МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
вычислительной техники	МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

Мобильное программное приложение для поддержки процессов функционирования предприятия автосервиса

Студент _____ ___Вишникин Иван Сергеевич_
(подпись, дата) (ФИО полностью)

Руководитель _____ _____Князев В.Н._____
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ _____Такташкин Д.В._____
(подпись, дата) (фамилия, инициалы)

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

Заведующий кафедрой _____ _____Козлов А.Ю._____
(подпись) (фамилия, инициалы)

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

Секретарь ГЭК _____ _____Попова Н.А._____
(подпись) (фамилия, инициалы)

Пенза, 2025

Федеральное государственное бюджетное образовательное учреждение высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____А.Ю. Козлов
«20» декабря 2024 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент____Вишникин Иван Сергеевич_____гр._21ВП2_факультета___ВТ___
<Фамилия, имя, отчество полностью>
2.Тема работы____Мобильное программное приложение для поддержки процессов функционирования предприятия автосервиса_____
_____
_____
Тема утверждена приказом ПГУ №_____от "_____"_____20_____

2. Руководитель работы____ктн доцент Князев В.Н._____

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

1) Разработать приложение для мобильных устройств, обеспечивающее поддержку процессов функционирования автосервиса.
2) Среда выполнения приложения: операционная система Android 10 и выше
Среда выполнения сервера: операционная система Windows 10 и выше
Языковые средства разработки: C#, UML, SQL, XAML
Инструментальные средства разработки: Microsoft SQL Server 2019, Microsoft Visual Studio 2022, Microsoft Project, Visual Paradigm
Технологии разработки: RUP, ООП, Entity Framework, MAUI
3) Основные функции мобильного приложения:
а) просмотр списка услуг и цен;
б) запись на услугу автосервиса;
в) просмотр истории обслуживания;
г) получение уведомлений.

4. Перечень подлежащих разработке вопросов

1) *Анализ предметной области и постановка задачи*
2) *Проектирование*
3) *Реализация*
4) *Контроль качества программного обеспечения*
5) *Оформление пояснительной записки*

5. 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 – 14.04.2025	
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 165 листов, 56 рисунков, 16 таблиц, 29 использованных источника, 7 приложений.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, АВТОСЕРВИС, ANDROID, MS SQL SERVER, C#, MAUI, .NET, RUP, UML

Объектом разработки является мобильное приложение для поддержки процессов функционирования автосервиса.

Цель работы – разработка мобильного программного приложения для поддержки процессов функционирования автосервиса, позволяющего просматривать список услуг и цен, записываться на услуги, просматривать историю обслуживания, а также контролировать процессы работы автосервиса.

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

Разработка приложения производилась на языке программирования C#. В качестве среды разработки использовалась среда программирования Microsoft Visual Studio 2022, операционная система Windows 10. Приложение предназначено для использования под управлением ОС Android версии 10.0 и выше, а его серверная часть работает под управлением операционной системы Windows 10 и выше.

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 «Wilgood Авто» 10
1.2.2 Веб-приложение «CheckAuto» 11
1.2.3 Веб-приложение «AutoСпец» 12
1.2.4 Сравнение аналогов 12
1.3 Анализ целевой аудитории 13
1.4 Анализ функциональных требований 15
1.5 Анализ нефункциональных требований 20
1.6 Планирование разработки и оценка бюджета 21
2 Проектирование мобильного приложения 32
2.1 Выбор технологий и инструментальных средств разработки 32
2.2 Проектирование архитектуры программного обеспечения 33
2.3 Проектирование интерфейса пользователя 35
2.4 Логическое и концептуальное проектирование базы данных 38
2.5 Проектирование серверной части приложения 41
3 Реализация мобильного приложения 43
3.1 Реализация базы данных 43
3.2 Реализация серверной части приложения 43
3.3 Реализация клиентской части приложения 45
4 Тестирование программного продукта 49
4.1 Тестирование серверной части приложения 49
4.2 Тестирование клиентской части приложения 50
4.2.1 Функциональное тестирование 50
4.2.2 Модульное тестирование 51
4.3 Метрики кода 52
4.3.1 Метрики размера программы 53

4.3.2 Метрики Холстеда 53
Заключение 56
Список использованных источников 58
Приложение А (справочное) Глоссарий 61
Приложение Б (обязательное) Исходный код скриптов базы данных 63
Приложение В (обязательное) Исходный код серверной части приложения. 68
Приложение Г (обязательное) Исходный код мобильного приложения 84
Приложение Д (обязательное) Исходный код функций тестирования 153
Приложение Е (рекомендованное) Результаты тестирования серверной части приложения 156
Приложение Ж (рекомендованное) Результаты тестирования мобильного приложения 159

Введение

В современном мире информационные технологии играют большую роль в жизни всего общества. С каждым годом наблюдается рост цифровизации различных сфер жизни людей, что приводит к автоматизации процессов, снижению издержек, развитию малого и среднего бизнеса, созданию новых рынков.

В Российской Федерации уделено особое внимание развитию информационных технологий. Данный процесс поддерживается государством по ряду причин: от необходимости в постоянном экономическом развитии и улучшения качества жизни граждан до необходимости в обеспечении цифровой безопасности и суверенитета страны. В связи с этим правительством Российской Федерации была утверждена программа «Цифровая экономика Российской Федерации» 9 мая 2017 года. Этот документ определяет стратегию развития « цифровой экономики Российской Федерации, в которой данные в цифровой форме являются ключевым фактором производства во всех сферах социально-экономической деятельности, что повышает конкурентоспособность страны, качество жизни граждан, обеспечивает экономический рост» [2].

На тщательное внимание к цифровизации также указывает Указ Президента Российской Федерации «О стратегии развития информационного общества в Российской Федерации на 2017 2030 годы» от 9 мая 2017 года [3].

В мировой тенденции развития информационных технологий можно особенно выделить распространение мобильных устройств с доступом к интернету. В подтверждение этих слов можно взять информацию из государственной программы «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». В настоящее время мы можем наблюдать как данная тенденция лишь продолжает усиливаться [2].

Также как и информационные технологии, в Российской Федерации стремительно развивается сфера автопарка страны. Согласно данным Росстата, на конец 2023 года в России насчитывалось более 51,5 млн легковых

автомобилей, из которых 94% находятся в собственности граждан. Кроме того, по данным агентства «Автостат», по состоянию на 1 июля 2024 года в стране зарегистрировано 55,06 млн автотранспортных средств. Из этого можно сделать вывод, что увеличение количества автомобилей приводит к росту спроса на качественные услуги автосервисов. Актуальности данному вопросу добавляет тот факт, что средний возраст легковых автомобилей, которые требуют своевременного и постоянного обслуживания, в России составляет почти 15 лет, и этот показатель продолжает расти [3, 4].

Исходя из всего вышеперечисленного следует, что разработка мобильного приложения для поддержки процессов функционирования предприятия автосервиса 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 его пользователям удобный и простой доступ к записи на услуги в автосервисе. С помощью приложения пользователи смогут просматривать необходимую информацию об услуге, а именно вид производимых работ, цену, возможное время записи и другое. Пользователи смогут через свое мобильное устройство удаленно записаться на предлагаемую услугу без необходимости совершения звонка. Необходимо лишь иметь выход в сеть Интернет.

Помимо этого, приложение будет предоставлять возможность посмотреть историю обслуживания автомобиля, что систематизирует необходимую автосервису информацию об автомобиле и поможет пользователю быстро найти список произведенных работ при необходимости. Также пользователь будет получать уведомления о скорой необходимости технического автомобиля.

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

- регистрация новых пользователей;
- вход зарегистрированных пользователей в систему;
- просмотр списка услуг и цен;
- запись на услугу;
- просмотр истории обслуживания;
- получение уведомления;
- оставление отзыва.

1.2 Анализ программных продуктов аналогичного назначения

В настоящее время существует большое множество приложений, позволяющих записываться на услуги в автосервисах. Рассмотрим наиболее известные из них и проведем их сравнительный анализ.

1.2.1 Мобильное приложение «Wilgood Авто»

Wilgood – это сеть автосервисов, предлагающая широкий спектр услуг по ремонту и обслуживанию автомобилей. Клиенты могут записываться онлайн через веб-сайт и мобильное приложение [6].

После авторизации пользователь попадает на главную страницу приложения, с которой есть доступ ко всему функционалу: запись на услуги, просмотр акций, выбор автомобиля. Навигация в приложении устроена с помощью нижней панели, в которой размещены вкладки приложения.

Главная страница мобильного приложения «Wilgood Авто» представлена на рисунке 1.

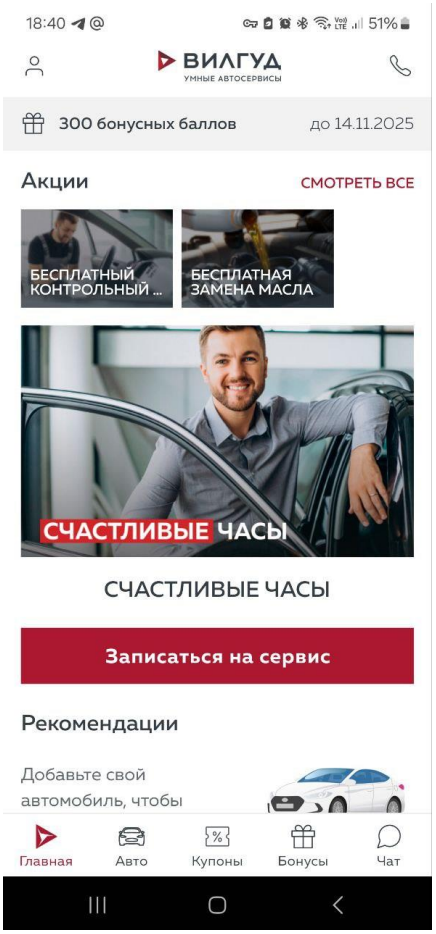

Рисунок 1 – Главная страница мобильного приложения «Wilgood Авто»

К достоинствам приложения можно отнести наличие мобильного приложения, удобную систему записи и возможность выбора конкретного времени, функцию отслеживания статуса ремонта.

К недостаткам мобильного приложения можно отнести задержки с обновлением статуса заказов, нестабильную работу приложения, проблемы с отображением интерфейса.

1.2.2 Веб-приложение «CheckAuto»

CheckAuto – это сервис, предлагающий онлайн-запись на диагностику и ремонт автомобилей в автосервисах города Пенза. Клиенты могут записываться удаленно через веб-сайт, что делает процесс удобным и доступным в любое время дня [7].

При входе на сайт пользователь попадает на страницу с формой для записи на услугу. Навигация происходит с помощью панели в верхней части экрана. Пользователь может посмотреть новости и акции, отзывы, статьи, контакты и другое.

Главная страница приложения «CheckAuto» представлена на рисунке 2.

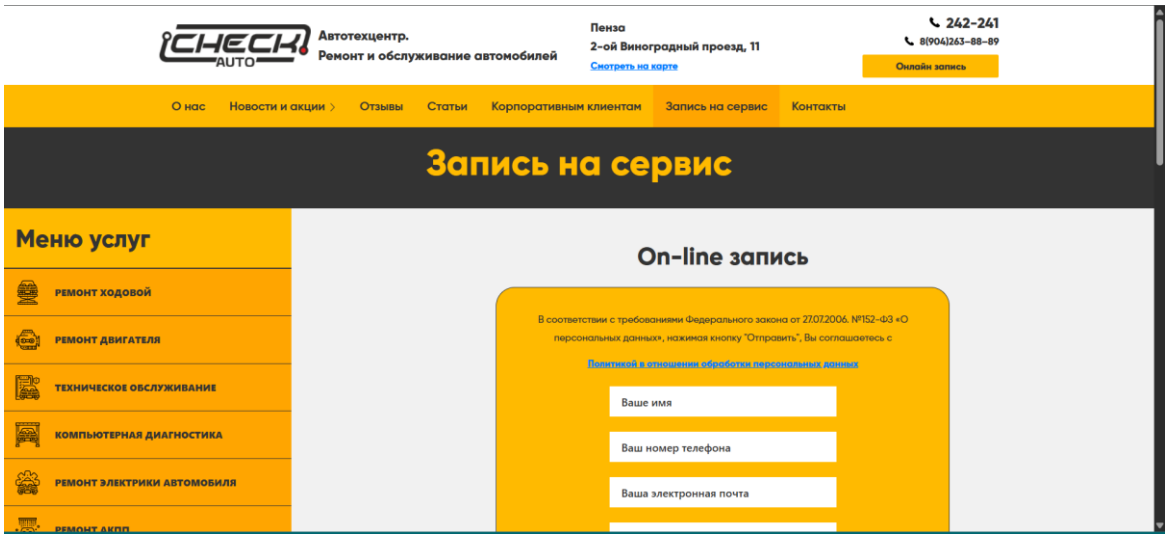

Рисунок 2 – Главная страница веб-приложения «CheckAuto»

К достоинствам приложения можно отнести наличие возможности выбора автосервиса среди нескольких вариантов, отзывы пользователей, которые помогут выбрать надежный сервис, быстрый процесс записи.

К недостаткам мобильного приложения можно отнести отсутствие мобильного приложения, а также устаревший интерфейс.

1.2.3 Веб-приложение «AutoСпец»

AutoСпец – современный многопрофильный автотехцентр, основанный в 2008 году и расположенный в Пензе. Компания предоставляет широкий спектр услуг по ремонту и обслуживанию автомобилей, а также реализует автозапчасти включая диагностику, шиномонтаж и реализует автозапчасти [8].

При входе на сайт пользователь попадает на главную страницу с несколькими возможностями: заказать звонок или же записаться онлайн. Также с помощью навигационной панели, расположенной в верхней части экрана, пользователю предлагаются возможности посмотреть услуги, запчасти, акции, вакансии, отзывы и контакты.

Главная страница веб-приложения «AutoСпец» представлена на рисунке 3.

