
        }

        if (isEditing) {
            service!!.name = name
            service.price = price
            service.description = description
        } else {
            val newService =
Service(UUID.randomUUID().toString(), name, price, description)
            DataRepository.services.add(newService)
        }
        saveAndReload()
    }
    .setNegativeButton(R.string.dialog_button_cancel,
null)
    .show()
}

private fun saveAndReload() {
    lifecycleScope.launch {
        DataRepository.saveData(requireContext())
        adapter.updateData(DataRepository.services)
    }
}

override fun onDestroyView() {
    super.onDestroyView()
    _binding = null
}
}

=====

com\example\marmar\fragment\admin\AdminStatsFragment.kt
-----
package com.example.marmar.fragment.admin

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import com.example.marmar.R

```

```

import com.example.marmar.data.model.AppointmentStatus
import com.example.marmar.data.model.UserRole
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.FragmentAdminStatsBinding

class AdminStatsFragment : Fragment() {

    private var _binding: FragmentAdminStatsBinding? = null
    private val binding get() = _binding!!

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding = FragmentAdminStatsBinding.inflate(inflater,
container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        displayStats()
    }

    private fun displayStats() {
        // Статистика по пользователям
        val users = DataRepository.users
        binding.statsTotalUsersText.text =
getString(R.string.stats_total_users) + " ${users.size}"
        binding.statsAdminsText.text =
getString(R.string.stats_admins) + " ${users.count { it.role ==
UserRole.ADMIN }}"
        binding.statsMastersText.text =
getString(R.string.stats_masters) + " ${users.count { it.role ==
UserRole.MASTER }}"
        binding.statsClientsText.text =
getString(R.string.stats_clients) + " ${users.count { it.role ==
UserRole.CLIENT }}"

        // Статистика по доходам
        val completedAppointments =
DataRepository.appointments.filter { it.status ==
AppointmentStatus.COMPLETED }
        val totalRevenue = completedAppointments.sumOf {
            val priceStr =
DataRepository.getServiceById(it.serviceId)?.price?.replace(Regex(
"^[^0-9.-]"), "")?.split("-")?.firstOrNull()?.trim() ?: "0"
            priceStr.toDoubleOrNull() ?: 0.0
        }
        val salonIncome = totalRevenue * 0.30
    }
}

```

```

        binding.statsTotalRevenueText.text =
getString(R.string.stats_total_revenue) + "
${String.format("%.0f", totalRevenue).replace(",", " ")} руб."
        binding.statsSalonIncomeText.text =
getString(R.string.stats_salon_income) + "
${String.format("%.0f", salonIncome).replace(",", " ")} руб."

        // Статистика по записям
        val appointments = DataRepository.appointments
        binding.statsTotalAppointmentsText.text =
getString(R.string.stats_total_appointments) + "
${appointments.size}"
        binding.statsCompletedAppointmentsText.text =
getString(R.string.stats_completed_appointments) + "
${appointments.count { it.status == AppointmentStatus.COMPLETED
}}}"
        binding.statsCancelledAppointmentsText.text =
getString(R.string.stats_cancelled_appointments) + "
${appointments.count { it.status == AppointmentStatus.CANCELLED
}}}"
    }

    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}

```

```
=====
```

```

com\example\marmar\fragment\auth>LoginFragment.kt
-----
package com.example.marmar.fragment.auth

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.findNavController
import com.example.marmar.R
import com.example.marmar.data.model.UserRole
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.FragmentLoginBinding

class LoginFragment : Fragment() {

    private var _binding: FragmentLoginBinding? = null
    private val binding get() = _binding!!

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,

```

```

        savedInstanceState: Bundle?
    ): View {
        _binding = FragmentLoginBinding.inflate(inflater,
container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        binding.loginButton.setOnClickListener {
            handleLogin()
        }

        binding.registerPromptButton.setOnClickListener {
findNavController().navigate(R.id.action_loginFragment_to_register
Fragment)
        }
    }

    private fun handleLogin() {
        binding.passwordInputLayout.error = null

        val login = binding.loginEditText.text.toString().trim()
        val password = binding.passwordEditText.text.toString()

        if (login.isEmpty() || password.isEmpty()) {
            Toast.makeText(context,
getString(R.string.error_fields_empty), Toast.LENGTH_SHORT).show()
            return
        }

        val user = DataRepository.findUserByLogin(login)

        if (user != null && user.password == password) {
            DataRepository.currentUser = user
            Toast.makeText(context,
getString(R.string.success_login), Toast.LENGTH_SHORT).show()

            val isProfileFilled = !user.fullName.isNullOrBlank()
            && !user.phone.isNullOrBlank()

            if (isProfileFilled) {
                when (user.role) {
                    UserRole.CLIENT ->
findNavController().navigate(R.id.action_loginFragment_to_clientHo
meFragment)
                    UserRole.MASTER ->
findNavController().navigate(R.id.action_loginFragment_to_masterHo
meFragment)

```

```

                                UserRole.ADMIN ->
findNavController().navigate(R.id.action_loginFragment_to_adminHomeFragment)
                                }
                                } else {

findNavController().navigate(R.id.action_loginFragment_to_profileEditFragment)
                                }
                                } else {
                                    binding.passwordInputLayout.error =
getString(R.string.error_login_password_incorrect)
                                }
                            }

                            override fun onDestroyView() {
                                super.onDestroyView()
                                _binding = null
                            }
                        }
}

```

=====

com\example\marmar\fragment\auth\RegisterFragment.kt

package com.example.marmar.fragment.auth

```

import android.graphics.Color
import android.os.Bundle
import android.text.Editable
import android.text.TextWatcher
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.core.content.ContextCompat
import androidx.fragment.app.Fragment
import androidx.lifecycle.LifecycleScope
import androidx.navigation.fragment.findNavController
import com.example.marmar.R
import com.example.marmar.data.model.User
import com.example.marmar.data.model.UserRole
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.FragmentRegisterBinding
import kotlinx.coroutines.launch
import java.util.UUID

```

```

class RegisterFragment : Fragment() {

    private var _binding: FragmentRegisterBinding? = null
    private val binding get() = _binding!!

    override fun onCreateView(

```

```

        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding = FragmentRegisterBinding.inflate(inflater,
container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        setupPasswordStrengthChecker()

        binding.registerButton.setOnClickListener {
            handleRegistration()
        }

        // ДОБАВЛЕННЫЙ СЛУШАТЕЛЬ
        binding.loginPromptButton.setOnClickListener {

findNavController().navigate(R.id.action_registerFragment_to_login
Fragment)
        }

        // ... остальной код класса без изменений ...

        private fun handleRegistration() {
            binding.loginInputLayout.error = null
            binding.confirmPasswordInputLayout.error = null

            val login = binding.loginEditText.text.toString().trim()
            val password = binding.passwordEditText.text.toString()
            val confirmPassword =
binding.confirmPasswordEditText.text.toString()

            if (DataRepository.findUserByLogin(login) != null) {
                binding.loginInputLayout.error =
getString(R.string.error_login_taken)
                return
            }

            if (password != confirmPassword) {
                binding.confirmPasswordInputLayout.error =
getString(R.string.error_passwords_not_match)
                return
            }

            if (calculateStrength(password) < 4) {
                Toast.makeText(context,
getString(R.string.error_password_weak), Toast.LENGTH_LONG).show()
                return
            }
        }
    }

```

```

    }

    val newUser = User(
        id = UUID.randomUUID().toString(),
        login = login,
        password = password,
        role = UserRole.CLIENT
    )

    DataRepository.users.add(newUser)
    DataRepository.currentUser = newUser

    lifecycleScope.launch {
        DataRepository.saveData(requireContext())
        Toast.makeText(context,
            getString(R.string.success_registration),
            Toast.LENGTH_LONG).show()

        findNavController().navigate(R.id.action_registerFragment_to_profileEditFragment)
    }

    private fun setupPasswordStrengthChecker() {
        binding.passwordEditText.addTextChangedListener(object :
            TextWatcher {
                override fun beforeTextChanged(s: CharSequence?,
                    start: Int, count: Int, after: Int) {}
                override fun onTextChanged(s: CharSequence?, start:
                    Int, before: Int, count: Int) {
                    updatePasswordStrength(s.toString())
                }
                override fun afterTextChanged(s: Editable?) {}
            })
    }

    private fun updatePasswordStrength(password: String) {
        val strength = calculateStrength(password)
        binding.strengthProgress.progress = strength

        val strengthColor = when(strength) {
            0 -> Color.GRAY
            1 -> Color.RED
            2 -> Color.rgb(255, 165, 0) // Orange
            3 -> Color.YELLOW
            4 -> ContextCompat.getColor(requireContext(),
                R.color.success_green)
            else -> Color.GRAY
        }

        binding.strengthProgress.progressDrawable.setColorFilter(strengthC
            olor, android.graphics.PorterDuff.Mode.SRC_IN)
    }

```

```

        val hasLength = password.length >= 8
        val hasUpper = password.any { it.isUpperCase() }
        val hasDigit = password.any { it.isDigit() }
        val hasSpecial = password.any { !it.isLetterOrDigit() }

        val checkMark = "✓"
        val crossMark = "X"

        val criteriaText = buildString {
            appendLine("${if (hasLength) checkMark else crossMark}
Длина от 8 символов")
            appendLine("${if (hasUpper) checkMark else crossMark}
Заглавная буква")
            appendLine("${if (hasDigit) checkMark else crossMark}
Цифра")
            append("${if (hasSpecial) checkMark else crossMark}
Спец. символ (!@#...)")
        }
        binding.strengthCriteriaText.text = criteriaText
    }

    private fun calculateStrength(password: String): Int {
        var score = 0
        if (password.length >= 8) score++
        if (password.any { it.isUpperCase() }) score++
        if (password.any { it.isDigit() }) score++
        if (password.any { !it.isLetterOrDigit() }) score++
        return score
    }

    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}

```

=====

com\example\marmar\fragment\client\BookingFragment.kt

package com.example.marmar.fragment.client