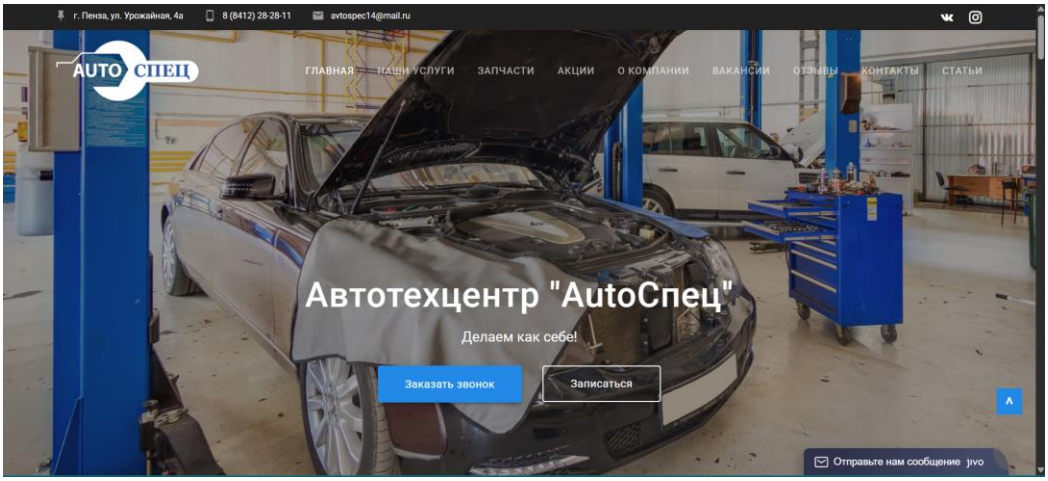

Рисунок 3 – Главная страница веб-приложения «AutoСпец»

К достоинствам приложения можно отнести наличие современного дизайна, возможности записаться онлайн, отзывов пользователей.

К недостаткам мобильного приложения можно отнести отсутствие мобильного приложения для записи и отслеживания истории обслуживания, ограниченная информация о сервисе в онлайн-пространстве.

1.2.4 Сравнение аналогов

На данном этапе был произведен сравнительный анализ рассмотренных ранее приложений. В результате анализа была сформирована таблица 1.

Таблица 1 – Сравнение аналогов

Приложение	Наличие мобильного приложения	Возможность записи на услугу онлайн	Отслеживания статуса ремонта	Удобство интерфейса
Wilgood Авто	Да	Да	Да	Перегруженный
CheckAuto	Нет	Да	Нет	Устаревший
AutoСпец	Нет	Да	Нет	Эргономичный
Разрабатываемое приложение	Да	Да	Да	Эргономичный

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

1.3 Анализ целевой аудитории

Целевой аудиторией проекта являются работники и клиенты автосервисов. Согласно информации, взятой из годовой отчетности ГИБДД, количество водительских удостоверений, выданных в 2024 году, равняется 1,34 млн. А самой востребованной у россиян оказалась категория В (легковые автомобили), которые чаще всего являются автомобилями для частного пользования. Также по информации агентства «Автостат» на 2025 год в Российской Федерации насчитывается 117,5 тыс. автосервисов [5].

На рисунке 4 представлены доли основных производителей смартфонов на мировом рынке за последний год [9].

Продажи смартфонов в 2024 году, млн штук

№	Вендор	Продажи в 2024 г.	Доля рынка в 2024 г.	Продажи в 2023 г.	Доля рынка в 2023 г.	Рост 2023/2024
1	Apple	232,1	18,7%	234,3	20,1%	-0,9%
2	Samsung	223,4	18,0%	226,7	19,5%	-1,4%
3	Xiaomi	168,5	13,6%	146	12,5%	15,4%
4	Transsion	106,9	8,6%	94,9	8,2%	12,7%
5	Oppo	104,8	8,5%	103,4	8,9%	1,4%
	Другие	402,9	32,5%	358,9	30,8%	12,3%
	Всего	**12388**	**100,0%**	**11641**	**100,0%**	**6,4%**

Источник: IDC, 2025

Рисунок 4 – Доли производителей смартфонов на мировом рынке

На графике можно увидеть, что основной долей операционных систем смартфонов на мировом рынке является Android (около 70%). К смартфонам с данной операционной сети относятся продукты следующих компаний: Samsung, Xiaomi, Oppo и другие. Компания Apple и ее операционная система iOS занимает 18,7% мирового рынка. Поэтому с целью увеличения количества пользователей, которые смогут использовать разрабатываемое мобильное приложение, было решено создавать его именно для операционной системы Android.

Также стоит отметить, что в последние годы в России предпринимаются попытки разработки собственных мобильных операционных систем, таких как Aurora OS (разработка компании «Открытая мобильная платформа» при поддержке Ростелекома). Однако доля подобных решений на рынке остаётся крайне незначительной, и они в основном используются в государственных и корпоративных проектах. Ввиду отсутствия массового потребительского спроса на устройства с такими ОС и слабой экосистемы приложений, поддержка российских мобильных платформ в рамках данного проекта на текущем этапе не планируется. При этом при необходимости такая адаптация возможна в будущем за счёт модульной архитектуры приложения.

На рисунке 5 приведена статистика распространения разных версий системы Android [10].

Рисунок 5 – Распространение различных версий системы Android

На диаграмме мы можем наблюдать, что 87,5 % пользователей системы Android использует версии не старее 10.0. Поэтому в системные характеристики приложения будет заложены такие требования к системе, чтобы оно успешно функционировало для Android 10.0 и более новых версий системы.

1.4 Анализ функциональных требований

Приложение может применяться клиентами, мастерами и администраторами автосервиса. Приложение позволяет клиенту просматривать список услуг и цен, записываться на услугу, просматривать историю обслуживания, оставить отзыв. Для мастера приложение предоставляет возможность добавить, удалить или отредактировать свое расписание, посмотреть записанных к нему клиентов и отправить им уведомлении о статусе выполнения работы. Администратору же предоставляется возможность добавить нового мастера, удалить мастера или отредактировать информацию о мастере, также администратор может добавить, удалить или отредактировать список услуг и цен и посмотреть статистику об автосервисе.

Для обеспечения корректной работы системы будет реализована авторизация с распределением ролей: клиент, мастер и администратор. После входа в систему каждому пользователю предоставляется доступ только к тем функциям, которые соответствуют его роли. Интерфейс будет адаптироваться под каждую категорию пользователей, что повышает удобство использования и снижает вероятность ошибок. Все данные, включая расписания, записи, отзывы и статистику, хранятся в единой базе данных, обеспечивающей целостность и актуальность информации. Система разрабатывается с учётом требований надёжности и удобства эксплуатации.

На основе всего вышесказанного была создана диаграмма вариантов использования, представленная на рисунке 6 [17].

Составим спецификации для основных вариантов использования: «Вход в систему» (рисунок 7).

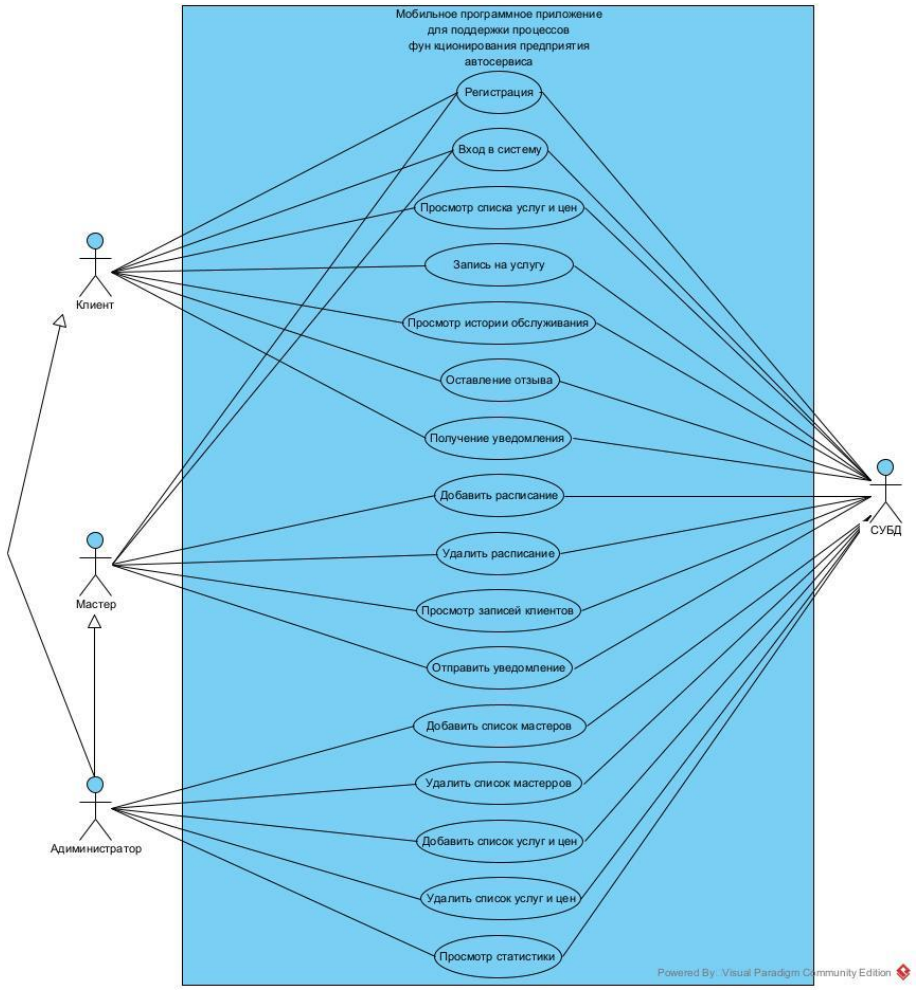

Рисунок 6 – Диаграмма вариантов использования мобильного приложения

Наименование: вход в систему
ID: 1
Краткое описание: пользователь входит в систему
Действующие лица: пользователь, СУБД
Предусловие: –
Основной поток: Пользователь вводит логин Пользователь вводит пароль Система ищет информацию по логину пользователя в БД ЕСЛИ пользователь найден Система проверяет соответствие пароля, введенного пользователем, с паролем в БД ЕСЛИ пароли соответствуют Система разрешает пользователю совершить вход. ИНАЧЕ Система сообщает пользователю, что пароль неправильный ИНАЧЕ Система сообщает пользователю, что пользователь с заданным логином не был найден
Постусловие: пользователь совершил вход в систему

Рисунок 7 – Спецификация варианта использования «Вход в систему»

Составим спецификации для «Просмотр записей клиентов» (рисунок 8), «Запись на услугу» (рисунок 9).

Наименование: просмотр записей клиентов
ID: 2
Краткое описание: мастер просматривает записи клиентов
Действующие лица: мастер, СУБД
Предусловие: мастер вошел в систему
Основной поток: Мастер открывает вкладку Записи клиентов Система делает запрос к БД в поиске записей клиентов ЕСЛИ записи найдены Система отображает найденную информацию мастеру ИНАЧЕ Система сообщает мастеру, что записи не найдены
Постусловие: мастер посмотрел записи клиентов

Рисунок 8 – Спецификация варианта использования «Просмотр записей клиентов»

Наименование: запись на услугу
ID: 3
Краткое описание: клиент выбирает и записывается на услугу
Действующие лица: клиент, СУБД
Предусловие: клиент вошел в систему
Основной поток: Клиент открывает вкладку запись на услугу Система делает запрос к БД в поиске доступных услуг ЕСЛИ записи найдены Система отображает найденную информацию клиенту Клиент выбирает услугу и доступное время, нажимает кнопку продолжить Клиент подтверждает внесенную им информацию о записи Система добавляет запись на услугу в БД ИНАЧЕ Система сообщает клиенту, что услуги не найдены
Постусловие: клиент записан на услугу

Рисунок 9 – Спецификация варианта использования «Запись на услугу»

Для некоторых из рассмотренных выше вариантов использования было принято решение разработать диаграммы последовательности и деятельности для более подробного рассмотрения [17].

Для варианта использования «Запись на услугу» была составлена диаграмма последовательности, представленная на рисунке 10.

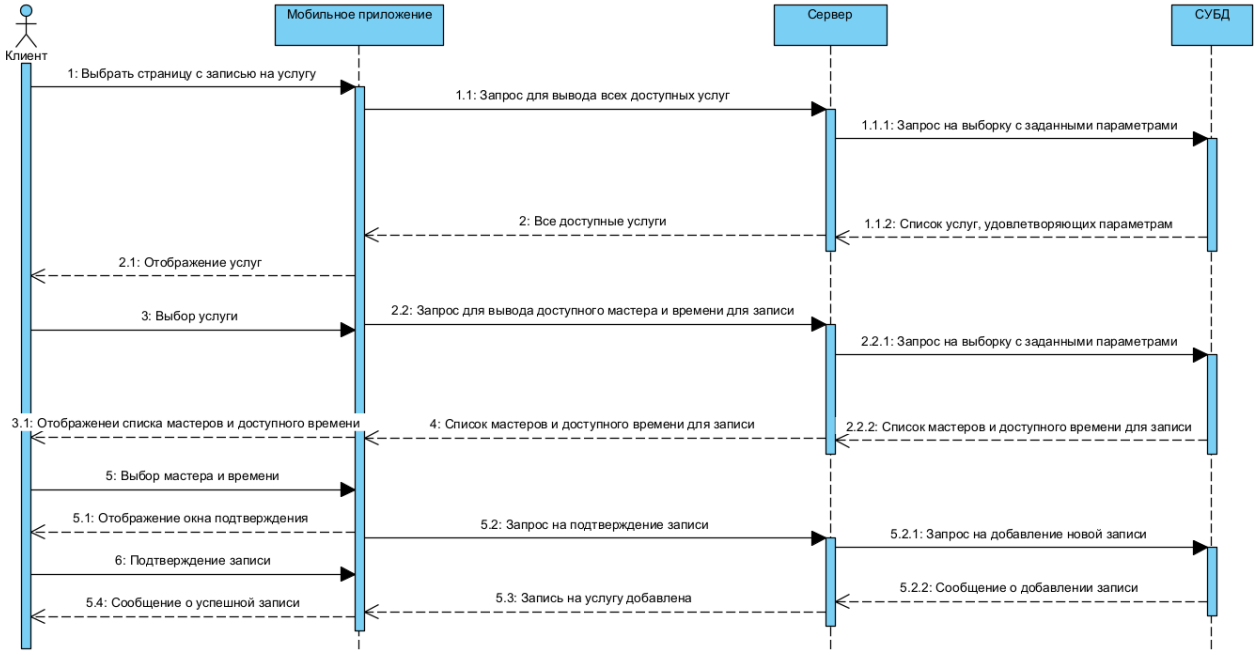

Рисунок 10 – Диаграмма последовательности варианта использования «Запись на услугу»

Для варианта использования «Добавить список мастеров» была составлена диаграмма последовательности, представленная на рисунке 11.

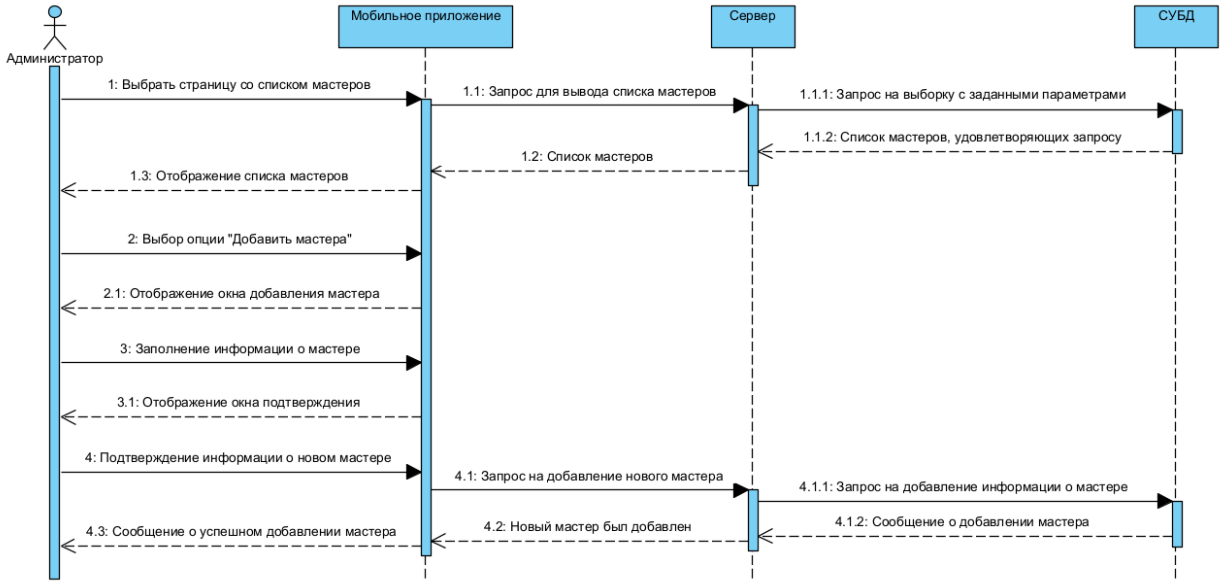

Рисунок 11 – Диаграмма последовательности варианта использования «Добавить список мастеров»

Для варианта использования «Отправить уведомление» была составлена диаграмма деятельности, представленная на рисунке 12.

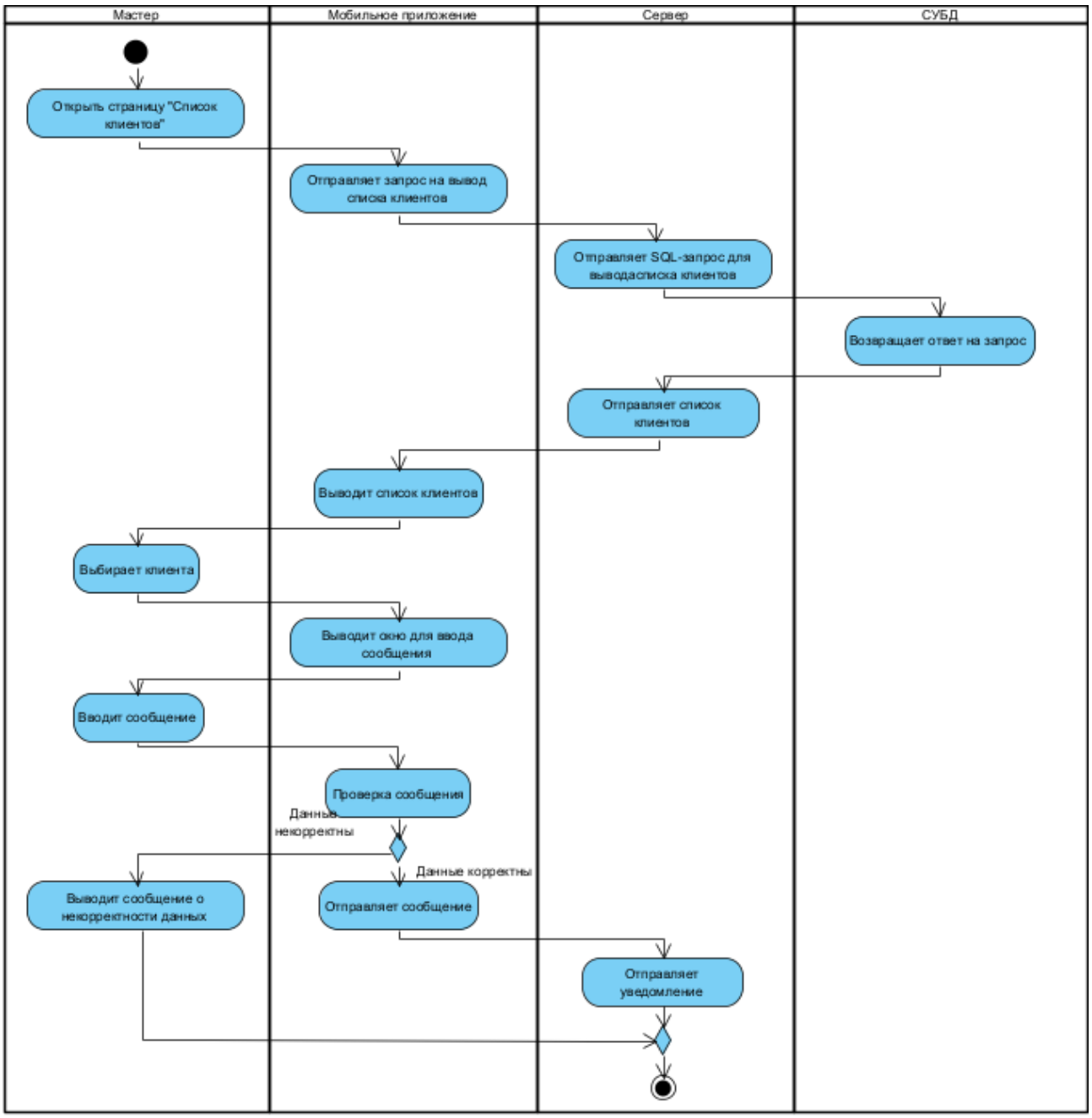

Рисунок 12 – Диаграмма деятельности варианта использования «Отправить уведомление»

Для варианта использования «Вход в систему» была составлена диаграмма деятельности, представленная на рисунке 13. Диаграмма отражает основные этапы процесса авторизации пользователя в системе. С её помощью можно проследить логику проверки введённых данных и обмена ими между различными участниками этого процесса.

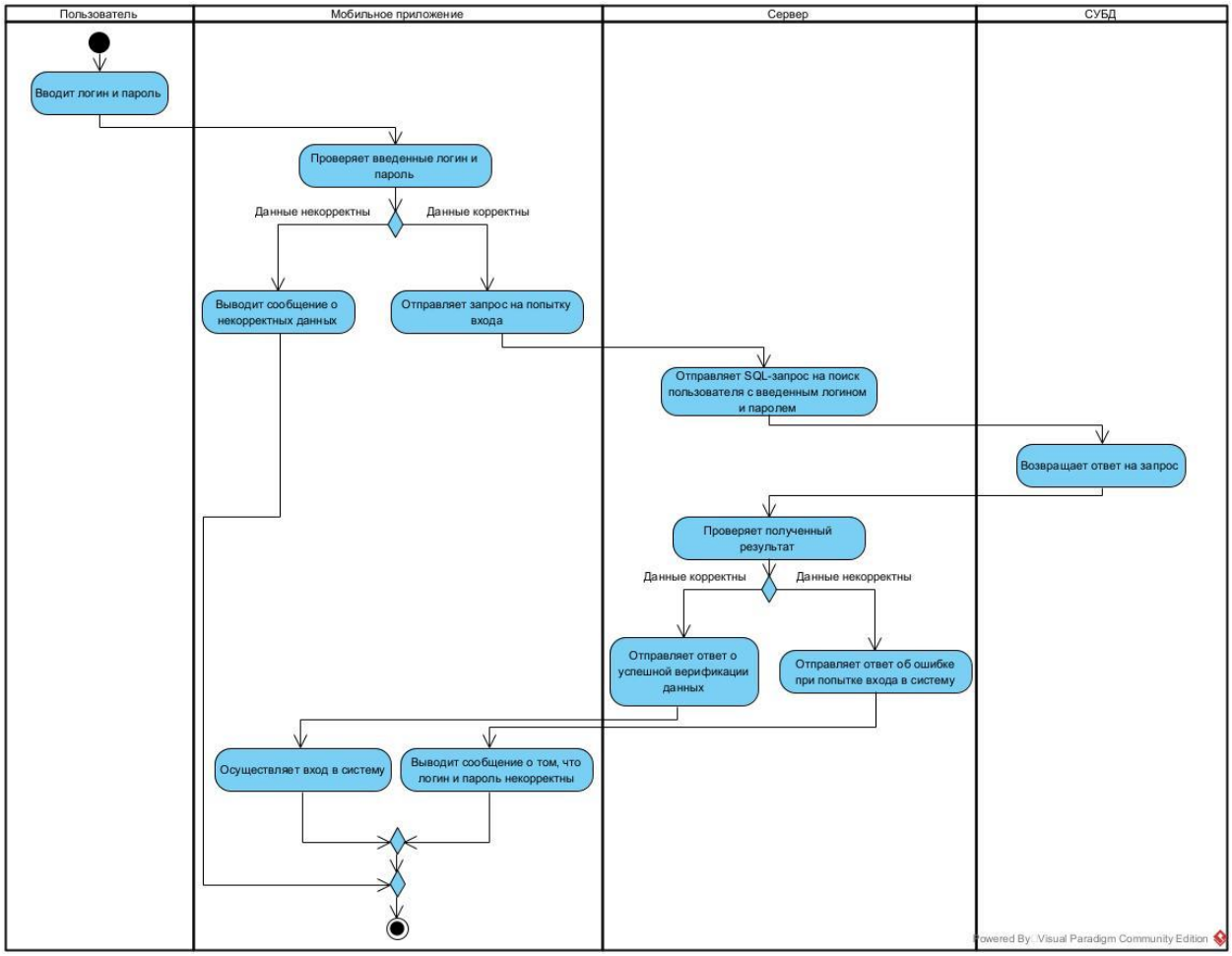

Рисунок 13 – Диаграмма деятельности варианта использования «Вход в систему»

В результате анализа функциональных требований основные функции разрабатываемого приложения были успешно выявлены. Для разрабатываемого приложения были построены диаграммы вариантов использования, деятельности, последовательности и для некоторых выявленных вариантов использования были составлены спецификации.

1.5 Анализ нефункциональных требований

Разработка серверной части приложения будет выполнена на языке программирования C#, в среде программирования Microsoft Visual Studio 2022. Данные на сервере хранятся в базе данных, реализованной с использованием СУБД Microsoft SQL Server, запросы к базе данных осуществляются на языке T-

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 эргономичным и понятным пользователю интерфейсом. Клиентское приложение должно быть отлажено и должно успешно работать под управлением ОС Android 10.0 и выше.

1.6 Планирование разработки и оценка бюджета

Грамотное планирование является необходимым условием для качественного и своевременного выполнения работ по любому проекту, включая проекты в сфере информационных технологий. Оно позволяет выстроить график таким образом, чтобы достичь всех поставленных целей в минимально возможные сроки, с приемлемым качеством и при наименьших затратах.

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
5	Анализ функциональных требований	1 день	3;4
6	Анализ требований к интерфейсу	1 день	5
7	Анализ целевой аудитории	2 дней	6
8	Анализ требований завершён	–	7
9	Выбор технологий разработки ПО	–	–
10	Анализ существующих технологий разработки ПО	4 дней	1
11	Выбор технологий разработки ПО	1 день	10
12	Выбор средств разработки ПО завершён	–	11
13	Проектирование программных средств	–	–

Продолжение таблицы 2

№	Название задачи	Длительность	Предшественники
14	Проектирование архитектуры приложения	5 дней	12;8
15	Проектирование базы данных	3 дней	14
16	Проектирование серверной части приложения	3 дней	15;14
17	Проектирование мобильного приложения	3 дней	14
18	Проектирование интерфейса пользователя	2 дней	14
19	Проектирование завершено	–	16;17;18
20	Реализация программных средств	–	–
21	Реализация базы данных	8 дней	19
22	Реализация серверной части приложения	21 дней	19
23	Реализация мобильного приложения	14 дней	19
24	Реализация завершена	–	22;23;21
25	Тестирование	–	–
26	Функциональное тестирование	4 дней	24
27	Модульное тестирование	2,25 дней	24
28	Тестирование завершено	–	26;27
29	Составление и оформление документации	7 дней	28
30	Конец проекта	–	29

Таким образом, для реализации проекта нужно выполнить 30 задач.

Ниже в таблице 3 представлено описание затрачиваемых ресурсов.

Таблица 3 – Перечень ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Программист 1	Трудовой	500,00 ₽/ч	750,00 ₽/ч
Программист 2	Трудовой	300,00 ₽/ч	500,00 ₽/ч
Программист 3	Трудовой	300,00 ₽/ч	500,00 ₽/ч
Дизайнер	Трудовой	350,00 ₽/ч	550,00 ₽/ч
Интернет	Затраты	2000 ₽/м	
Компьютер	Материальный	25 000,00 ₽	
Мобильное устройство	Материальный	10 000,00 ₽	

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

В таблице 4 приведено распределение ресурсов по задачам.