```

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ArrayAdapter
import android.widget.AutoCompleteTextView
import android.widget.Toast
import androidx.fragment.app.Fragment
import androidx.lifecycle.LifecycleScope
import androidx.navigation.fragment.findNavController
import androidx.navigation.fragment.navArgs // <-- ВАЖНЫЙ ИМПОРТ

```

```

import com.example.marmar.R
import com.example.marmar.adapter.TimeSlotAdapter
import com.example.marmar.data.model.*
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.FragmentBookingBinding
import kotlinx.coroutines.launch
import java.util.*

class BookingFragment : Fragment() {

    private var _binding: FragmentBookingBinding? = null
    private val binding get() = _binding!!
    // Инициализация аргументов через делегат navArgs()
    private val args: BookingFragmentArgs by navArgs()

    private var selectedService: Service? = null
    private var selectedMaster: User? = null
    private var selectedDate: Calendar = Calendar.getInstance()
    private var selectedTime: String? = null

    private lateinit var timeSlotAdapter: TimeSlotAdapter

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding = FragmentBookingBinding.inflate(inflater,
container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        setupServiceDropdown()
        setupMasterDropdown()
        setupCalendar()
        setupTimeSlots()

        // Используем args для получения ID услуги
        args.serviceId?.let { serviceId ->
            val service = DataRepository.getServiceById(serviceId)
            service?.let {
                (binding.serviceMenuLayout.editText as?
AutoCompleteTextView)?.setText(it.name, false)
                selectedService = it
                updateMasterList()
            }
        }

        binding.confirmBookingButton.setOnClickListener {
confirmBooking() }

```

```

    }

    // ... остальной код BookingFragment без изменений ...
    private fun setupServiceDropdown() {
        val services = DataRepository.services
        val serviceNames = services.map { it.name }
        val adapter = ArrayAdapter(requireContext(),
            android.R.layout.simple_dropdown_item_1line, serviceNames)
        (binding.serviceMenuLayout.editText as?
            AutoCompleteTextView)?.setAdapter(adapter)

        (binding.serviceMenuLayout.editText as?
            AutoCompleteTextView)?.setOnItemClickListener { parent, _,
            position, _ ->
            selectedService = services.find { it.name ==
            parent.adapter.getItem(position) as String }
            updateMasterList()
            clearMasterSelection()
            updateAvailableTimeSlots()
        }
    }

    private fun setupMasterDropdown() {
        (binding.masterMenuLayout.editText as?
            AutoCompleteTextView)?.setOnItemClickListener { parent, _,
            position, _ ->
            val masterName = parent.adapter.getItem(position) as
            String
            selectedMaster = DataRepository.getMasters().find {
            it.fullName == masterName }
            updateAvailableTimeSlots()
        }
    }

    private fun setupCalendar() {
        val today = Calendar.getInstance()
        val twoWeeksLater = Calendar.getInstance().apply {
            add(Calendar.DAY_OF_YEAR, 14) }
        binding.calendarView.minDate = today.timeInMillis
        binding.calendarView.maxDate = twoWeeksLater.timeInMillis

        binding.calendarView.setOnDateChangeListener { _, year,
            month, dayOfMonth ->
            selectedDate.set(year, month, dayOfMonth)
            updateAvailableTimeSlots()
        }
    }

    private fun setupTimeSlots() {
        timeSlotAdapter = TimeSlotAdapter(emptyList()) { time ->
            selectedTime = time
        }
        binding.timeSlotsRecycler.adapter = timeSlotAdapter
    }

```

```

    }

    private fun updateMasterList() {
        val serviceId = selectedService?.id ?: return
        val availableMasters = DataRepository.getMasters().filter
{
            it.providedServiceIds?.contains(serviceId) == true
        }
        val masterNames = availableMasters.mapNotNull {
it.fullName }
        val adapter = ArrayAdapter(requireContext(),
android.R.layout.simple_dropdown_item_1line, masterNames)
        (binding.masterMenuLayout.editText as?
AutoCompleteTextView)?.setAdapter(adapter)
    }

    private fun clearMasterSelection() {
        (binding.masterMenuLayout.editText as?
AutoCompleteTextView)?.setText("", false)
        selectedMaster = null
    }

    private fun updateAvailableTimeSlots() {
        if (selectedMaster == null) {
            timeSlotAdapter.updateData(emptyList())
            binding.noTimeSlotsText.visibility = View.VISIBLE
            binding.noTimeSlotsText.text = "Сначала выберите
мастера"
            binding.timeSlotsRecycler.visibility = View.GONE
            return
        }

        val master = selectedMaster!!
        val workStart = master.workHoursStart ?: 9
        val workEnd = master.workHoursEnd ?: 18

        val allSlots = (workStart until workEnd).map { "$it:00" }

        val bookedSlots = DataRepository.appointments
            .filter {
                val appDate = Calendar.getInstance().apply {
timeInMillis = it.dateTime }
                it.masterId == master.id &&
                    appDate.get(Calendar.YEAR) ==
selectedDate.get(Calendar.YEAR) &&
                    appDate.get(Calendar.DAY_OF_YEAR) ==
selectedDate.get(Calendar.DAY_OF_YEAR)
            }
            .map { it.getFormattedTime() }

        val availableSlots = allSlots.filter { it !in bookedSlots
    }

```

```

        if (availableSlots.isEmpty()) {
            binding.noTimeSlotsText.visibility = View.VISIBLE
            binding.noTimeSlotsText.text =
getString(R.string.booking_no_available_time)
            binding.timeSlotsRecycler.visibility = View.GONE
        } else {
            binding.noTimeSlotsText.visibility = View.GONE
            binding.timeSlotsRecycler.visibility = View.VISIBLE
        }
        timeSlotAdapter.updateData(availableSlots)
    }

    private fun confirmBooking() {
        val clientId = DataRepository.currentUser?.id

        if (selectedService == null || selectedMaster == null ||
selectedTime == null || clientId == null) {
            Toast.makeText(context,
R.string.error_all_fields_required, Toast.LENGTH_SHORT).show()
            return
        }

        val hour = selectedTime!!.split(":")[0].toInt()
        selectedDate.set(Calendar.HOUR_OF_DAY, hour)
        selectedDate.set(Calendar.MINUTE, 0)
        selectedDate.set(Calendar.SECOND, 0)

        val newAppointment = Appointment(
            id = UUID.randomUUID().toString(),
            clientId = clientId,
            masterId = selectedMaster!!.id,
            serviceId = selectedService!!.id,
            dateTime = selectedDate.timeInMillis,
            status = AppointmentStatus.PENDING_CONFIRMATION
        )

        DataRepository.appointments.add(newAppointment)
        lifecycleScope.launch {
            DataRepository.saveData(requireContext())
            Toast.makeText(context,
R.string.booking_success_message, Toast.LENGTH_LONG).show()

            findNavController().navigate(R.id.action_bookingFragment_to_client
AppointmentsFragment)
        }
    }

    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}

```

```

=====

com\example\marmar\fragment\client\ClientAppointmentsFragment.kt
-----
package com.example.marmar.fragment.client

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.RatingBar
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.fragment.app.Fragment
import androidx.lifecycle.LifecycleScope
import androidx.navigation.fragment.findNavController
import com.example.marmar.R
import com.example.marmar.adapter.AppointmentAdapter
import com.example.marmar.data.model.Appointment
import com.example.marmar.data.model.AppointmentStatus
import com.example.marmar.data.model.Review
import com.example.marmar.data.model.ReviewTargetType
import com.example.marmar.data.repository.DataRepository
import com.example.marmar.databinding.DialogLeaveReviewBinding
import
com.example.marmar.databinding.FragmentClientAppointmentsBinding
import kotlinx.coroutines.launch
import java.util.*

class ClientAppointmentsFragment : Fragment(),
AppointmentAdapter.AppointmentInteractionListener {

    private var _binding: FragmentClientAppointmentsBinding? =
null
    private val binding get() = _binding!!

    private lateinit var activeAdapter: AppointmentAdapter
    private lateinit var completedAdapter: AppointmentAdapter

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding =
FragmentClientAppointmentsBinding.inflate(inflater, container,
false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        setupRecyclerViews()
    }

```

```

        loadAppointments()
    }

    private fun setupRecyclerViews() {
        activeAdapter = AppointmentAdapter(emptyList(), this)
        completedAdapter = AppointmentAdapter(emptyList(), this)
        binding.activeAppointmentsRecycler.adapter = activeAdapter
        binding.completedAppointmentsRecycler.adapter =
completedAdapter
    }

    private fun loadAppointments() {
        val userId = DataRepository.currentUser?.id ?: return
        val allAppointments =
DataRepository.getAppointmentsForClient(userId)

        val active = allAppointments.filter { it.status !=
AppointmentStatus.COMPLETED && it.status !=
AppointmentStatus.CANCELLED }
        val completed = allAppointments.filter { it.status ==
AppointmentStatus.COMPLETED || it.status ==
AppointmentStatus.CANCELLED }

        binding.noActiveAppointmentsText.visibility = if
(active.isEmpty()) View.VISIBLE else View.GONE
        binding.noCompletedAppointmentsText.visibility = if
(completed.isEmpty()) View.VISIBLE else View.GONE

        activeAdapter.updateData(active)
        completedAdapter.updateData(completed)
    }

    override fun onPayClicked(appointment: Appointment) {
        val action =
ClientAppointmentsFragmentDirections.actionClientAppointmentsFragm
entToPaymentFragment(appointment.id)
        findNavController().navigate(action)
    }

    override fun onLeaveReviewClicked(appointment: Appointment) {
        // Диалог с выбором, на что оставить отзыв
        val options = arrayOf("На мастера", "На услугу")
        AlertDialog.Builder(requireContext())
            .setTitle("На что вы хотите оставить отзыв?")
            .setItems(options) { _, which ->
                val targetType = if (which == 0)
ReviewTargetType.MASTER else ReviewTargetType.SERVICE
                showReviewDialog(appointment, targetType)
            }
            .show()
    }
}

```