Таблица 4 – Назначение ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало проекта	–	–
2	Анализ требований	–	–

Продолжение таблицы 4

№	Название задачи	Ресурс	Единицы
3	Анализ предметной области и постановка задачи	Программист 1 Компьютер Интернет Мобильное устройство	50% 4 шт. 2 000 ₽ 1 шт.
4	Анализ программных продуктов аналогичного назначения	Программист 2 Программист 3	50% 50%
5	Анализ функциональных требований	Программист 1 Программист 2	50% 50%
6	Анализ требований к интерфейсу	Программист 1 Дизайнер	25% 100%
7	Анализ целевой аудитории	Программист 1	50%
8	Анализ требований завершён	–	–
9	Выбор технологий разработки ПО	–	–
10	Анализ существующих технологий разработки ПО	Программист 1 Программист 2 Программист 3	25% 25% 25%
11	Выбор технологий разработки ПО	Программист 1 Программист 2 Программист 3	50% 25% 25%
12	Выбор средств разработки ПО завершён	–	–
13	Проектирование программных средств	–	–
14	Проектирование архитектуры приложения	Программист 1 Программист 2 Программист 3	50% 25% 25%
15	Проектирование базы данных	Программист 1 Программист 2	50% 100%
16	Проектирование серверной части приложения	Программист 1 Программист 2	50% 100%
17	Проектирование мобильного приложения	Программист 1 Программист 3	50% 100%
18	Проектирование интерфейса пользователя	Дизайнер	100%
19	Проектирование завершено	–	–
20	Реализация программных средств	–	–
21	Реализация базы данных	Программист 1 Программист 2 Интернет	50% 100% 2 000 ₽
22	Реализация серверной части приложения	Программист 1 Программист 2	50% 100%
23	Реализация мобильного приложения	Программист 1 Программист 3 Дизайнер	50% 100% 100%

Окончание таблицы 4

№	Название задачи	Ресурс	Единицы
24	Реализация завершена	–	–
25	Тестирование	–	–
26	Функциональное тестирование	Программист 1 Программист 3 Программист 2 Интернет	25% 50% 25% 2 000 ₽
27	Модульное тестирование	Программист 1 Программист 3 Программист 2	25% 50% 25%
28	Тестирование завершено	–	–
29	Составление и оформление документации	Программист 1 Программист 2 Программист 3	100% 50% 50%
30	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 построен сетевой график (рисунок 14).

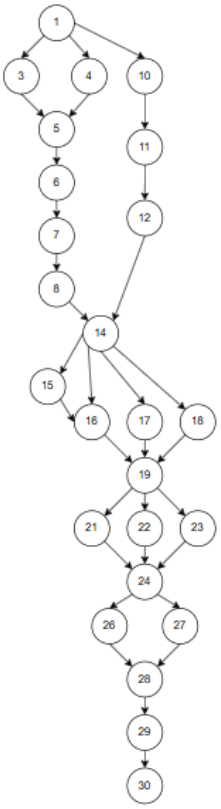

Рисунок 14 – Сетевой график проекта

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

Таблица 5 – Расчёты раннего времени начала работ

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
1	–	–	0	–	0
2	1	0	2	0 + 2	2
3	1	0	4	0 + 4	4
4	2,3	2,4	2	max(2+2;4+2) = max(4;6)	6
5	4	6	2	6 + 2	8
6	5	8	2	8 + 2	10
7	6	10	0	10 + 0	10
8	1	0	4	0 + 4	4
9	8	4	1	4 + 1	5
10	9	5	0	5 + 0	5
11	7,1	10	5	max(10+5;0+5) = max(15;5)	15
12	11	15	3	15 + 3	18
13	11,12	15,18	3	max(15,18) + 3 = 21	21
14	11	15	3	15 + 3	18
15	11	15	2	15 + 2	17
16	13,14,15	21,18,17	0	max(21,18,17) + 0 = 21	21
17	16	21	8	21 + 8	29
18	16	21	21	21 + 21	42
19	16	21	14	21 + 14	35
20	17,18,19	29,42,35	0	max(29,42,35) + 0 = 42	42
21	20	42	4	42 + 4	46
22	20	42	3	42 + 3	45
23	21,22	46,45	0	max(46,45) + 0 = 46	46
24	23	46	7	46 + 7	53
25	24	53	0	53 + 0	53

В таблице 6 рассчитано позднее время начала работ задач.

Таблица 6 – Расчеты позднего времени начала работ

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
25	–	51	0	51 − 0	51
24	25	51	7	51 − 7	44
23	24	44	0	44 − 0	44
21	23	44	4	44 − 4	40
22	23	44	3	44 − 3	41
20	21,22	40,41	0	min(40, 41) − 0	40
17	20	40	8	40 − 8	32
18	20	40	21	40 − 21	19
19	20	40	14	40 − 14	26
16	17,18,19	32,19,26	0	min(32,19,26) − 0	19
13	16	19	3	19 − 3	16
14	16	19	3	19 − 3	16
15	16	19	2	19 − 2	17
11	13,14,15	16,16,17	5	min(16,16,17)−5	11
10	11	11	0	11 − 0	11
7	11	11	5	11 − 5	6
6	7	6	2	6 − 2	4
5	6	4	1	4 − 1	3
4	5	3	1	3 − 1	2
2	4	2	2	2 − 2	0
3	4	2	4	1	0
9	10	11	1	11 − 1	10
8	9	10	4	10 − 4	6

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 проекта

Работа №	Раннее время начала	Позднее время начала	Резерв времени
1	0	0	0
2	2	2	0
3	4	4	0
4	5	5	0
5	6	6	0
6	8	8	0
7	11	11	0

Продолжение таблицы 7

Работа №	Раннее время начала	Позднее время начала	Резерв времени
8	11	16	5
9	13	18	5
10	16	21	5
11	19	24	5
12	22	27	5
13	25	30	5
14	19	19	0
15	25	27	2
16	31	31	0
17	37	42	5
18	40	40	0
19	33	38	5
20	40	40	0
21	44	44	0
22	43	43	0
23	44	44	0
24	51	59	8
25	66	66	0

На основе полученных расчётов можно сделать вывод, что критический путь проекта проходит через работы с номерами 1, 2, 3, 4, 5, 6, 7, 11, 14, 16, 18, 20, 21, 23, 25, а общее время выполнения проекта составляет 66 дней.

Работы, входящие в критический путь, не имеют временного резерва, и любое их смещение приведёт к увеличению продолжительности всего проекта.

В процессе анализа также были определены зависимости между задачами, позволяющие выявить потенциальные узкие места. Это особенно важно на этапах планирования и распределения ресурсов между участниками проекта. Поэтому данные задачи требуют особого внимания при контроле выполнения проекта.

Остальные работы, не входящие в критический путь, обладают временным резервом и могут быть смещены без влияния на сроки завершения проекта.

На рисунке 15 представлен граф проекта, на котором штриховыми линиями указан критический путь.

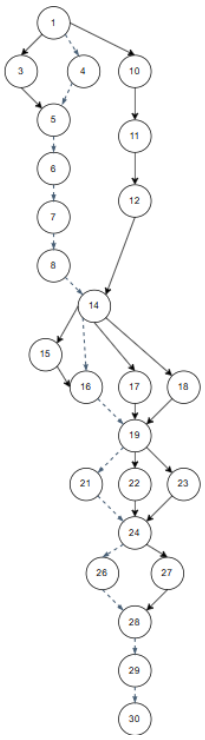

Рисунок 15 – Критический путь проекта

Для того, чтобы проверить корректность расчётов продолжительности проекта, а также вычислить величину его бюджета, воспользуемся системой управления проектами Microsoft Project.

Список задач из таблицы 3, внесённый в систему Microsoft Project, представлен на рисунке 16. В качестве времени начала проекта была принята дата 26.02.25.

	Название задачи	Длительность	Начало	Окончание	Предшествен
1	Начало проекта	0 дней	Ср 26.02.25	Ср 26.02.25	
2	**Анализ требований**	**10 дней**	**Ср 26.02.25**	**Вт 11.03.25**	
3	Анализ предметной области и постановка задачи	2 дней	Ср 26.02.25	Чт 27.02.25	1
4	Анализ программных продуктов аналогичного наз	4 дней	Ср 26.02.25	Пн 03.03.25	1
5	Анализ функциональных требований	2 дней	Вт 04.03.25	Ср 05.03.25	3;4
6	Анализ требований к интерфейсу	2 дней	Чт 06.03.25	Пт 07.03.25	5
7	Анализ целевой аудитории	2 дней	Пн 10.03.25	Вт 11.03.25	6
8	Анализ требований завершён	0 дней	Вт 11.03.25	Вт 11.03.25	7
9	**Выбор технологий разработки ПО**	**6 дней**	**Ср 26.02.25**	**Ср 05.03.25**	
10	Анализ существующих технологий разработки ПО	4 дней	Ср 26.02.25	Пн 03.03.25	1
11	Выбор технологий разработки ПО	2 дней	Вт 04.03.25	Ср 05.03.25	10
12	Выбор средств разработки ПО завершён	0 дней	Ср 05.03.25	Ср 05.03.25	11
13	**Проектирование программных средств**	**14 дней**	**Ср 12.03.25**	**Пн 31.03.25**	
14	Проектирование архитектуры приложения	6 дней	Ср 12.03.25	Ср 19.03.25	12;8
15	Проектирование базы данных	4 дней	Чт 20.03.25	Вт 25.03.25	14
16	Проектирование серверной части приложения	4 дней	Ср 26.03.25	Пн 31.03.25	15;14
17	Проектирование мобильного приложения	4 дней	Чт 20.03.25	Вт 25.03.25	14
18	Проектирование интерфейса пользователя	2 дней	Чт 20.03.25	Пт 21.03.25	14
19	Проектирование завершено	0 дней	Пн 31.03.25	Пн 31.03.25	16;17;18
20	**Реализация программных средств**	**30 дней**	**Вт 01.04.25**	**Пн 12.05.25**	
21	Реализация базы данных	8 дней	Чт 01.05.25	Пн 12.05.25	19
22	Реализация серверной части приложения	15 дней	Вт 01.04.25	Пн 21.04.25	19
23	Реализация мобильного приложения	15 дней	Вт 01.04.25	Пн 21.04.25	19
24	Реализация завершена	0 дней	Пн 12.05.25	Пн 12.05.25	22;23;21
25	**Тестирование**	**5 дней**	**Вт 13.05.25**	**Пн 19.05.25**	
26	Функциональное тестирование	5 дней	Вт 13.05.25	Пн 19.05.25	24
27	Модульное тестирование	2,25 дней	Вт 13.05.25	Чт 15.05.25	24
28	Тестирование завершено	0 дней	Пн 19.05.25	Пн 19.05.25	26;27
29	Составление и оформление документации	7 дней	Вт 20.05.25	Ср 28.05.25	28
30	Конец проекта	0 дней	Ср 28.05.25	Ср 28.05.25	29

Рисунок 16 – Список задач проекта в системе 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, 2, 3, 4, 5, 6, 7, 11, 14, 16, 18, 20, 21, 23, 25.

Можно заметить, что вычисленный программой критический путь совпал с критическим путём, вычисленным вручную, а общее время выполнения проекта составило 66 дней.

Диаграмма Ганта представлена на рисунке 17.

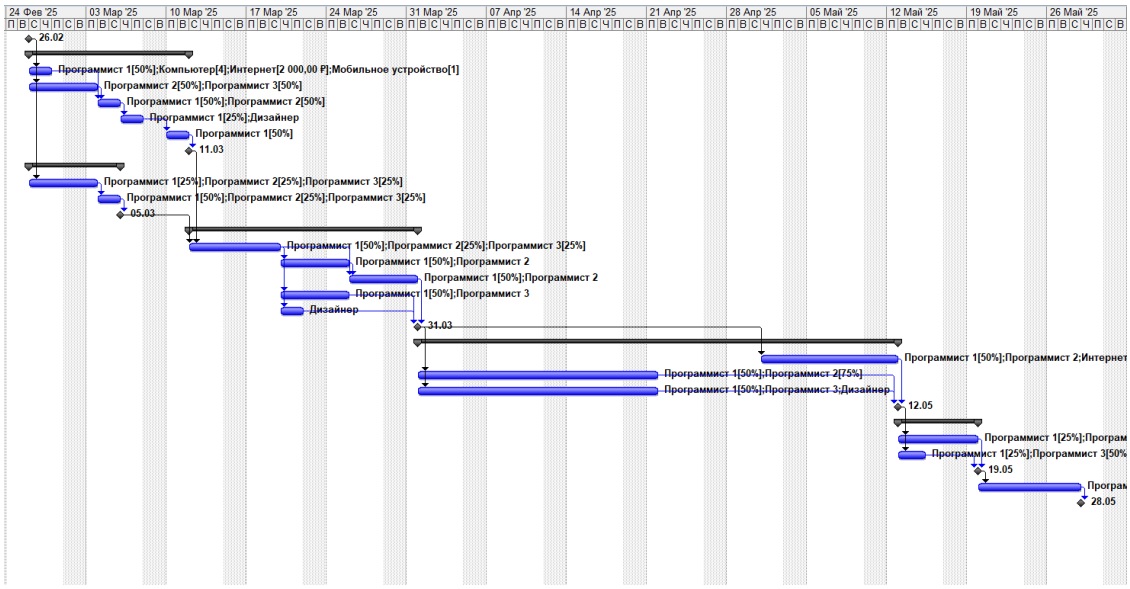

Рисунок 17 – Диаграмма Ганта

На рисунке 18 представлен график трудового ресурса «Программист 1».

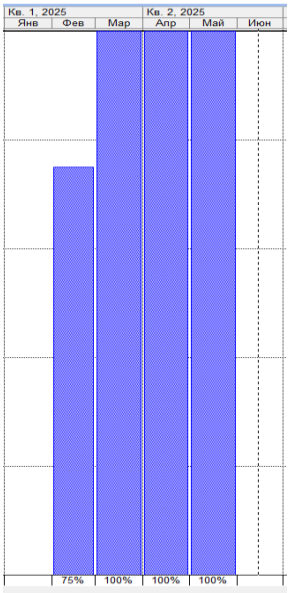

Рисунок 18 – График ресурсов для ресурса «Программист 1»

На рисунках 19, 20 представлены графики трудовых ресурсов «Программист 2», «Программист 3».

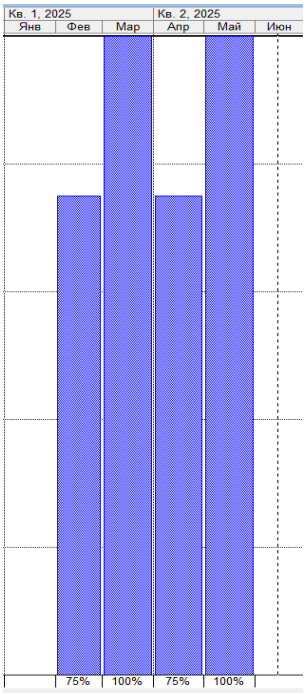

Рисунок 19 – График ресурсов для ресурса «Программист 2»

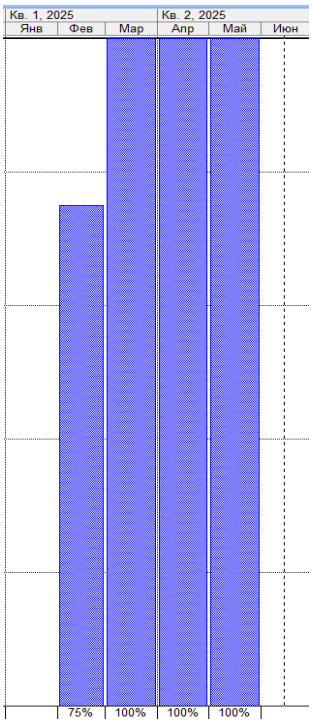

Рисунок 20 – График ресурсов для ресурса «Программист 3»

На рисунке 21 представлен график трудового ресурса «Дизайнер».

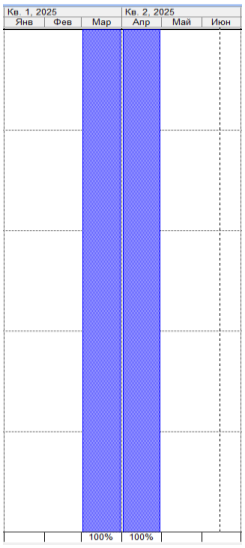

Рисунок 21 – График ресурсов для ресурса «Дизайнер»

На рисунке 22 представлен обзор проекта. Из рисунка видно, что общая длительность проекта составляет 66 дней (что совпадает со значением, вычисленным вручную), а предполагаемые затраты – 499 700 руб.

Проект2.mpp

на дату Вс 15.06.25

Даты			
Начало:	Ср 26.02.25	Окончание:	Ср 28.05.25
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней

Длительность			
Текущий план:	66 дней	Оставшаяся:	66 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	66 дней	Процент завершени:	0%

Трудозатраты			
Текущий план:	1 028 ч	Оставшиеся:	1 028 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	1 028 ч	Процент завершени:	0%

Затраты			
Текущий план:	499 700,00 ₽	Оставшиеся:	499 700,00 ₽
Базовый план:	0,00 ₽	Фактические:	0,00 ₽
Отклонение:	499 700,00 ₽		

Состояние задачи	
Еще не начатые задачи	30
Выполняющиеся задачи	0
Завершенные задачи:	0
Итого задач:	30

Состояние ресурса	
Трудовые ресурсы:	5
Трудовые ресурсы с прев. дос	0
Материальные ресурсы:	2
Итого ресурсов:	7

Рисунок 22 – 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 программного обеспечения было принято решение использовать методологию Rational Unified Process (RUP) – одну из наиболее известных и широко применяемых итеративных моделей жизненного цикла программных систем. Эта методология предлагает четко структурированный подход к созданию программного обеспечения, включая последовательные фазы, оценку рисков и промежуточные итерации, позволяющие гибко управлять изменениями.

Разработка согласно RUP разбивается на четыре ключевые фазы, каждая из которых может содержать одну или несколько итераций продолжительностью от двух до шести недель:

– фаза начала проекта – формируются цели, границы проекта, выявляются начальные требования и составляется экономическое обоснование;

– фаза уточнения – проводится углублённый анализ предметной области и уточнение требований, проектируется архитектура;

– фаза построения – реализуется основная функциональность программного продукта;

– фаза внедрения – создаётся финальная версия системы, проводится тестирование, устраняются выявленные ошибки и осуществляется передача продукта заказчику, включая обучение пользователей.

Каждая фаза завершается контрольной вехой, на которой оцениваются достигнутые результаты и принимаются ключевые решения о дальнейшем ходе разработки. Такой подход позволяет своевременно выявлять и минимизировать риски, а при необходимости – вносить изменения в проект ещё на ранних стадиях.

Методология RUP была выбрана благодаря ряду её преимуществ: во-первых, она предоставляет гибкость в управлении разработкой и позволяет оперативно реагировать на изменения требований; во-вторых, она способствует

раннему выявлению потенциальных проблем; в-третьих, в отличие от таких методологий, как XP (eXtreme Programming), RUP не требует постоянного участия заказчика в процессе, что является значимым фактором при ограниченной доступности клиента.

Для анализа требований и построения модели системы использовался язык UML (Unified Modeling Language). Визуализация диаграмм и архитектурных решений выполнялась с помощью Visual Paradigm, что позволило эффективно организовать процесс моделирования и структурировать проектную документацию.

Для планирования и управления проектом применялось приложение Microsoft Project, которое дало возможность сформировать иерархическую структуру работ (WBS), составить календарный план и провести оценку ресурсов – как трудозатрат, так и материальных и денежных.

Клиентская часть приложения разрабатывается с использованием платформы .NET MAUI и языка программирования C# в среде Microsoft Visual Studio 2022, что обеспечивает кроссплатформенность и высокую производительность. Серверная часть реализуется также на языке C# с использованием ASP.NET, а для хранения данных применяется СУБД Microsoft SQL Server, что обеспечивает надёжность и масштабируемость системы. Работа с базой данных организована с помощью Entity Framework Core, позволяющего выполнять как SQL-запросы, так и использовать объектно-ориентированное представление данных через LINQ.

Таким образом, совокупность используемых инструментов, подходов и технологий обеспечивает гибкую, надёжную и масштабируемую разработку программного обеспечения в соответствии с современными требованиями к качеству и архитектуре ИТ-систем.

2.2 Проектирование архитектуры программного обеспечения

Было принято решение использовать в приложении двухуровневую архитектуру «клиент-сервер». Её схема представлена на рисунке 23.

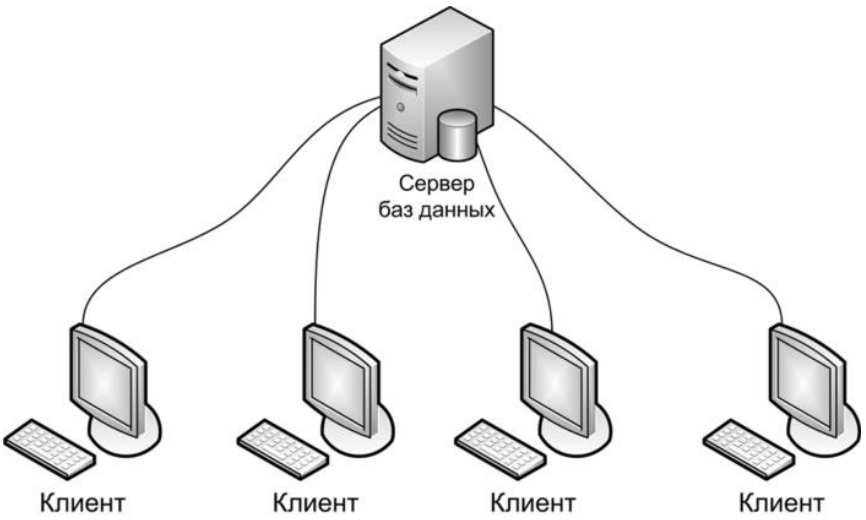

Рисунок 23 – Двухуровневая архитектура «клиент-сервер»

В данной программной архитектуре все ключевые компоненты – база данных, система управления базами данных, серверная логика приложения – размещаются на стороне сервера. Серверная часть отвечает за обработку запросов, выполнение различных операций с данными и реализуется в виде программного интерфейса API, доступного по сети.

Клиентская часть предназначена для взаимодействия пользователя с системой. Она обеспечивает визуальный и удобный интерфейс, через который пользователь может выполнять запросы к серверу. В рамках текущего проекта в качестве клиента выступает мобильное приложение на платформе Android.

Взаимодействие между клиентом и сервером реализовано по следующему принципу: мобильное приложение формирует и отправляет HTTP-запрос на сервер. Сервер, получив запрос, интерпретирует его и формирует соответствующий SQL-запрос к базе данных. После обработки запроса СУБД возвращает результат серверу, который, в свою очередь, подготавливает ответ и пересылает его клиенту. Клиентское приложение отображает полученные данные в понятном для пользователя виде.

Ключевое преимущество такой архитектуры – централизованное хранение данных на сервере. Это позволяет не только снизить нагрузку на клиентское устройство и уменьшить размер мобильного приложения, но и упрощает обеспечение безопасности и резервного копирования. Более того, разделение

клиентской и серверной логики открывает возможность разрабатывать и поддерживать сразу несколько клиентских приложений (например, под различные операционные системы), не внося изменений в серверную часть.

2.3 Проектирование интерфейса пользователя

На основе выявленных функциональных требований был спроектирован интерфейс мобильного приложения. Он включает в себя страницы входа, регистрации, главную страницу приложения, записи на услугу, просмотр истории обслуживания, создания расписания и др.

Страница входа представлена на рисунке 24.

Рисунок 24 – Страница входа

Рассмотрим элементы интерфейса: 1 – поле логина, 2 – поле пароля, 3 – кнопка «Войти», 4 – кнопка «Зарегистрироваться». При нажатии на кнопку «Войти» 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 25.

Рисунок 25 – Страница регистрации

Рассмотрим элементы интерфейса: 1 – поле ФИО, 2 – поле email, 3 – поле номера телефона, 4 – поле пароля, 5 – поле повтора пароля. При нажатии кнопки «Регистрация» система проверяет, являются ли введённые ФИО, email, пароль и номер телефона допустимыми. Если это условие не выполняется, будет выведено сообщение о некорректно заполненных полях. В противном случае будет проверено, не занят ли данный email. Если он занят, выводится соответствующее сообщение. Иначе авторизация будет считаться выполненной успешно.

Главная страница клиента представлена на рисунке 26.

Рисунок 26 – Главная страница клиента

Рассмотрим элементы интерфейса: 1 – кнопка «Записаться», 2 – кнопка «Посмотреть историю», 3. 4. 5 – элементы навигационной панели. На данной

странице отображаются ключевые функции, доступные пользователю: просмотр истории обслуживания, запись на услугу, доступ к профилю. Переход осуществляется по нажатию соответствующих элементов интерфейса.

Страница записи на услугу представлена на рисунке 27.

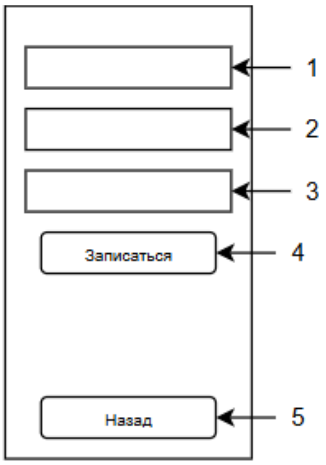

Рисунок 27 – Страница записи на услугу

Рассмотрим элементы интерфейса: 1 – выбор автомобиля, 2 – выбор услуги, 3 – выбор даты и времени. При выборе услуги, автомобиля, даты и времени, а затем нажатии кнопки «Записаться» система выполняет проверку доступности указанного времени. При успешной проверке создаётся новая запись. В случае, если время недоступно, пользователю выводится предупреждение.

Страница заполнения расписания представлена на рисунке 28.

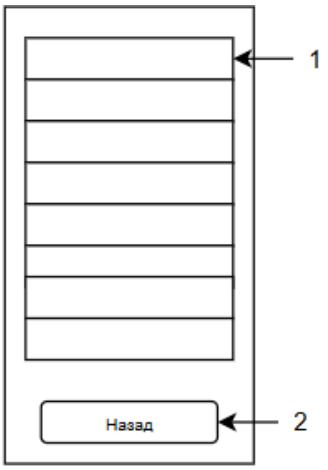

Рисунок 28 – Страница заполнения расписания

Рассмотрим элементы интерфейса: 1 – поля временных слотов расписания. На данной странице мастер указывает доступные временные слоты для записи клиентов. Ввод осуществляется путём выбора даты и времени начала и окончания рабочего интервала. После нажатия на слот времени введённая информация сохраняется в базе данных и становится доступной для отображения при бронировании услуги клиентом.

2.4 Логическое и концептуальное проектирование базы данных

На основе анализа предметной области и функциональных требований была построена концептуальная модель базы данных [25]. Она представлена в виде диаграммы «сущность-связь» (рисунок 29).

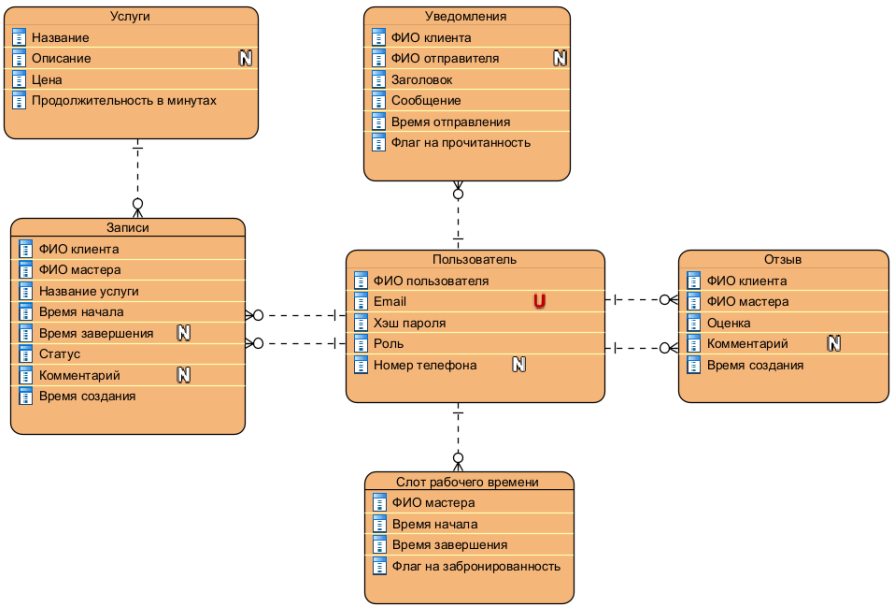

Рисунок 29 – Диаграмма «сущность-связь»

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 клиентов автосервиса	Стержневая	500	50% в год
2	Мастер	Список мастеров автосервиса	Стержневая	30	10% в год

Продолжение таблицы 8

№ п/п	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
3	Администратор	Список администраторов автосервиса	Стержневая	2	0% в год
5	Услуга	Описание оказываемых услуг	Стержневая	50	10% в полгода
6	Запись на обслуживание	Предварительная запись клиента	Ассоциативная	300	100% в месяц

В таблице 9 представлено описание связей между сущностями предметной области.