```

app:destination="@id/masterHomeFragment"
app:popUpTo="@id/nav_graph" app:popUpToInclusive="true" />
    <action
android:id="@+id/action_profileEditFragment_to_adminHomeFragment"
app:destination="@id/adminHomeFragment"
app:popUpTo="@id/nav_graph" app:popUpToInclusive="true" />
    </fragment>

    <!-- CLIENT FLOW -->
    <fragment
        android:id="@+id/clientHomeFragment"

android:name="com.example.marmar.fragment.client.ClientHomeFragmen
t"
        android:label="Главное меню"
        tools:layout="@layout/fragment_client_home">
        <action
android:id="@+id/action_clientHomeFragment_to_masterListFragment"
app:destination="@id/nav_masters" />
            <action
android:id="@+id/action_clientHomeFragment_to_serviceListFragment"
app:destination="@id/nav_services" />
                <action
android:id="@+id/action_clientHomeFragment_to_clientAppointmentsFr
agment" app:destination="@id/nav_my_appointments" />
                    <action
android:id="@+id/action_clientHomeFragment_to_bookingFragment"
app:destination="@id/nav_booking" />
                        <action
android:id="@+id/action_clientHomeFragment_to_paymentFragment"
app:destination="@id/paymentFragment"/>
                            <action
android:id="@+id/action_clientHomeFragment_to_clientReviewsFragmen
t" app:destination="@id/nav_my_reviews" />
                                </fragment>
                                <fragment android:id="@+id/nav_masters"
android:name="com.example.marmar.fragment.client.MasterListFragmen
t" android:label="@string/masters_title"
tools:layout="@layout/fragment_master_list">
                                    <action
android:id="@+id/action_masterListFragment_to_masterDetailFragment
" app:destination="@id/masterDetailFragment" />
                                        </fragment>
                                        <fragment android:id="@+id/nav_services"
android:name="com.example.marmar.fragment.client.ServiceListFragme
nt" android:label="@string/services_title"
tools:layout="@layout/fragment_service_list">
                                            <action
android:id="@+id/action_serviceListFragment_to_bookingFragment"
app:destination="@id/nav_booking" />
                                                </fragment>
                                                <fragment android:id="@+id/nav_my_appointments"
android:name="com.example.marmar.fragment.client.ClientAppointment

```

```

sFragment" android:label="@string/my_appointments_title"
tools:layout="@layout/fragment_client_appointments">
    <action
android:id="@+id/action_clientAppointmentsFragment_to_paymentFragm
ent" app:destination="@id/paymentFragment" />
    </fragment>
    <fragment android:id="@+id/nav_my_reviews"
android:name="com.example.marmar.fragment.client.ClientReviewsFrag
ment" android:label="@string/my_reviews_title"
tools:layout="@layout/fragment_client_reviews"/>
    <fragment android:id="@+id/nav_booking"
android:name="com.example.marmar.fragment.client.BookingFragment"
android:label="@string/booking_title"
tools:layout="@layout/fragment_booking">
        <argument android:name="serviceId" app:argType="string"
app:nullable="true" android:defaultValue="null"/>
        <action
android:id="@+id/action_bookingFragment_to_clientAppointmentsFragm
ent" app:destination="@id/nav_my_appointments"
app:popUpTo="@id/clientHomeFragment"
app:popUpToInclusive="false"/>
        </fragment>
        <fragment android:id="@+id/masterDetailFragment"
android:name="com.example.marmar.fragment.client.MasterDetailFragm
ent" android:label="Профиль мастера"
tools:layout="@layout/fragment_master_detail">
            <argument android:name="masterId" app:argType="string" />
        </fragment>
        <fragment android:id="@+id/paymentFragment"
android:name="com.example.marmar.fragment.client.PaymentFragment"
android:label="Оплата" tools:layout="@layout/fragment_payment">
            <argument android:name="appointmentId"
app:argType="string" />
        </fragment>

<!-- MASTER FLOW -->
<fragment
    android:id="@+id/masterHomeFragment"

android:name="com.example.marmar.fragment.master.MasterHomeFragmen
t"
    android:label="Главная"
    tools:layout="@layout/fragment_master_home">
    <action
android:id="@+id/action_masterHomeFragment_to_nav_master_cabinet"
app:destination="@id/nav_master_cabinet" />
    <action
android:id="@+id/action_masterHomeFragment_to_nav_master_requests"
app:destination="@id/nav_master_requests" />
    <action
android:id="@+id/action_masterHomeFragment_to_nav_master_schedule"
app:destination="@id/nav_master_schedule" />

```

```

        <action
android:id="@+id/action_masterHomeFragment_to_nav_master_earnings"
app:destination="@id/nav_master_earnings" />
    </fragment>
    <fragment
        android:id="@+id/nav_master_cabinet"

android:name="com.example.marmar.fragment.master.MasterCabinetFrag
ment"
        android:label="@string/master_cabinet_title">
        <argument
            android:name="masterIdToEdit"
            app:argType="string" />
    </fragment>
    <fragment android:id="@+id/nav_master_requests"
android:name="com.example.marmar.fragment.master.MasterRequestsFra
gment" android:label="@string/master_requests_title" />
        <fragment android:id="@+id/nav_master_schedule"
android:name="com.example.marmar.fragment.master.MasterScheduleFra
gment" android:label="@string/master_schedule_title" />
        <fragment android:id="@+id/nav_master_reviews"
android:name="com.example.marmar.fragment.master.MasterReviewsFrag
ment" android:label="@string/master_reviews_title" />
        <fragment android:id="@+id/nav_master_earnings"
android:name="com.example.marmar.fragment.master.MasterEarningsFra
gment" android:label="@string/master_earnings_title" />

    <!-- ADMIN FLOW -->
    <fragment android:id="@+id/adminHomeFragment"
android:name="com.example.marmar.fragment.admin.AdminHomeFragment"
android:label="Панель управления"
tools:layout="@layout/fragment_admin_home"/>
        <fragment android:id="@+id/nav_admin_stats"
android:name="com.example.marmar.fragment.admin.AdminStatsFragment
" android:label="@string/admin_stats_title" />
        <fragment android:id="@+id/nav_admin_accounts"
android:name="com.example.marmar.fragment.admin.AdminAccountsFragm
ent" android:label="@string/admin_accounts_title">
            <action
android:id="@+id/action_adminAccountsFragment_to_profileEditFragme
nt" app:destination="@id/profileEditFragment" />
        </fragment>
        <fragment android:id="@+id/nav_admin_masters"
android:name="com.example.marmar.fragment.admin.AdminMastersFragme
nt" android:label="@string/admin_masters_title">
            <action
android:id="@+id/action_adminMastersFragment_to_nav_master_cabinet
" app:destination="@id/nav_master_cabinet" />
        </fragment>
        <fragment android:id="@+id/nav_admin_services"
android:name="com.example.marmar.fragment.admin.AdminServicesFragm
ent" android:label="@string/admin_services_title" />

```

```

        <fragment android:id="@+id/nav_admin_reviews"
android:name="com.example.marmar.fragment.admin.AdminReviewsFragme
nt" android:label="@string/admin_reviews_title" />
        <fragment android:id="@+id/nav_admin_appointments"
android:name="com.example.marmar.fragment.admin.AdminAppointmentsF
ragment" android:label="@string/admin_appointments_title" />

</navigation>

```

```
=====
```

```
values\colors.xml
```

```
-----
```

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFF</color>

    <!-- Marmar Palette -->
    <color name="pale_pink">#FFE3EE</color>
    <color name="pink">#FF78A5</color>
    <color name="purple_accent">#6A0DAD</color>

    <!-- UI Colors -->
    <color name="text_primary">#212121</color>
    <color name="text_secondary">#757575</color>
    <color name="divider">#BDBDBD</color>
    <color name="success_green">#4CAF50</color>
    <color name="error_red">#F44336</color>

    <color name="pattern_icon_color">#E0E0E0</color> <!-- Светло-
серый цвет для узора -->
</resources>

```

```
=====
```

```
values\strings.xml
```

```
-----
```

```

<resources>
    <string name="app_name">marmar - салон красоты</string>
    <string name="nav_header_desc">Аватар пользователя</string>
    <string name="login_welcome">Добро пожаловать в
Marmar</string>
    <string name="login_button_text">Войти</string>
    <string name="login_no_account_prompt">Нет аккаунта?
Зарегистрироваться</string>
    <string name="login_hint">Логин</string>
    <string name="password_hint">Пароль</string>
    <string name="register_title">Создание аккаунта</string>
    <string name="login_register_hint">Придумайте логин</string>
    <string name="password_register_hint">Придумайте
пароль</string>
    <string name="confirm_password_hint">Повторите пароль</string>

```

```

    <string name="password_strength_label">Надежность
    пароля:</string>
    <string name="password_strength_criteria">✓ Длина от 8
    символов\n✓ Заглавная буква\n✓ Цифра\n✓ Спец. символ</string>
    <string
    name="register_button_text">Зарегистрироваться</string>
    <string name="error_fields_empty">Логин и пароль не могут быть
    пустыми</string>
    <string name="error_login_password_incorrect">Неправильный
    логин или пароль</string>
    <string name="success_login">Успешный вход!</string>
    <string name="error_login_taken">Этот логин уже занят</string>
    <string name="error_passwords_not_match">Пароли не
    совпадают</string>
    <string name="error_password_weak">Пароль слишком слабый.
    Следуйте критериям надежности.</string>
    <string name="success_registration">Регистрация успешна!
    Теперь заполните свой профиль.</string>
    <string name="logo_content_description">Логотип салона
    Marmar</string>

    <!-- Client Home -->
    <string name="client_status_title">Ваш статус:</string>
    <string name="client_status_client">Клиент</string>
    <string name="client_status_lover">Любимый клиент</string>
    <string name="client_status_vip">VIP клиент</string>
    <string name="to_next_status_prefix">До следующего
    статуса:</string>
    <string name="to_next_status_suffix">записей</string>
    <string name="status_vip_max">Вы достигли максимального
    статуса!</string>
    <string name="home_upcoming_appointments">Предстоящие
    записи</string>
    <string name="home_recommendations">Рекомендации</string>
    <string name="home_no_upcoming_appointments">У вас нет
    предстоящих записей. Время записаться!</string>
    <string name="home_recommendation_leave_review">Оставить отзыв
    на %1$s</string>
    <string name="home_tile_masters">Наши мастера</string>
    <string name="home_tile_services">Наши услуги</string>
    <string name="home_tile_book">Записаться</string>

    <!-- Appointments -->
    <string name="my_appointments_title">Мои записи</string>
    <string name="appointments_active_header">Актуальные</string>
    <string
    name="appointments_completed_header">Завершенные</string>
    <string name="button_pay">Оплатить</string>
    <string name="button_leave_review">Оставить отзыв</string>
    <string name="status_pending">Ожидает подтверждения</string>
    <string name="status_awaiting_payment">Ожидает оплаты</string>
    <string name="status_confirmed">Подтверждено</string>
    <string name="status_completed">Завершено</string>