Таблица 9 – Описание связей между сущностями предметной области

№ п/п	Сущность 1	Сущ-ность 2	Семантика связи	Мощ-ность	Обязательность	Ограничения
1	Пользователь	Роль	Пользователь имеет одну из ролей: Клиент, Мастер, Админ	1:1	Обязательно для пользователя – роль должна быть указана	Значением может быть только:Adminили Master илиClient
2	Клиент	Запись	Клиент записывается на услугу	1:M	Обязательно для записи – у каждой записи должен быть клиент	Клиент должен существовать
3	Мастер	Запись	Мастер выполняет услугу по записи	1:M	Обязательно для записи – у каждой записи должен быть мастер	Мастер должен существовать
4	Запись	Услуга	Каждая запись относится к конкретной услуге	1:1	Обязательно для записи – услуга должна быть указана	Услуга должна существовать

Продолжение таблицы 9

№ п/п	Сущность 1	Сущность 2	Семантика связи	Мощ-ность	Обязательность	Ограничения
5	Мастер	Рабочее время	У каждого мастера есть слоты рабочего времени	1:M	Обязательно для рабочего слота – слот назначается мастеру	Мастер должен существовать
6	Клиент	Уведомление	Клиент получает уведомления	1:M	Обязательно для уведомления – должно быть получателем	Клиент должен существовать
7	Пользователь	Уведомление	Пользователь может быть отправителем уведомления	1:M	Необязательно – уведомление может быть системным	
8	Клиент	Отзыв	Клиент оставляет отзыв	1:M	Обязательно для отзыва – должен быть клиент	Клиент должен существовать
9	Мастер	Отзыв	Мастер получает отзывы	1:M	Обязательно для отзыва – должен быть мастер	Мастер должен существовать
10	Отзыв	Оценка	Отзыв содержит рейтинг	1:1	Обязательно для отзыва	Значение может быть толькоот 1 до 5
11	Запись	Статус	Запись имеет статус выполнения	1:1	Обязательно для записи	Значение может быть тольо: Scheduled или InProgress или Completed или Cancelled
12	Услуга	Цена и длительность	Каждая услуга имеет фиксированную цену и длительность	1:1	Обязательно для услуги	

Окончание таблицы 9

№ п/п	Сущность 1	Сущность 2	Семантика связи	Мощ-ность	Обязательность	Ограничения
13	Рабочий слот	Состояние	Рабочий слот может быть занят или свободен	1:1	Обязательно для рабочего слота	

Её структура описана диаграммой в нотации IDEF1X, представленной на рисунке 30.

Рисунок 30 – Логическая модель базы данных

Представленная логическая модель базы данных отображена уже более подробно. Здесь не только присутствуют типы сущностей, их связи, но и хранится информация о атрибутах сущностей. Однако на данный момент сущности представлены в абстрактной форме, без привязки к конкретной реализации базы данных [25].

2.5 Проектирование серверной части приложения

Серверная часть разрабатываемого приложения реализована в виде REST API, обеспечивающего взаимодействие с клиентом по протоколу HTTP. При этом используются стандартные HTTP-методы: GET – для получения

информации о ресурсе, POST – для создания новых ресурсов, PUT – для их обновления и DELETE – для удаления. Обращение к данным осуществляется через уникальные идентификаторы, представляющие URI соответствующих ресурсов.

В качестве архитектурного подхода применяется модель MVC (Model-View-Controller), получившая широкое распространение при разработке web-приложений. Структура данной архитектуры представлена на рисунке 31 [14].

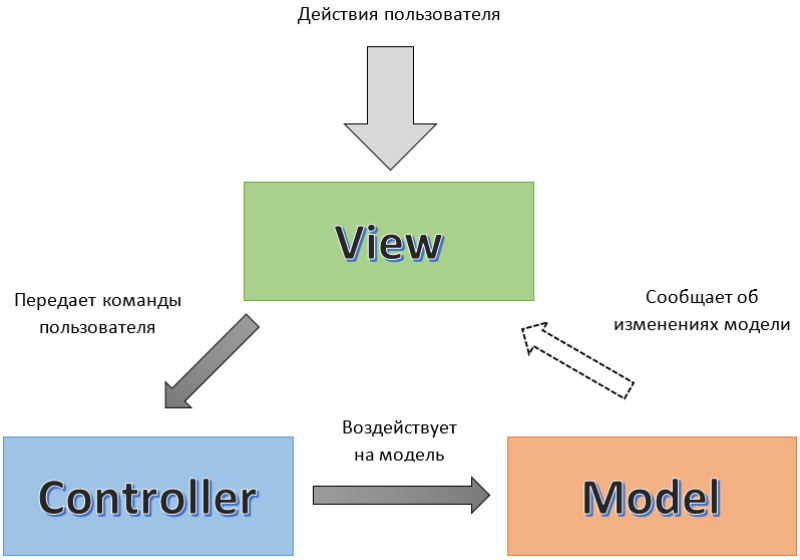

Рисунок 31 – Архитектура MVC

В серверной части приложения реализованы классы модели, отвечающие за работу с базой данных, включая выполнение операций чтения и записи. Контроллеры, в свою очередь, обеспечивают обработку входящих HTTP-запросов, управление логикой взаимодействия и формирование соответствующих ответов. Клиентская часть приложения использует только слой представления, отвечающий за визуальное отображение информации и взаимодействие с пользователем.

3 Реализация мобильного приложения

3.1 Реализация базы данных

На основе составленной ранее логической модели БД с учётом специфики СУБД Microsoft SQL Server при помощи программы Visual Paradigm была построена физическая модель [25]. Её структура представлена на рисунке 32.

Рисунок 32 – Физическая модель базы данных

На основе составленной физической модели БД был разработан скрипт на языке SQL для создания базы данных. Его код представлен в приложении Б. Развёртывание спроектированной базы данных было выполнено путём запуска данного скрипта с использованием утилиты SQL Server Management Studio, используемой для администрирования баз данных под управлением СУБД Microsoft SQL Server.

3.2 Реализация серверной части приложения

Разработка серверной части приложения производилась в среде программирования Microsoft Visual Studio 2022 на языке C#. Для взаимодействия

с базой данных использовалась технология Entity Framework [12]. Диаграмма классов серверной части представлена на рисунке 33.

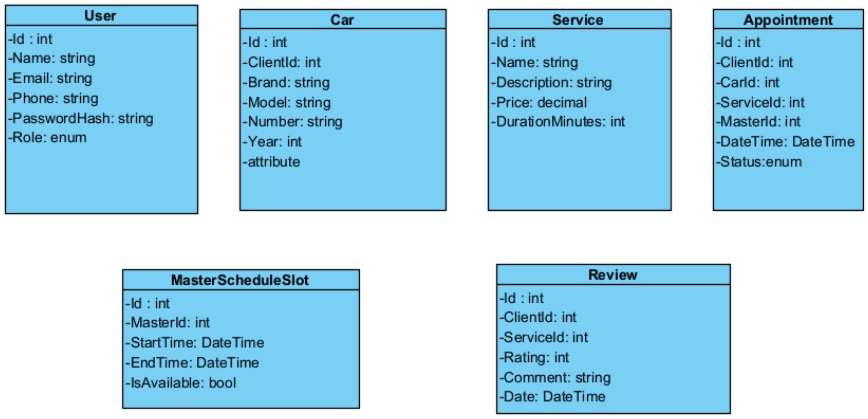

Рисунок 33 – Диаграмма классов серверной части (слой модели)

Как можно заметить, классы User, Car, Service, Appointment, Review, MasterScheduleSlot были созданы на основании одноимённых таблиц базы данных и используются для хранения и обработки основной информации о пользователях, транспортных средствах, предоставляемых услугах, записях на обслуживание, отзывах клиентов и расписании мастеров.

Дополнительно были реализованы вспомогательные классы-представления AppointmentDetailsDto, UserProfileDto, MasterScheduleDto, ServiceDto, которые не соответствуют напрямую таблицам базы данных, а формируются из нескольких связанных сущностей. Эти классы используются для удобства отображения и передачи данных между серверной и клиентской частью, сокращая количество запросов к базе и обеспечивая нужный уровень агрегации данных.

Таким образом, структура модели данных охватывает как базовые сущности предметной области, так и оптимизированные формы представления данных, необходимые для корректной работы интерфейса и обеспечения производительности.

Диаграмма классов слоя контроллеров представлена на рисунке 34.

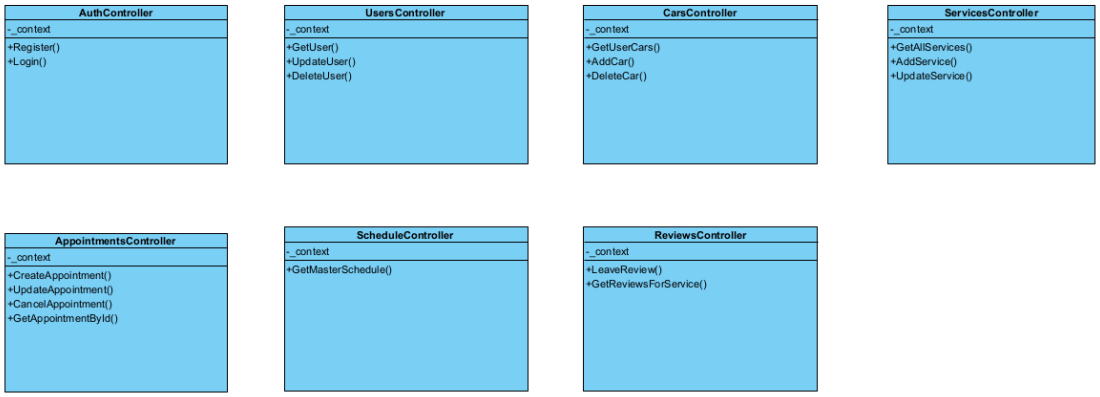

Рисунок 34 – Диаграмма классов серверной части (слой контроллеров)

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

Таблица 10 – Описание классов слоя контроллеров

Класс	Назначение
AuthController	Обработка запросов регистрации, входа, выхода и смены пароля пользователей
UsersController	Получение информации о пользователях, редактирование профиля, удаление
CarsController	Добавление, удаление и редактирование автомобилей клиента
ServicesController	Получение и редактирование списка доступных услуг
AppointmentsController	Создание, отмена и просмотр записей на обслуживание
ScheduleController	Получение доступного расписания работы мастеров
ReviewsController	Добавление и получение отзывов клиентов

Листинг серверной части представлен в приложении В.

3.3 Реализация клиентской части приложения

Разработка мобильного приложения производилась на языке C# с использованием среды программирования Visual Studio 2022. Диаграмма компонентов, отражающая структуру программы, представлена на рисунке 35.

Приложение реализовано с использованием архитектурного шаблона MVVM, что обеспечивает разделение логики, данных и пользовательского интерфейса. Такой подход упрощает поддержку и масштабирование проекта, а также способствует повышению читаемости кода [12].

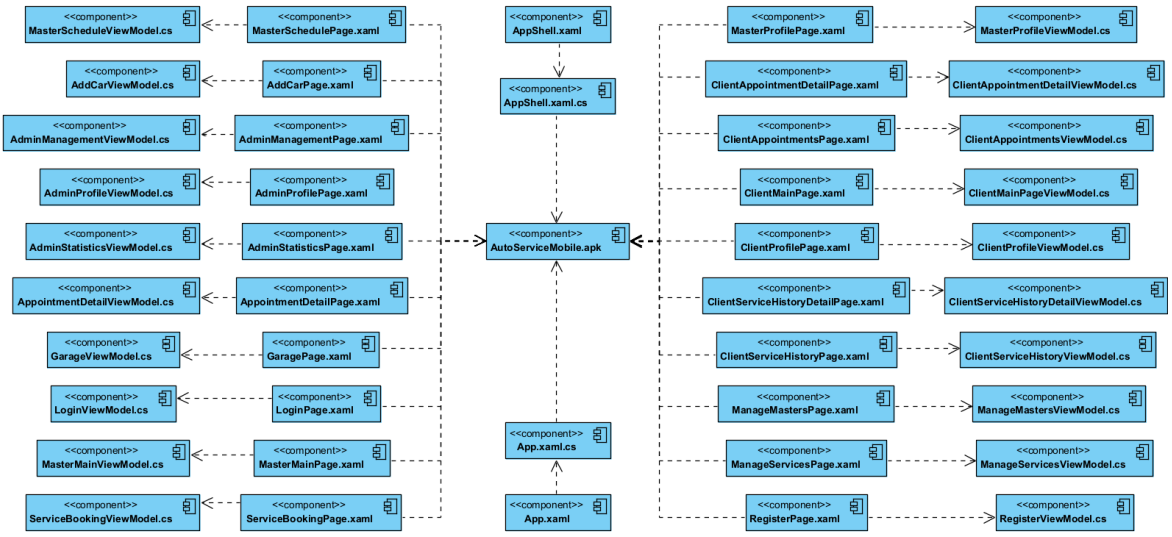

Рисунок 35 – Диаграмма компонентов мобильного приложения

Описание компонентов представлено в таблице 11.