```

```

    <string name="status_cancelled">Отменено</string>
    <string name="cancellation_reason">Причина отмены:
%1$s</string>

    <!-- Payment Screen -->
    <string name="payment_bank_name">✦ Гламурный Банк ✦</string>
    <string name="payment_sum">Сумма к оплате:</string>
    <string name="payment_recipient">Получатель:</string>
    <string name="payment_recipient_value">Салон красоты
"Marmar"</string>
    <string name="payment_payer">Плательщик:</string>
    <string name="payment_success">Оплачено!</string>
    <string name="payment_processing">Обработка
платежа...</string>
    <string name="button_pay_confirm">Оплатить %1$s</string>

    <!-- Reviews -->
    <string name="my_reviews_title">Мои отзывы</string>
    <string name="review_on_master">Отзыв на мастера:
%1$s</string>
    <string name="review_on_service">Отзыв на услугу:
%1$s</string>
    <string name="review_reply_prefix">Ответ:</string>
    <string name="dialog_leave_review_title">Оставить
отзыв</string>
    <string name="dialog_review_rating_label">Ваша
оценка:</string>
    <string name="dialog_review_text_hint">Напишите здесь ваш
отзыв...</string>
    <string name="dialog_button_send">Отправить</string>
    <string name="dialog_button_cancel">Отмена</string>
    <string name="review_success_toast">Спасибо за ваш
отзыв!</string>

    <!-- Masters -->
    <string name="masters_title">Мастера</string>
    <string name="master_detail_experience">Стаж: %1$d
лет</string>
    <string
name="master_detail_specialization">Специализация:</string>
    <string name="master_detail_services">Оказываемые
услуги:</string>
    <string name="master_detail_reviews">Отзывы клиентов:</string>
    <string name="master_detail_no_reviews">На этого мастера еще
нет отзывов.</string>
    <string name="master_average_rating">Средний рейтинг:
%.1f</string>

    <!-- Services -->
    <string name="services_title">Услуги</string>

    <!-- Booking -->
    <string name="booking_title">Запись на приём</string>

```

```

    <string name="booking_step1_service">1. Выберите
услуги</string>
    <string name="booking_step2_master">2. Выберите
мастера</string>
    <string name="booking_step3_date">3. Выберите дату</string>
    <string name="booking_step4_time">4. Выберите время</string>
    <string name="booking_no_available_time">На выбранную дату нет
свободного времени</string>
    <string name="booking_button_confirm">Записаться</string>
    <string name="booking_success_message">Вы успешно записаны!
Следите за статусом в разделе \"Мои записи\".</string>
    <string name="error_all_fields_required">Пожалуйста, выберите
услугу, мастера, дату и время</string>

    <!-- Master Cabinet -->
    <string name="master_cabinet_title">Рабочий кабинет</string>
    <string name="cabinet_work_days_title">Рабочие дни</string>
    <string name="day_monday">ПН</string>
    <string name="day_tuesday">БТ</string>
    <string name="day_wednesday">СР</string>
    <string name="day_thursday">ЧТ</string>
    <string name="day_friday">ПТ</string>
    <string name="day_saturday">СБ</string>
    <string name="day_sunday">ВС</string>
    <string name="cabinet_work_hours_title">Рабочие часы</string>
    <string name="cabinet_work_hours_from">С</string>
    <string name="cabinet_work_hours_to">До</string>
    <string name="cabinet_provided_services_title">Оказываемые
услуги</string>
    <string name="button_save_changes">Сохранить
изменения</string>
    <string name="cabinet_save_success">Настройки успешно
сохранены!</string>
    <string name="error_invalid_hours">Время \"С\" должно быть
раньше времени \"До\"</string>

    <!-- Master Requests -->
    <string name="master_requests_title">Заявки на запись</string>
    <string name="requests_pending_header">Ожидают
подтверждения</string>
    <string name="requests_active_header">Активные записи</string>
    <string name="button_confirm">Подтвердить</string>
    <string name="button_cancel_appointment">Отменить</string>
    <string name="dialog_cancel_reason_title">Причина
отмены</string>
    <string name="dialog_cancel_reason_hint">Укажите причину
отмены записи...</string>
    <string name="error_reason_required">Причина отмены
обязательна</string>
    <string name="appointment_cancelled_toast">Запись
отменена</string>

    <!-- Master Reviews -->

```

```

    <string name="master_reviews_title">Мои отзывы</string>
    <string name="button_reply">Ответить</string>
    <string name="dialog_add_reply_title">Ваш ответ на
отзыв</string>
    <string name="dialog_add_reply_hint">Напишите здесь ваш
ответ...</string>
    <string name="reply_sent_success">Ответ успешно
отправлен!</string>

    <!-- Master Schedule -->
    <string name="master_schedule_title">Моё расписание</string>
    <string name="schedule_total_for_day">Итог за день:
%s</string>
    <string name="schedule_no_appointments_for_day">Нет записей на
этот день</string>

    <!-- Master Earnings -->
    <string name="master_earnings_title">Мои доходы</string>
    <string name="earnings_period">За последние 30 дней</string>
    <string name="earnings_total_revenue">Общая выручка</string>
    <string name="earnings_salon_share">Доля салона (30%)</string>
    <string name="earnings_your_income">Итого к получению</string>
    <string name="earnings_no_data">Нет данных о доходах за этот
период</string>

    <!-- Admin -->
    <string name="admin_stats_title">Сводная статистика</string>
    <string name="stats_users_title">Пользователи</string>
    <string name="stats_total_users">Всего аккаунтов:</string>
    <string name="stats_admins">Администраторы:</string>
    <string name="stats_masters">Мастера:</string>
    <string name="stats_clients">Клиенты:</string>
    <string name="stats_income_title">Доходы</string>
    <string name="stats_total_revenue">Общая выручка (за всё
время):</string>
    <string name="stats_salon_income">Доход салона (за всё
время):</string>
    <string name="stats_appointments_title">Записи</string>
    <string name="stats_total_appointments">Всего
записей:</string>
    <string
name="stats_completed_appointments">Завершено:</string>
    <string name="stats_cancelled_appointments">Отменено:</string>

    <string name="admin_accounts_title">Управление
аккаунтами</string>
    <string name="dialog_change_role_title">Сменить роль для
%s</string>
    <string name="dialog_confirm_password_title">Подтверждение
действия</string>
    <string name="dialog_confirm_password_prompt">Для смены роли
на \ "Администратор\ ", введите ваш текущий пароль.</string>

```

```

    <string name="dialog_delete_confirmation_title">Подтверждение
удаления</string>
    <string name="dialog_delete_user_prompt">Вы уверены, что
хотите удалить пользователя %s? Это действие необратимо.</string>
    <string name="dialog_delete_service_prompt">Вы уверены, что
хотите удалить услугу \"%s\"? Это действие необратимо.</string>
    <string name="dialog_delete_review_prompt">Вы уверены, что
хотите удалить этот отзыв? Это действие необратимо.</string>
    <string name="dialog_delete_appointment_prompt">Вы уверены,
что хотите удалить эту запись? Это действие необратимо.</string>
    <string name="button_delete">Удалить</string>
    <string name="button_change_role">Сменить роль</string>
    <string name="button_edit">Редактировать</string>

```

```

    <string name="admin_services_title">Управление
услугами</string>
    <string name="dialog_service_add_title">Добавить новую
услугу</string>
    <string name="dialog_service_edit_title">Редактировать
услугу</string>
    <string name="service_name_hint">Название услуги</string>
    <string name="service_price_hint">Цена (напр., 1500 руб. или
2000-3000 руб.)</string>
    <string name="service_description_hint">Описание
услуги</string>

```

```

    <string name="admin_masters_title">Управление
мастерами</string>
    <string name="admin_reviews_title">Управление
отзывами</string>
    <string name="admin_appointments_title">Управление
записями</string>
</resources>

```

=====

values\themes.xml

```

<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Base.Theme.Marmar"
parent="Theme.Material3.DayNight.NoActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/pink</item>
        <item name="colorPrimaryVariant">@color/pale_pink</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/purple_accent</item>
        <item
name="colorSecondaryVariant">@color/purple_accent</item>
        <item name="colorOnSecondary">@color/white</item>
        <!-- Status bar color. -->