Таблица 11 – Компоненты клиентского приложения

Компонент	Назначение
App.xaml	XAML-файл описания подключаемых ресурсов к приложению
App.xaml.cs	Является начальной точкой загрузки и закрытия приложения
AppShell.xaml	XAML-файл описания структуры интерфейса приложения
AppShell.xaml.cs	Регистрирует страницы приложения, настраивает маршруты навигации
AddCarPage.xaml AddCarViewModel.cs	Добавление автомобиля клиентом
AdminManagementPage.xaml AdminManagementViewModel.cs	Управление пользователями и правами доступа
AdminProfilePage.xaml AdminProfileViewModel.cs	Профиль администратора
AdminStatisticsPage.xaml AdminStatisticsViewModel.cs	Просмотр статистики по сервису
AppointmentDetailPage.xaml AppointmentDetailViewModel.cs	Детали записи на услугу
ClientAppointmentDetailPage.xaml ClientAppointmentDetailViewModel.cs	Подробности клиентской записи
ClientAppointmentsPage.xaml ClientAppointmentsViewModel.cs	Список записей клиента
ClientMainPage.xaml ClientMainPageViewModel.cs	Главная страница клиента

Продолжение таблицы 11

Компонент	Назначение
ClientProfilePage.xaml ClientProfileViewModel.cs	Профиль клиента
ClientServiceHistoryDetailPage.xaml ClientServiceHistoryDetailViewModel.cs	Детальный просмотр истории обслуживания
ClientServiceHistoryPage.xaml ClientServiceHistoryViewModel.cs	История всех обслуживаний клиента
GaragePage.xaml GarageViewModel.cs	Список автомобилей клиента
LoginPage.xaml LoginViewModel.cs	Страница входа в систему
ManageMastersPage.xaml ManageMastersViewModel.cs	Управление мастерами
ManageServicesPage.xaml ManageServicesViewModel.cs	Управление услугами
MasterMainPage.xaml MasterMainViewModel.cs	Главная страница мастера
MasterProfilePage.xaml MasterProfileViewModel.cs	Профиль мастера
MasterSchedulePage.xaml MasterScheduleViewModel.cs	Расписание работы мастера
RegisterPage.xaml RegisterViewModel.cs	Страница регистрации нового пользователя
ServiceBookingPage.xaml ServiceBookingViewModel.cs	Запись на услугу

Диаграмма развертывания, на которой описывается физическое расположение компонентов системы, представлена на рисунке 36.

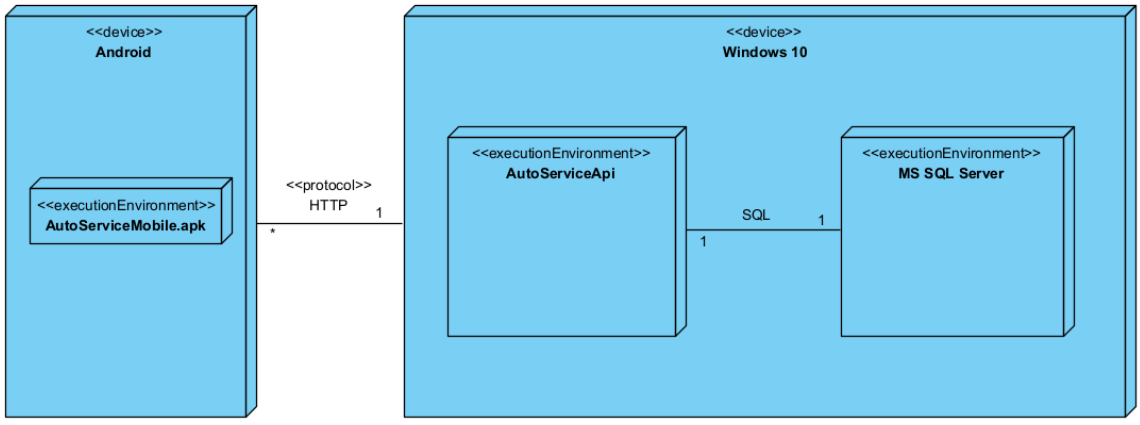

Рисунок 36 – Диаграмма развертывания мобильного приложения

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. Результаты тестирования функционала серверной части представлены в таблице 12. Скриншоты тестов находятся в приложении Д.

Таблица 12 – Результаты тестирования серверной части

№	Описание теста	Запрос		Результат
		Метод	Параметры	
1	Добавление клиента с email ivan22@mail.com и паролем ivan22	POST	email=ivan22@mail.com, password= ivan22	Тест пройден. Пользователь успешно зарегистрирован. HTTP 200 (рисунок Е.1)
2	Авторизация клиента с email ivan22@mail.com и паролем ivan22	POST	email=ivan22@mail.com, password=ivan22	Тест пройден. Авторизация выполнена. HTTP 200 (рисунок Е.2)
3	Получение сведений о клиенте с id=2	GET	id=2	Тест пройден. Информация получена. HTTP 200 (рисунок Е.3)
4	Изменение пароля клиента	PUT	id=2, oldPassword=test, newPassword=testtest	Тест пройден. Пароль изменён. HTTP 200 (рисунок Е.4)
5	Создание записи на услугу «Диагностика»	POST	clientId=2, serviceId=1, carId=1, masterId=2, date=2025–06–15, time=13:00	Тест пройден. Запись создана. HTTP 200 (рисунок Е.5)
6	Отмена записи с id=12	DELETE	appointmentId=12	Тест пройден. Запись отменена. HTTP 204 (рисунок Е.6)
7	Изменение времени записи с id=12	PUT	appointmentId=12, newTime=14:30	Тест пройден. Время обновлено. HTTP 200 (рисунок Е.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. Его результаты представлены в таблице 13. Скриншоты тестов находятся в приложении Е.

Тестирование охватывало ключевые сценарии взаимодействия пользователя с интерфейсом приложения. Особое внимание было уделено валидации данных.

Таблица 13 – Функциональное тестирование

Вариант использования	Тест и параметры	Результат
Регистрация	Ввести валидные данные: ФИО: Иванов И.С. Email: ivannnn@mail.ru Пароль: ivannn	Тест пройден. Пользователь успешно зарегистрирован в системе (рисунок Ж.1)
	Ввести существующий Email: Email: ivan222@mail.ru	Тест пройден. Выведено сообщение о занятости почты (рисунок Ж.2)
	Ввести недопустимые символы: ФИО: ADJS@@@ Email: jjj@@	Тест пройден. Выведено предупреждение о недопустимых данных (рисунок Ж.3)
	Ввести короткий пароль: Пароль: qo1	Тест пройден. Выведено предупреждение о недопустимой длине (рисунок Ж.4)
Вход в систему	Ввести верные данные: Email: ivannnn@mail.ru Пароль: ivannn	Тест пройден. Пользователь авторизован (рисуноки Ж.5 и Ж.6)
	Ввести неверный пароль: Email: ivan22@mail.ru Пароль: oo1	Тест пройден. Выведено сообщение о неверных данных (рисунок Ж.7)
Запись на услугу	Выбрать услугу: Замена масла Выбрать авто: Toyota Camry Дата: 14.06.2025 Время: 10:00	Тест пройден. Запись успешно создана (рисунок Ж.8)

Продолжение таблицы 13

Вариант использования	Тест и параметры	Результат
Добавление мастера	Ввести корректные данные мастера: ФИО: Иванов И.И. Email: master@mail.ru Телефон: +79991234567	Тест пройден. Мастер добавлен, отображён в списке (рисунок Ж.9)
	Ввести email уже существующего пользователя: Email: oldmaster@mail.ru	Тест пройден. Выведено сообщение: «Пользователь с таким email уже существует» (рисунок Ж.10)
	Ввести номер телефона в некорректном формате: Телефон: 12345	Тест пройден. Отображено предупреждение о неверном формате (рисунок Ж.11)
	Не заполнить обязательные поля: Оставить пустыми: ФИО, email	Тест пройден. Появилось сообщение: «Заполните все обязательные поля» (рисунок Ж.12)

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.

Результат проведения модульных тестов представлен на рисунке 37.

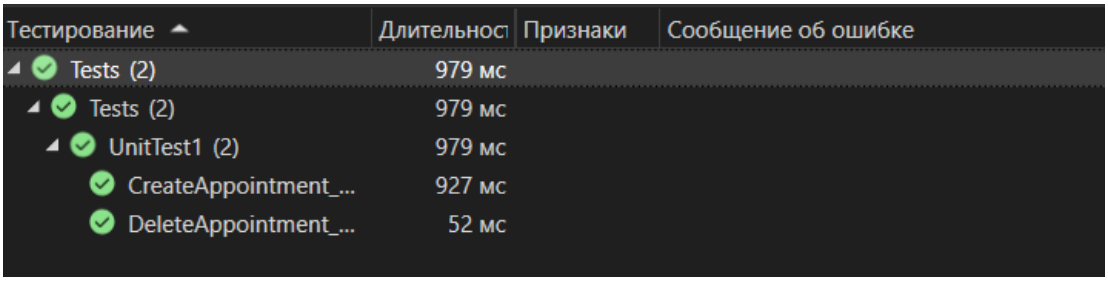

Рисунок 37 – Результаты модульного тестирования

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 CreateAppointment класса AppointmentController, , реализующий создание записи на услугу. Его код представлен в листинге 1.

Листинг 1 – Метод создания новой записи на услугу

```
/// <summary>
/// Метод обработки создания новой записи на услугу.
/// Доступен только для авторизованного клиента.
/// </summary>
/// <param name="appointment">Данные создаваемой записи</param>
/// <returns>Результат выполнения с созданной записью</returns>
[HttpPost]
[Authorize(Roles = "Client")]
public async Task<ActionResult<Appointment>>
CreateAppointment(Appointment appointment)
{
    var userId =
int.Parse(User.FindFirstValue(ClaimTypes.NameIdentifier)!); //
Получение ID клиента из токена
    appointment.ClientId = userId; // Присвоение ID клиента записи

    _context.Appointments.Add(appointment); // Добавление записи в
контекст
    await _context.SaveChangesAsync(); // Сохранение изменений в
базе

    // Возврат ответа 201 с указанием метода получения этой записи
    return CreatedAtAction(nameof(GetAppointment), new { id =
appointment.Id }, appointment);
}
```

Данный метод находится в серверной части приложения и выполняет создания новой записи на услугу.

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	11
Количество строк комментариев	4

Также вычислим процент комментариев. Он равен (4/11)*100=36.36. 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:

- 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). Данные словари были сформированы на основе анализа исходного кода приложения и позволяют точно определить параметры, необходимые для расчёта метрик.

Таблица 15 – Словарь операторов

Операторы	Число вхождений
=	2
.	4
var	1
await	1
return	1
()	1
n1 = 6	N1 = 10

Таблица 16 – Словарь операндов

Операнды	Число вхождений
userId	2
appointment	4
_context	1
Appointments	1
SaveChangesAsync	1
GetAppointment	1
n2 = 6	N2 = 9

Таким образом, число уникальных операторов программы $n1 = 6$, число уникальных операндов программы $n2 = 6$, общее число операторов в программе $N1 = 10$, общее число операндов в программе $N2 = 9$.

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

Расчёт словаря программы производится по формуле:

$$n = n1 + n2 = 6 + 6 = 12,$$

где n1 – количество операторов в программе;

n2 – количество операндов программе.

Расчёт длины программы производится по формуле:

$$N = N1 + N2 = 10 + 9 = 19,$$

где N1 – число вхождений всех операторов в программе;

N2 – число вхождений всех операндов программе.

Расчёт объёма программы производится по формуле:

$$V = N * log_2 n = 19 * log_2 12 = 68{,}10,$$

где N – длина программы;

n – словарь программы.

Расчёт сложности программы производится по формуле:

$$D = (n1/2) * (N2/n2) = (6 / 2) * (9 / 6) = 4{,}5,$$

где n1 – количество операторов в программе;

n2 – количество операндов программе;

N2 – число вхождений всех операндов программе.

Расчёт оценки интеллектуальных усилий производится по формуле:

$$E = D * V = 4{,}5 * 68{,}10 = 306{,}45,$$

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 были созданы программные средства для поддержки процессов функционирования автосервиса.

На этапе анализа предметной области и требований был выполнен анализ программного обеспечения аналогичного назначения. Анализ показал, что существующие решения обладают рядом недостатков, например устаревший интерфейс, отсутствие мобильного приложения. Проведённый анализ целевой аудитории, а также государственных программ «Цифровая экономика Российской Федерации» и «Стратегии развития информационного общества» подтвердили актуальность цифровизации сервисных услуг, в том числе в сфере автосервиса. Были определены основные функциональные и нефункциональные требования к системе.

В рамках планирования проекта была проведена оценка трудозатрат и бюджета. Общая продолжительность реализации проекта составила 66 дней, а расчётные затраты – 499700 рублей, что подтверждает его экономическую целесообразность.

На этапе проектирования были разработаны концептуальная и логическая модели базы данных. Для хранения и обработки данных была выбрана СУБД Microsoft SQL Server 2019, а взаимодействие с базой реализовано с использованием языка T-SQL. Серверная часть системы была реализована на платформе ASP.NET Core с использованием языка C# в среде Microsoft Visual Studio 2022, а клиентская часть – на фреймворке .NET MAUI, что обеспечивает высокую скорость работы приложения и современный пользовательский интерфейс.

На этапе реализации было произведено физическое проектирование базы данных, развертывание базы данных в СУБД Microsoft SQL Server 2019. Также был реализован весь спроектированный функционал.

В перспективе планируется опубликовать разработанное приложение в магазине мобильных приложений для системы Android Google Play, а серверная

часть будет размещена на облачном хостинге, что обеспечит широкую доступность системы для пользователей.

Таким образом, все цели и задачи, поставленные в рамках выпускной квалификационной работы, были достигнуты. Разработанная система обладает широким функционалом и может быть успешно внедрена в деятельность предприятий автосервиса.

Список использованных источников

1. Демура, Н.А. Цифровизация: сущность и роль в развитии национальной экономики / Н.А. Демура, Н.П. Путинцева // Научный результат. Экономические исследования. – 2021. – №1. – С. 22 – 30 (дата обращения: 03.04.2025).

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: 03.04.2025).

3. 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: 03.04.2025).

4. Росстат статистика легковых автомобилей. [Электронный ресурс]. URL: https://www.akm.ru/news/rosstat_v_rf_na_konets_2023_goda_naschityvalos_51_5_mln_legkovykh_avtomobiley/ (дата обращения: 18.06.2025).