```

```

        <item
name="android:statusBarColor"?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>

    <style name="Theme.Marmar" parent="Base.Theme.Marmar" />

    <!-- ДОБАВЬТЕ ЭТИ ДВА СТИЛЯ -->
    <style name="Theme.Marmar.AppBarOverlay"
parent="ThemeOverlay.AppCompat.Dark.ActionBar" />
    <style name="Theme.Marmar.PopupOverlay"
parent="ThemeOverlay.AppCompat.Light" />

    <style name="RatingBar" parent="Theme.AppCompat">
        <item name="colorControlNormal">@color/divider</item>
        <item name="colorControlActivated">@color/pink</item>
    </style>

</resources>

=====

xml\backup_rules.xml
-----
<?xml version="1.0" encoding="utf-8"?><!--
    Sample backup rules file; uncomment and customize as necessary.
    See https://developer.android.com/guide/topics/data/autobackup
    for details.
    Note: This file is ignored for devices older than API 31
    See https://developer.android.com/about/versions/12/backup-
restore
-->
<full-backup-content>
    <!--
        <include domain="sharedpref" path="."/>
        <exclude domain="sharedpref" path="device.xml"/>
    -->
</full-backup-content>

=====

xml\data_extraction_rules.xml
-----
<?xml version="1.0" encoding="utf-8"?><!--
    Sample data extraction rules file; uncomment and customize as
    necessary.
    See https://developer.android.com/about/versions/12/backup-
restore#xml-changes
    for details.
-->
<data-extraction-rules>
    <cloud-backup>

```

```
        <!-- TODO: Use <include> and <exclude> to control what is
backed up.
        <include .../>
        <exclude .../>
        -->
    </cloud-backup>
    <!--
    <device-transfer>
        <include .../>
        <exclude .../>
    </device-transfer>
    -->
</data-extraction-rules>
```

=====

Приложение Д
(рекомендуемое)
Результаты тестирования серверной части

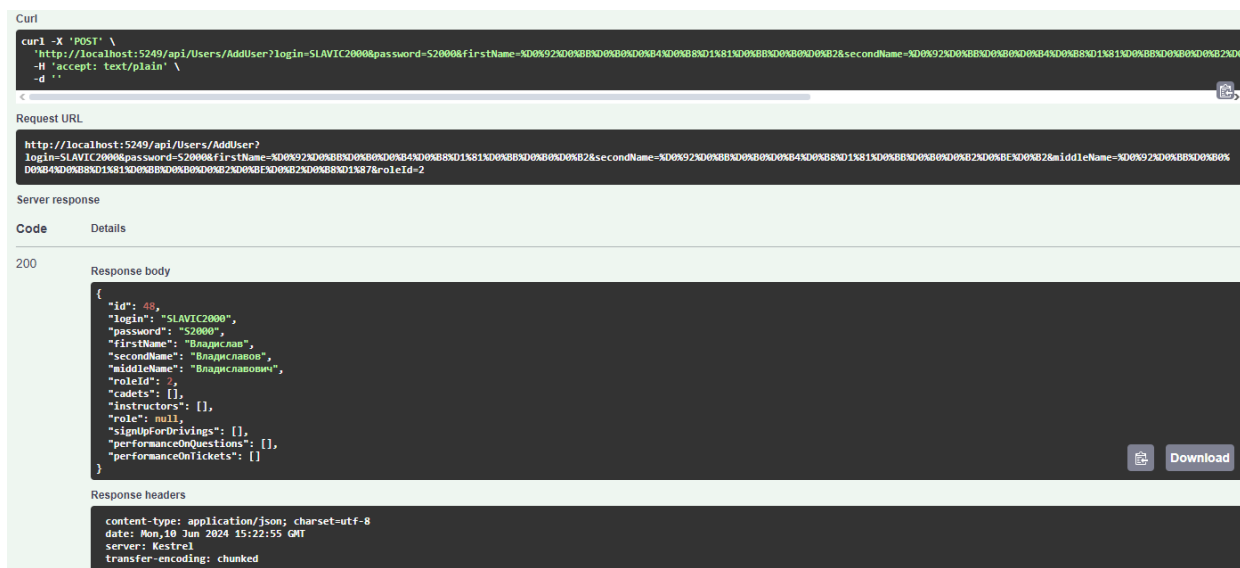


Рисунок Д.1 – Добавление нового пользователя



Рисунок Д.2 – Вход существующего пользователя в систему



Рисунок Д.3 – Получение информации о пользователе

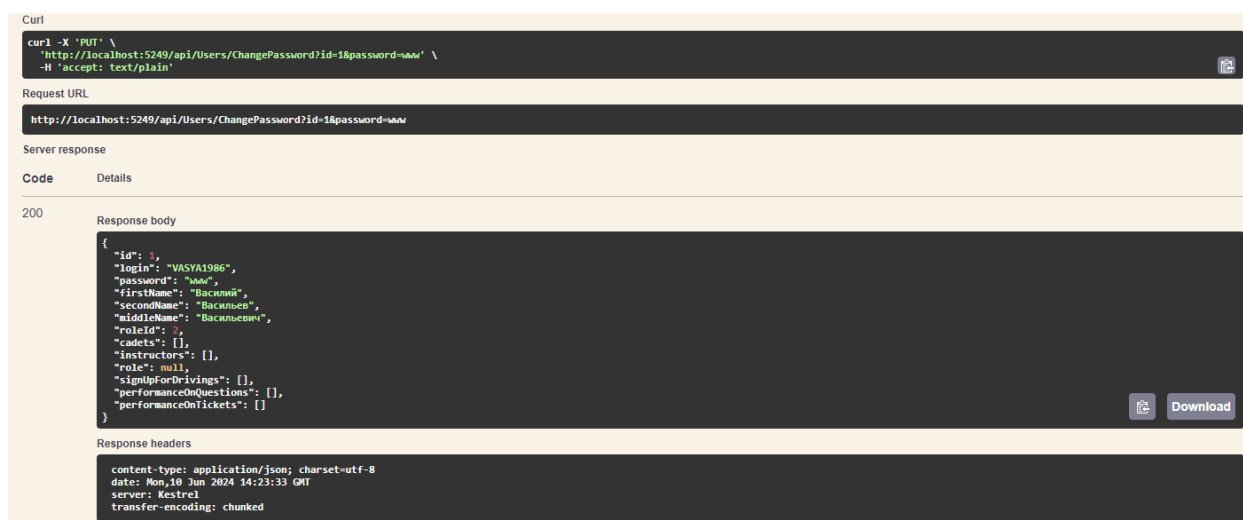


Рисунок Д.4 – Изменение пароля пользователя



Рисунок Д.5 – Запись на процедуру



Рисунок Д.6 – Отмена записи на процедуру

Приложение Е
(рекомендуемое)

Результаты тестирования клиентской части

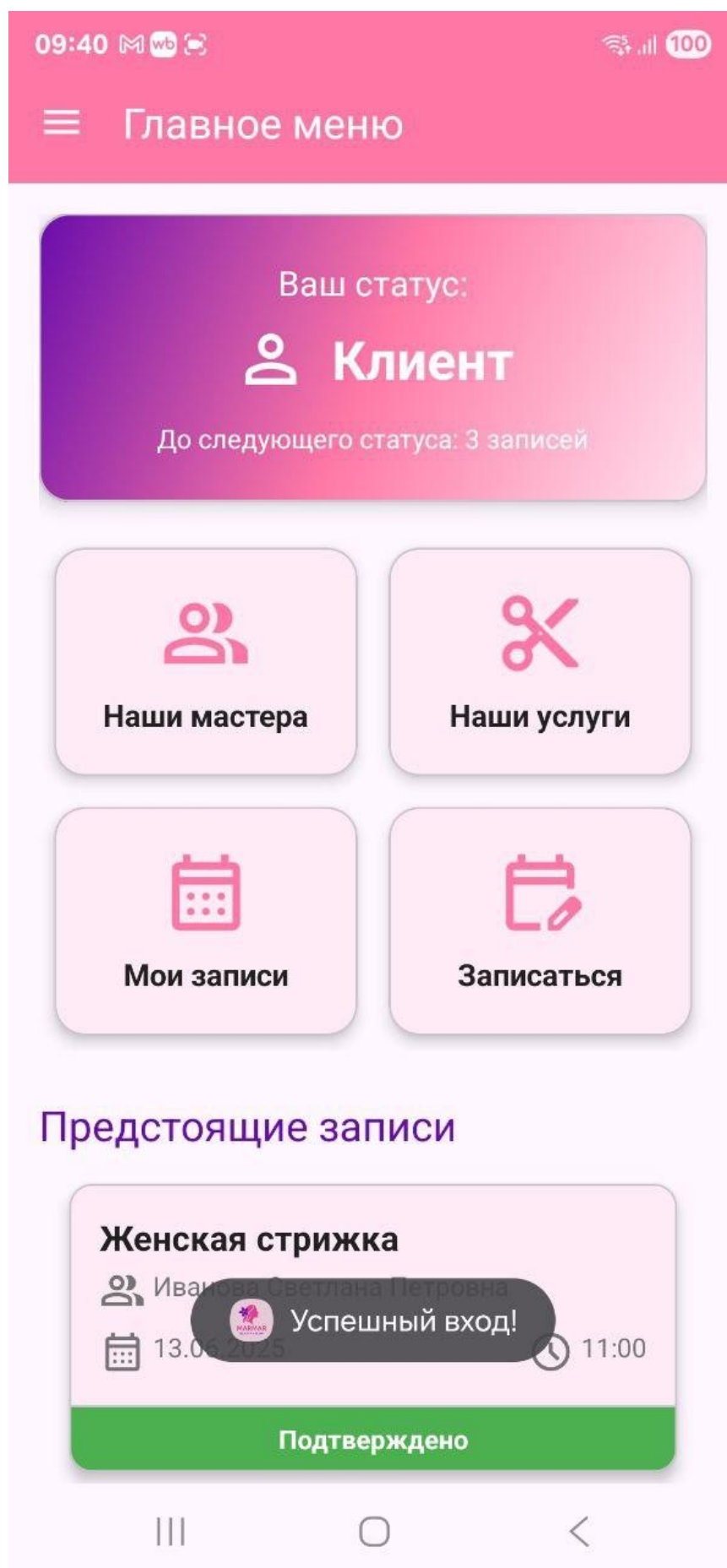


Рисунок Е.1 – Успешная регистрация в системе

09:40 M wb 5G 100

← Редактирование профиля

Заполните ваш профиль



Выбрать фото

ФИО (обязательно)

Петрова Полина Петровна

Номер телефона (обязательно)

+79112223344

Возраст

28

Электронная почта (по желанию)

polina@mail.com

Сохранить и продолжить

III O <

Рисунок Е.2 – Просмотр профиля

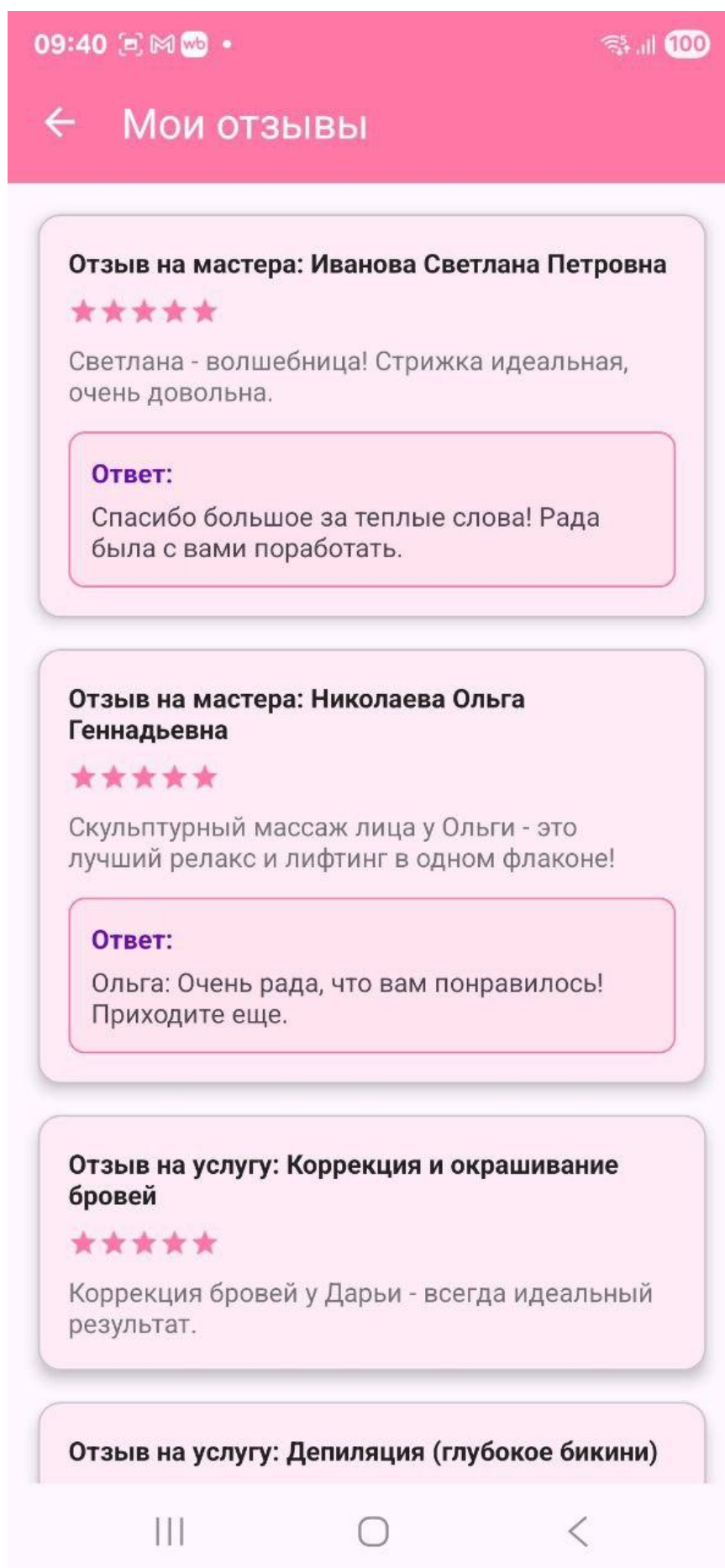


Рисунок Е.3 – Оставление отзыва

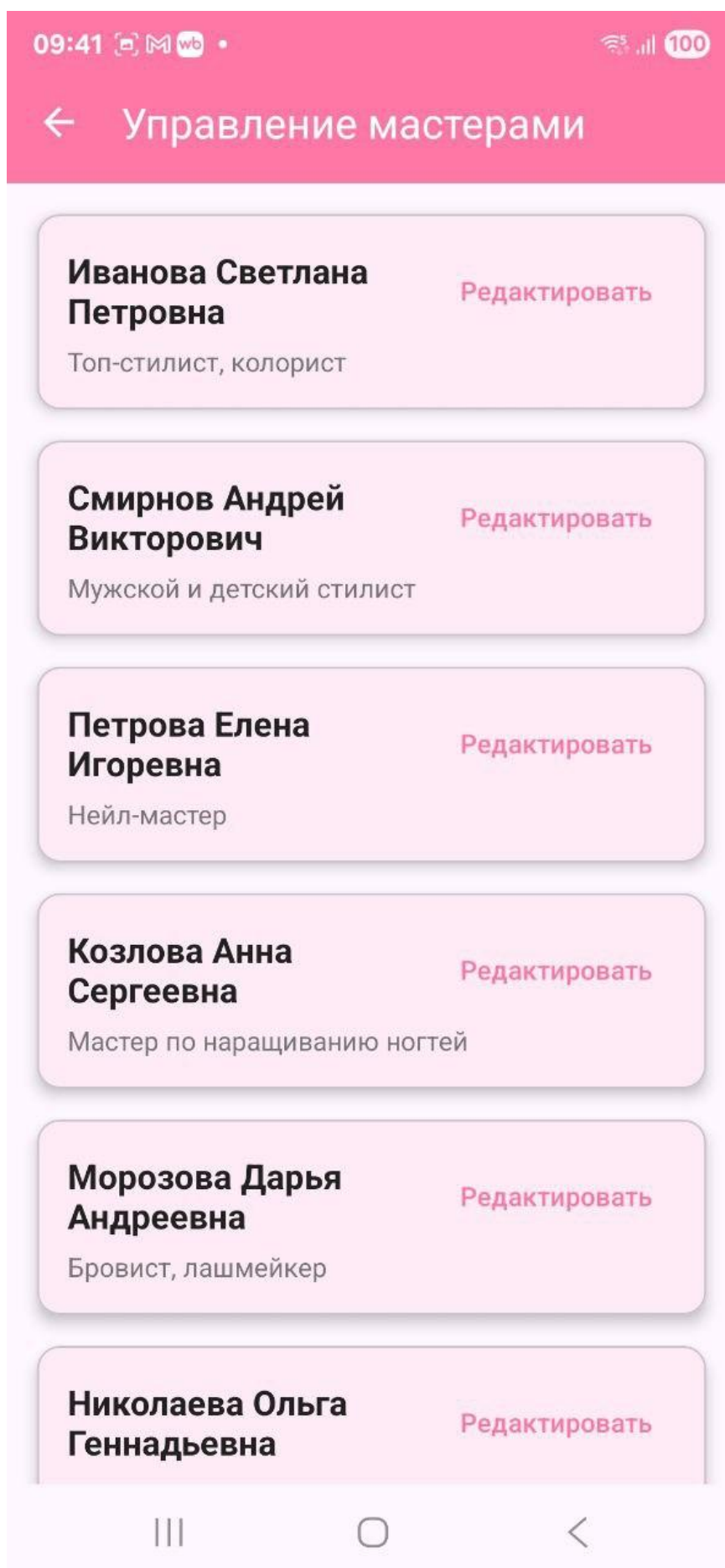


Рисунок Е.4 – Управление мастерами

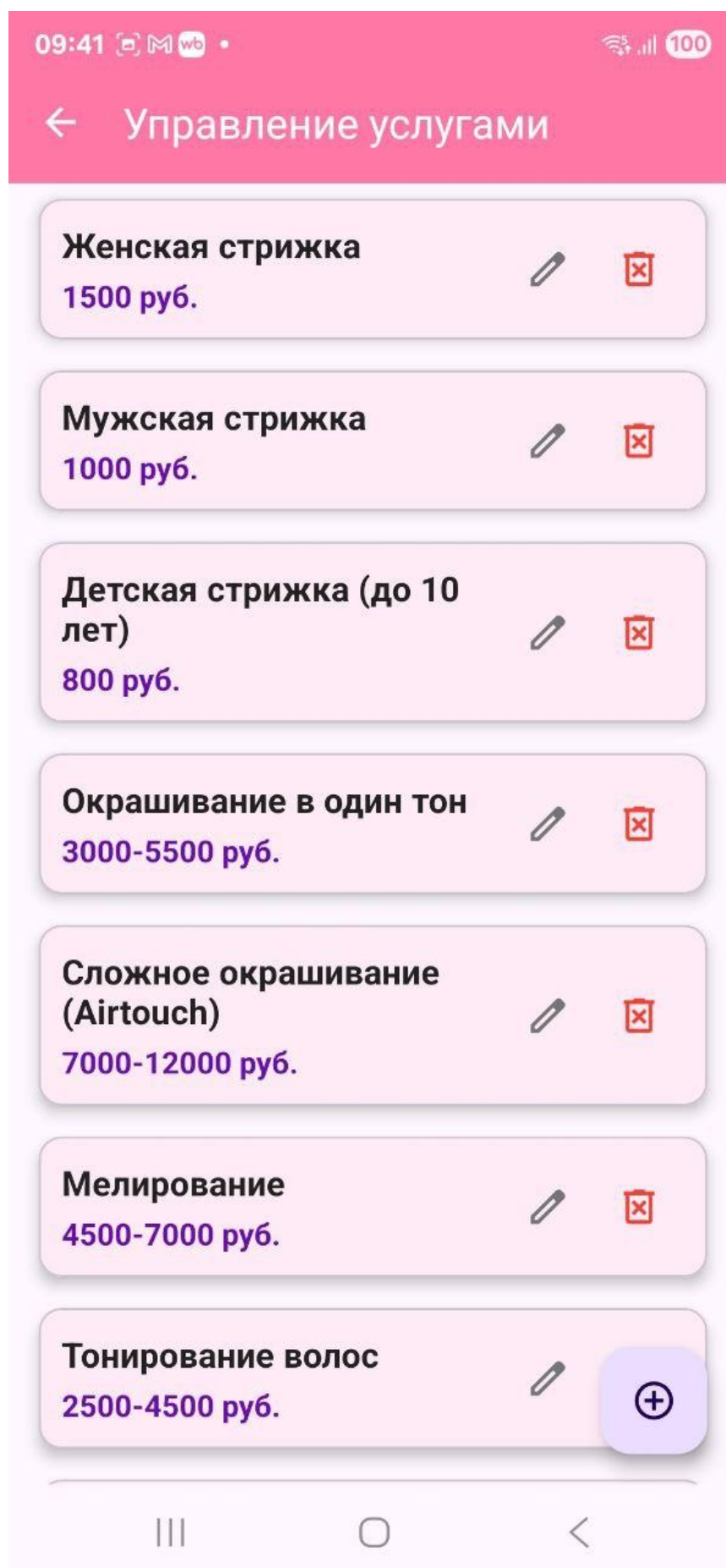


Рисунок Е.5 – Управление услугами

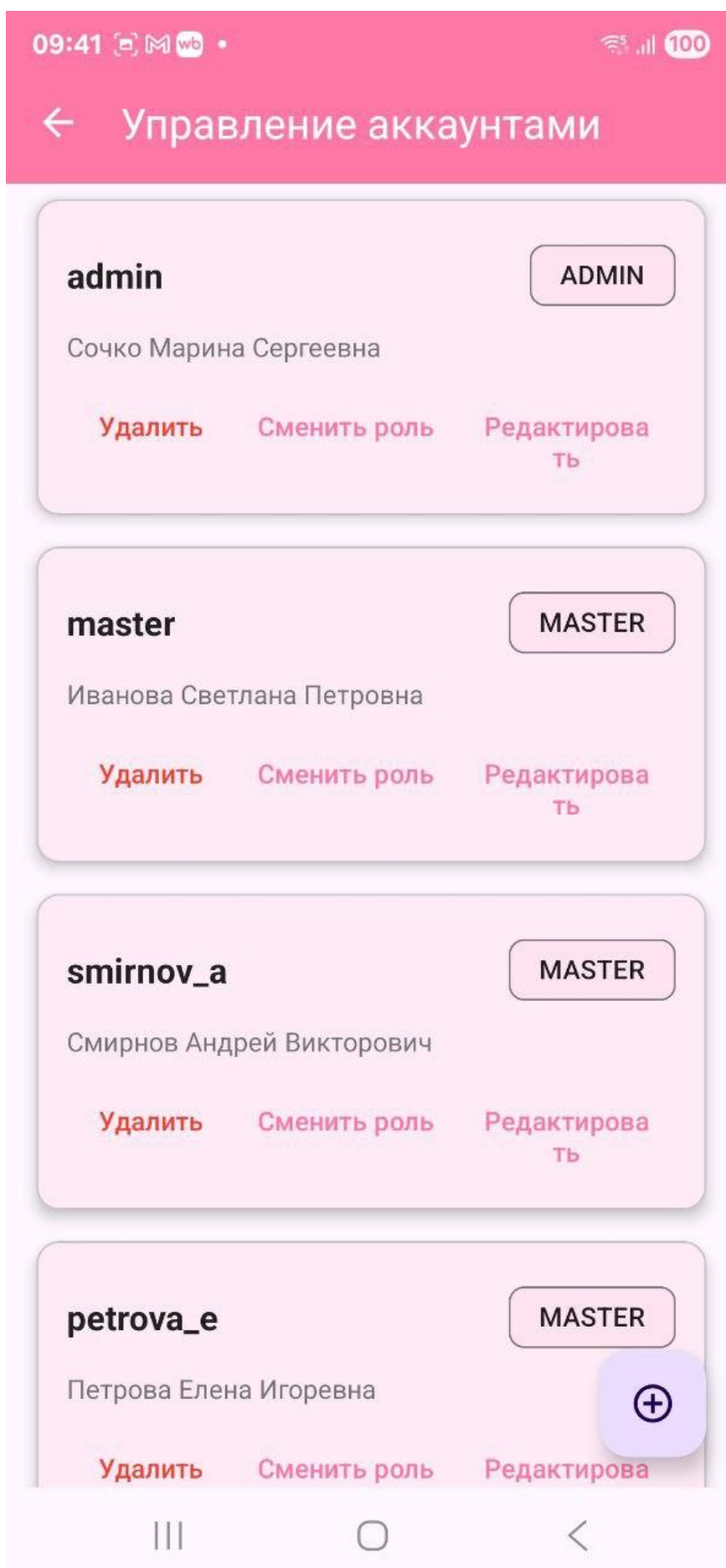


Рисунок Е.6 – Управление аккаунтами

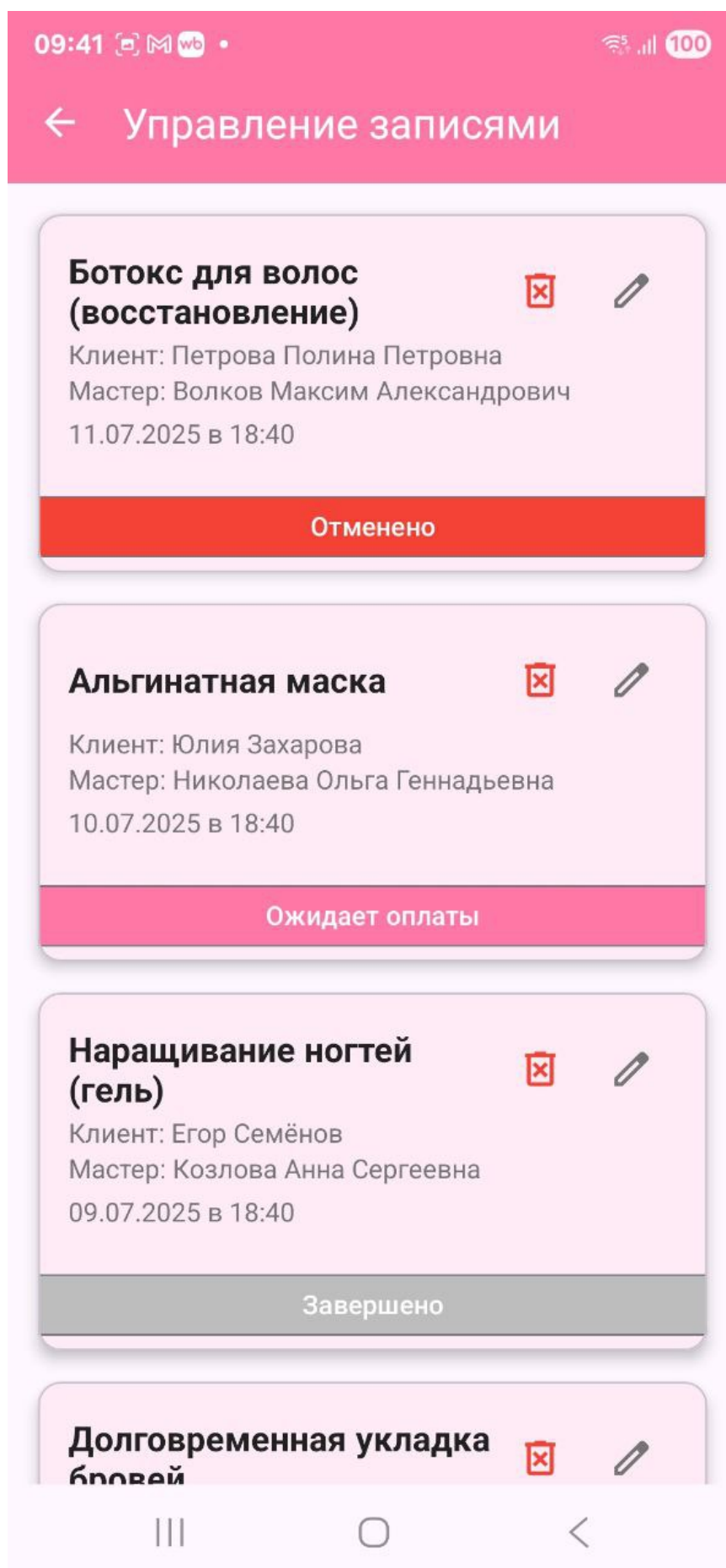


Рисунок Е.7 – Управление записями

09:41 wb • 100

Создание аккаунта

Придумайте логин

Придумайте пароль

Надежность пароля:

✓ Длина от 8 символов

✓ Заглавная буква

✓ Цифра

✓ Спец. символ

Повторите пароль

Зарегистрироваться

Есть аккаунт? Войдите

Рисунок Е.8 – Регистрация

09:42  100

← Запись на приём

1. Выберите услугу

Выберите услугу
Женская стрижка

2. Выберите мастера

Выберите мастера
Иванова Светлана Петровна

3. Выберите дату

Июнь 2025 г.

П	В	С	Ч	П	С	В
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

4. Выберите время

9:00

Рисунок Е.9 – Запись на свободное время

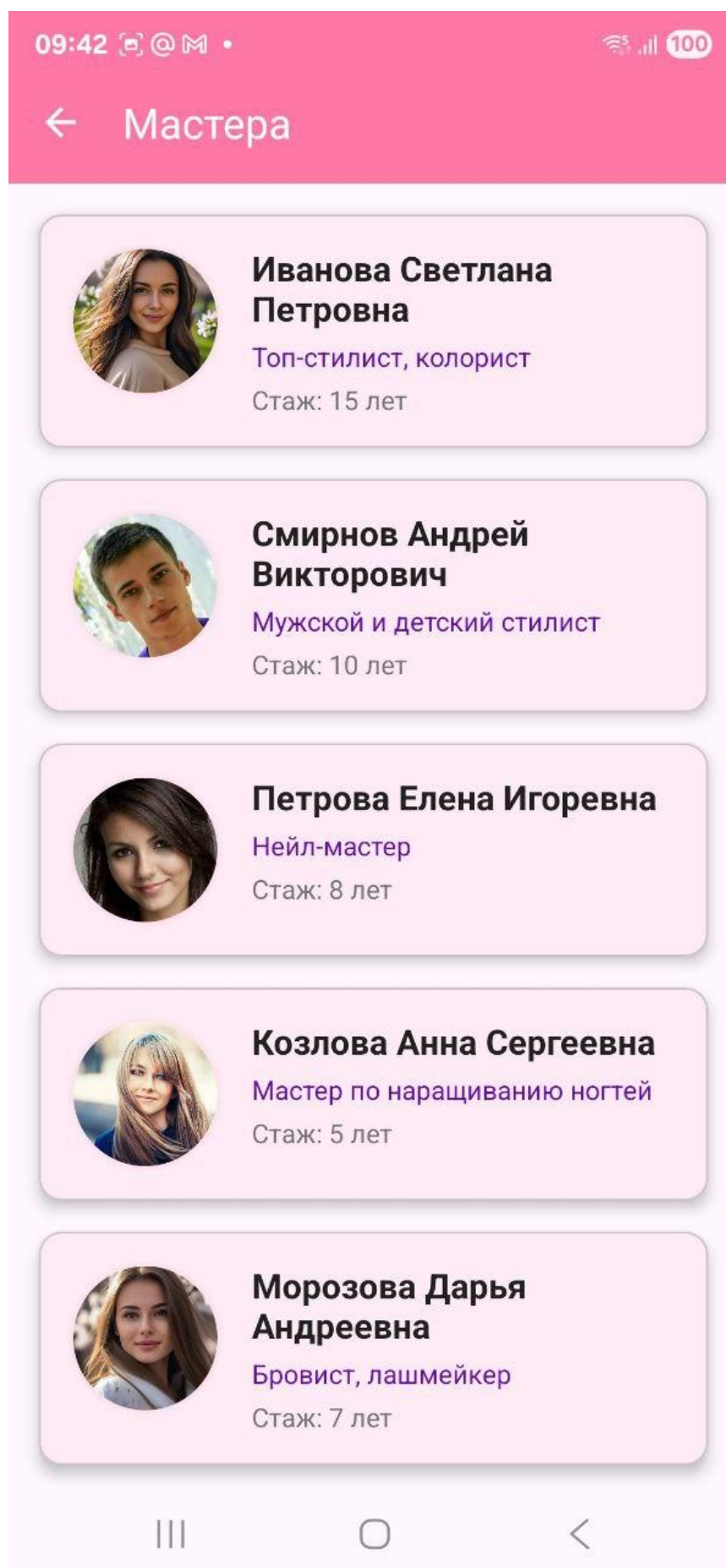


Рисунок Е.10 – Просмотр мастеров

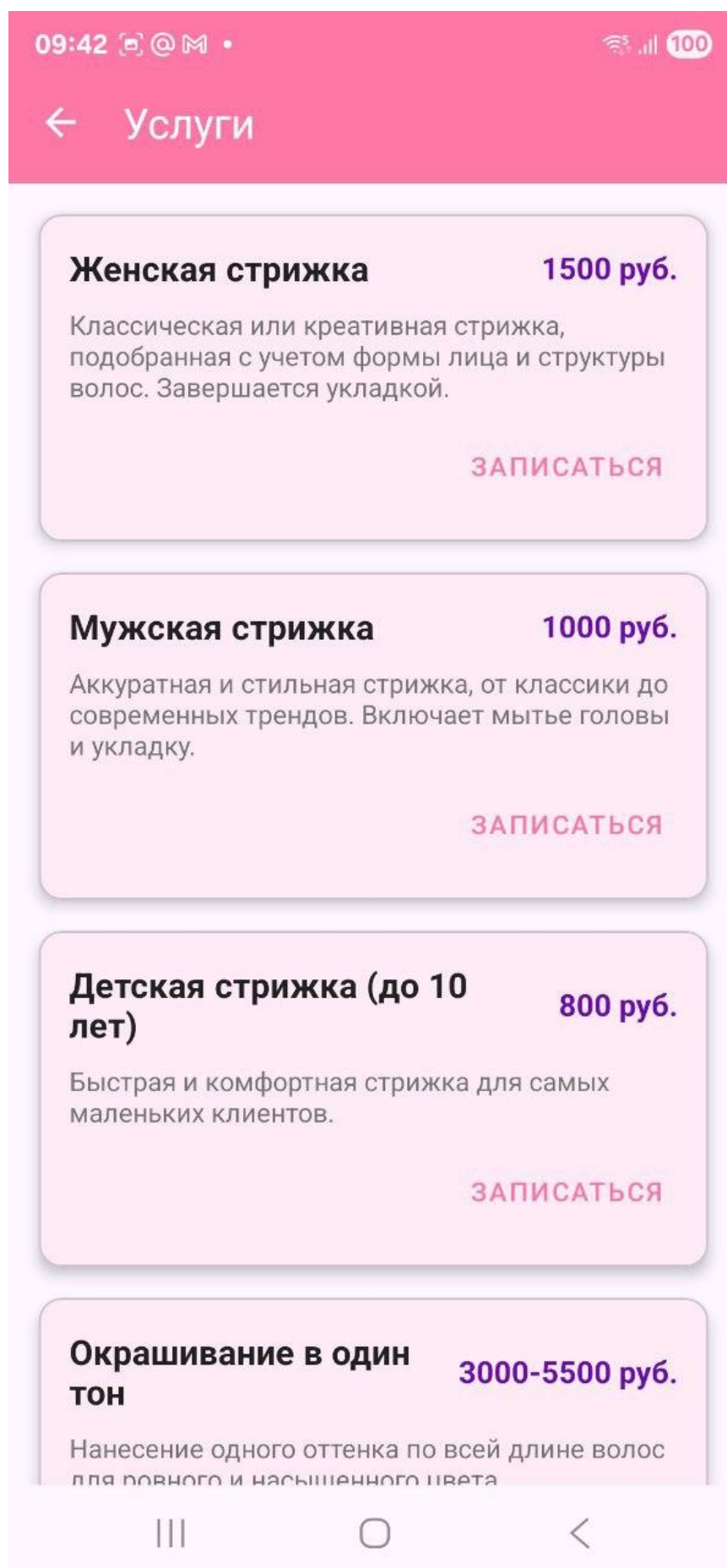


Рисунок Е.11 – Просмотр услуг

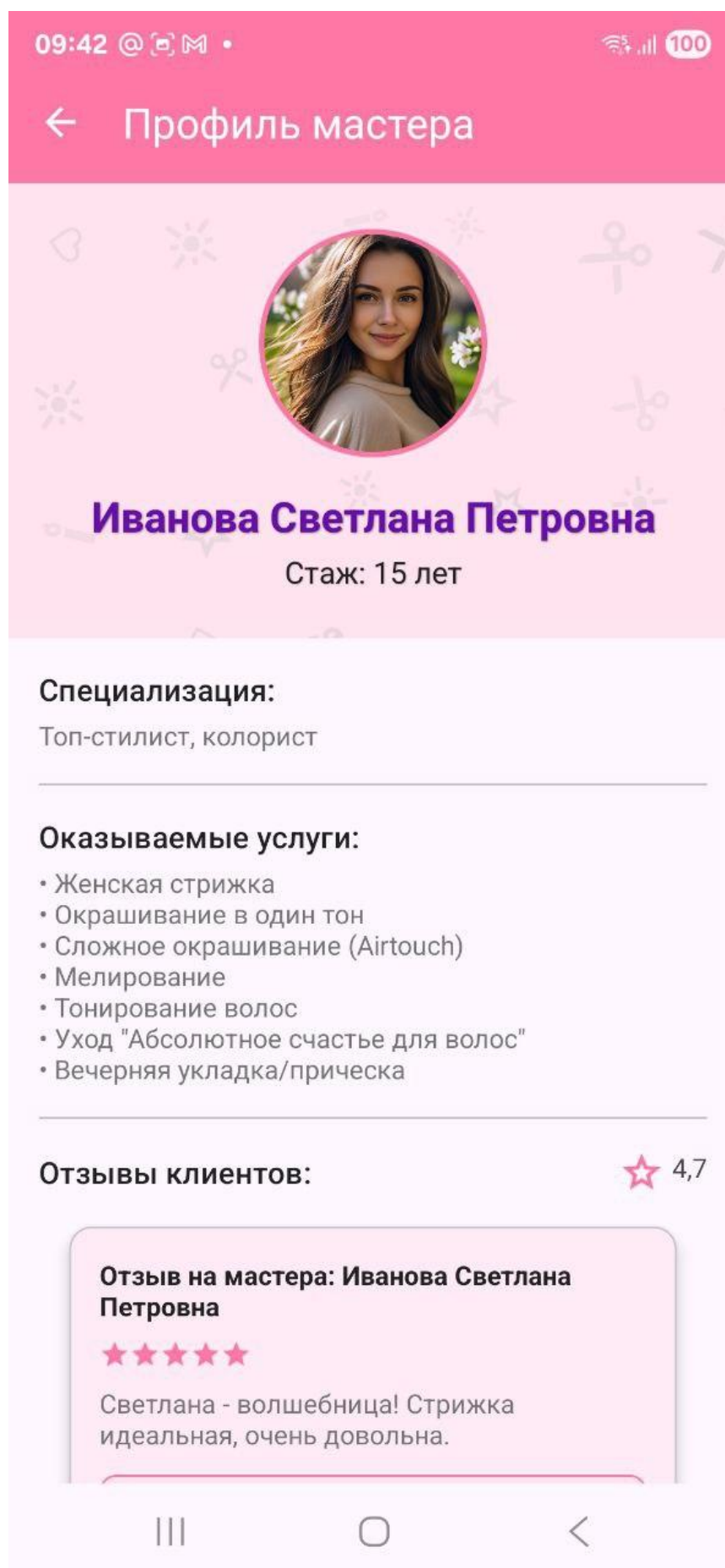


Рисунок Е.12– Просмотр портфолио