5. Отчет «Рынок автосервиса в России», подготовленном экспертами агентства «АВТОСТАТ». [Электронный ресурс]. URL: https://www.autostat.ru/infographics/59270/ (дата обращения: 03.04.2025)

6. Сайт компании Wilgood. [Электронный ресурс]. URL: https://wilgood.ru/ (дата обращения: 03.04.2025)

7. Сервис записи на автосервис CheckAuto58. [Электронный ресурс]. URL: https://checkauto58.ru/zapis-na-servis (дата обращения: 03.04.2025)

8. Сайт автосервиса Avto-Spec. [Электронный ресурс]. URL: https://avto-spec.pro (дата обращения: 03.04.2025)

9. Аналитический обзор мирового рынка смартфонов в 2024 году, Merlion. [Электронный ресурс]. URL: https://merlion.com/press-

center/news/analiticheskiy-obzor-mirovoy-rynok-smartfonov-v-2024-godu (дата обращения: 03.04.2025)

10. Android Distribution Chart, Composables. [Электронный ресурс]. URL: https://composables.com/android-distribution-chart (дата обращения: 03.04.2025)

11. Якобсон, И. UML. Методология объектно-ориентированного анализа и проектирования / И. Якобсон, М. Кристерсон, П. Джонсон. – М. : Питер, 2019. – 384 с.

12. Троелсен, Э. Язык программирования C# 9.0 и платформа .NET 5 / Э. Троелсен, Ф. Джепикс. – М. : Вильямс, 2021. – 992 с.

13. Бьюдер, А. Xamarin и .NET MAUI: разработка кроссплатформенных мобильных приложений / А. Бьюдер. – СПб. : БХВ-Петербург, 2023. – 368 с.

14. Макдональд, М. Android. Программирование для профессионалов / М. Макдональд. – М. : Вильямс, 2020. – 832 с.

15. Жарков, С. В. Microsoft SQL Server: администрирование и сопровождение / С. В. Жарков. – М. : Диалектика, 2022. – 544 с.

16. Кормен, Т. Введение в алгоритмы / Т. Кормен и др. – М. : Вильямс, 2018. – 1296 с.

17. Павлов, А. Профессиональное проектирование UML-диаграмм / А. Павлов. – М. : Лаборатория знаний, 2017. – 304 с.

18. Тененбаум, А. Структуры данных и алгоритмы в C# / А. Тененбаум. – М. : Диалектика, 2021. – 528 с.

19. Грин, Э. Современный автосервис: технологии и управление / Э. Грин. – СПб. : Наука и Техника, 2020. – 286 с.

20. Мартин, Р. Чистый код: создание, анализ и рефакторинг / Р. Мартин. – М. : Питер, 2021. – 464 с.

21. Что такое .NET MAUI и зачем он нужен разработчику? // Хабр [Электронный ресурс]. URL: https://habr.com/ru/articles/664630/ (дата обращения: 15.05.2025)

22. Построение диаграмм классов UML: руководство для начинающих // Tproger [Электронный ресурс]. URL: https://tproger.ru/articles/class-diagram-uml (дата обращения: 10.05.2025)

23. Основы SQL и реляционных баз данных // SQL Academy [Электронный ресурс]. URL: https://sql-academy.org/ru/guide/sql-basics (дата обращения: 12.05.2025)

24. Разработка Android-приложений на C# с помощью Xamarin // Microsoft Learn [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/xamarin/android/ (дата обращения: 11.05.2025)

25. Как проектировать базы данных под Microsoft SQL Server // Хабр [Электронный ресурс]. URL: https://habr.com/ru/articles/535680/ (дата обращения: 18.05.2025)

26. Принципы построения архитектуры автосервиса // АвтоБизнес [Электронный ресурс]. URL: https://abiznes.ru/articles/architecture-service (дата обращения: 19.05.2025)

27. Всё о платформе .NET: история, компоненты, развитие // Tproger [Электронный ресурс]. URL: https://tproger.ru/articles/all-about-dotnet (дата обращения: 17.05.2025)

28. Как выбрать СУБД: сравнение SQL Server, PostgreSQL и MySQL // SkillFactory Blog [Электронный ресурс]. URL: https://skillfactory.ru/blog/kak-vybrat-sql-db/ (дата обращения: 15.05.2025)

29. Управление клиентами в автосервисе: CRM и цифровые технологии // Бизнес.Ру [Электронный ресурс]. URL: https://www.business.ru/article/1908-avtoservis-crm (дата обращения: 14.05.2025)

30. Что такое ORM и зачем он нужен при работе с C# // DotNet.ru [Электронный ресурс]. URL: https://dotnet.ru/library/orm-csharp-guide (дата обращения: 13.05.2025)

Приложение А

(справочное)

Глоссарий

Android – операционная система для смартфонов, планшетов, ноутбуков, телевизоров и других устройств, используемая на подавляющем большинстве современных мобильных устройств.

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

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

СУБД (Система Управления Базой Данных) – специализированное программное обеспечение, предоставляющее программный интерфейс, который дает возможность управлять созданием и использованием баз данных.

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания и модификации данных в реляционной базе данных.

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

REST API – архитектурный стиль взаимодействия компонентов распределенного приложения сети.

Приложение Б

(обязательное)

Исходный код скриптов базы данных

```
USE [master]

GO

/****** Object:  Database [AutoServiceDB]    Script Date:
18.06.2025 14:47:52 ******/

CREATE DATABASE [AutoServiceDB]

 CONTAINMENT = NONE

 ON  PRIMARY

( NAME = N'CarServiceDB', FILENAME = N'C:\Program Files\Microsoft
SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\CarServiceDB.mdf' , SIZE
= 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )

 LOG ON

( NAME = N'CarServiceDB_log', FILENAME = N'C:\Program
Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\CarServiceDB_log.ldf' , SIZE
= 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

 WITH CATALOG_COLLATION = DATABASE_DEFAULT, LEDGER = OFF

GO

ALTER DATABASE [AutoServiceDB] SET COMPATIBILITY_LEVEL = 160

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [AutoServiceDB].[dbo].[sp_fulltext_database] @action =
'enable'

end

GO

ALTER DATABASE [AutoServiceDB] SET ANSI_NULL_DEFAULT OFF

GO

ALTER DATABASE [AutoServiceDB] SET ANSI_NULLS OFF

GO

ALTER DATABASE [AutoServiceDB] SET ANSI_PADDING OFF

GO

ALTER DATABASE [AutoServiceDB] SET ANSI_WARNINGS OFF

GO

GO

ALTER DATABASE [AutoServiceDB] SET READ_COMMITTED_SNAPSHOT OFF
```

```
GO

ALTER DATABASE [AutoServiceDB] SET HONOR_BROKER_PRIORITY OFF

GO

ALTER DATABASE [AutoServiceDB] SET RECOVERY FULL

GO

ALTER DATABASE [AutoServiceDB] SET  MULTI_USER

GO

ALTER DATABASE [AutoServiceDB] SET PAGE_VERIFY CHECKSUM

GO

ALTER DATABASE [AutoServiceDB] SET DB_CHAINING OFF

GO

ALTER DATABASE [AutoServiceDB] SET FILESTREAM(
NON_TRANSACTED_ACCESS = OFF )

GO

ALTER DATABASE [AutoServiceDB] SET TARGET_RECOVERY_TIME = 60
SECONDS

GO

ALTER DATABASE [AutoServiceDB] SET DELAYED_DURABILITY = DISABLED

GO

ALTER DATABASE [AutoServiceDB] SET ACCELERATED_DATABASE_RECOVERY =
OFF

GO

EXEC sys.sp_db_vardecimal_storage_format N'AutoServiceDB', N'ON'

GO

ALTER DATABASE [AutoServiceDB] SET QUERY_STORE = ON

GO

ALTER DATABASE [AutoServiceDB] SET QUERY_STORE (OPERATION_MODE =
READ_WRITE, CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 30),
DATA_FLUSH_INTERVAL_SECONDS = 900, INTERVAL_LENGTH_MINUTES = 60,
MAX_STORAGE_SIZE_MB = 1000, QUERY_CAPTURE_MODE = AUTO,
SIZE_BASED_CLEANUP_MODE = AUTO, MAX_PLANS_PER_QUERY = 200,
WAIT_STATS_CAPTURE_MODE = ON)

GO

USE [AutoServiceDB]

CREATE TABLE [dbo].[Appointments](
```

```
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [ClientId] [int] NOT NULL,
    [MasterId] [int] NOT NULL,
    [ServiceId] [int] NOT NULL,
    [StartDateTime] [datetime] NOT NULL,
    [EndDateTime] [datetime] NULL,
    [Status] [nvarchar](20) NOT NULL,
    [Comment] [nvarchar](500) NULL,
    [CreatedAt] [datetime] NOT NULL,
CREATE TABLE [dbo].[Cars](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [ClientId] [int] NOT NULL,
    [Brand] [nvarchar](100) NOT NULL,
    [Model] [nvarchar](100) NOT NULL,
    [Number] [nvarchar](20) NOT NULL,
    [Year] [int] NULL,
CREATE TABLE [dbo].[Notifications](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [UserId] [int] NOT NULL,
    [SenderId] [int] NULL,
    [Title] [nvarchar](100) NOT NULL,
    [Message] [nvarchar](500) NOT NULL,
    [SentAt] [datetime] NOT NULL,
    [IsRead] [bit] NOT NULL,
CREATE TABLE [dbo].[Reviews](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [ClientId] [int] NOT NULL,
    [MasterId] [int] NOT NULL,
    [Rating] [int] NOT NULL,
    [Comment] [nvarchar](500) NULL,
    [CreatedAt] [datetime] NOT NULL,
CREATE TABLE [dbo].[Services](
```

```
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Name] [nvarchar](100) NOT NULL,
    [Description] [nvarchar](500) NULL,
    [Price] [decimal](18, 2) NOT NULL,
    [DurationMinutes] [int] NOT NULL,
CREATE TABLE [dbo].[Users](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [FullName] [nvarchar](100) NOT NULL,
    [Email] [nvarchar](100) NOT NULL,
    [PasswordHash] [nvarchar](256) NOT NULL,
    [Role] [nvarchar](20) NOT NULL,
    [PhoneNumber] [nvarchar](20) NULL,
CREATE TABLE [dbo].[WorkSlots](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [MasterId] [int] NOT NULL,
    [StartDateTime] [datetime] NOT NULL,
    [EndDateTime] [datetime] NOT NULL,
    [IsBooked] [bit] NOT NULL,
```

Приложение В

(обязательное)

Исходный код серверной части приложения

Листинг В.1 – Файл Program.cs

```
namespace AutoServiceApi
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);
            builder.WebHost.UseUrls("http://0.0.0.0:5198");
            // Add services to the container.

            // Подключение к БД
            builder.Services.AddDbContext<AppDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("De
faultConnection")));


            builder.Services.AddControllers();
                // JWT Bearer
                options.AddSecurityRequirement(new
OpenApiSecurityRequirement
                {
                    {
                        new OpenApiSecurityScheme
                        {
                            Reference = new OpenApiReference {
Type = ReferenceType.SecurityScheme, Id = "Bearer" }
                        },
                        Array.Empty<string>()
                    }
                });
            });

            builder.Services.AddAuthentication("Bearer")
                .AddJwtBearer("Bearer", options =>
                {
                    options.TokenValidationParameters = new
TokenValidationParameters
                    {
                        ValidateIssuer = true,
                        ValidateAudience = true,
                        ValidateLifetime = true,
                        ValidateIssuerSigningKey = true,
                        ValidIssuer =
builder.Configuration["Jwt:Issuer"],
                        ValidAudience =
builder.Configuration["Jwt:Audience"],
                        IssuerSigningKey = new
SymmetricSecurityKey(

Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]!))
```

```
                    };
                });

            var app = builder.Build();
            app.UseHttpsRedirection();

            app.UseAuthorization();


            app.MapControllers();

            app.Run();
        }
    }
}
}
```

Листинг В.2 – Файл WorkSlot.cs

```
public class WorkSlot
{
    public int Id { get; set; }

    public int MasterId { get; set; }

    public DateTime StartDateTime { get; set; }
}
```

Листинг В.3 – Файл User.cs

```
public enum UserRole { Client, Master, Admin }

public class User
{
    public int Id { get; set; }
    public string FullName { get; set; } = null!;
    public string Email { get; set; } = null!;
    public string PasswordHash { get; set; } = null!;
    public string Role { get; set; }
    public string? PhoneNumber { get; set; }
}
```

Листинг В.4 – Файл Service.cs

```
public class Service
{
    public int Id { get; set; }
    public string Name { get; set; } = null!;

    public ICollection<Appointment>? Appointments { get; set; }
}
```

Листинг В.5 – Файл Review.cs

```
namespace AutoServiceApi.Models;
```

```
public class Review
{
    public int Id { get; set; }

    public int ClientId { get; set; }

    public int MasterId { get; set; }
    public User Master { get; set; } = null!;
}
```

Листинг В.6 – Файл Notification.cs

```
namespace AutoServiceApi.Models;

public class Notification
{
    public int Id { get; set; }

    public int? SenderId { get; set; }
    public User? Sender { get; set; }
}
```

Листинг В.7 – Файл Car.cs

```
namespace AutoServiceApi.Models
{
    public class Car
    {
        public int Id { get; set; }

        [Required]
        [MaxLength(100)]
        public string Model { get; set; }
        public User Client { get; set; }
    }
}
```

Листинг В.8 – Файл Appointment.cs

```
namespace AutoServiceApi.Models;

public enum AppointmentStatus { Scheduled, InProgress, Completed,
Cancelled }


public class Appointment
{
    public int Id { get; set; }

    public int ClientId { get; set; }

    public User Client { get; set; } = null!;
}
```

Листинг В.9 – Файл RegisterRequest.cs

```
public class RegisterRequest
    {
        public string FullName { get; set; } = null!;
        public string Email { get; set; } = null!;
        public string Password { get; set; } = null!;
        public string? PhoneNumber { get; set; }
        public string Role { get; set; } =
UserRole.Client.ToString();
    }
```

Листинг В.10 – Файл LoginRequest.cs

```
namespace AutoServiceApi.Models.Auth
{
    public class LoginRequest
    {
        public string Email { get; set; } = null!;
        public string Password { get; set; } = null!;
    }
}
```

Листинг В.11 – Файл AppDbContext.cs

```
public class AppDbContext : DbContext
{
    public AppDbContext(DbContextOptions<AppDbContext> options) :
base(options) { }
    protected override void OnModelCreating(ModelBuilder
modelBuilder)
    {
        // USER - APPOINTMENTS (Client)
        modelBuilder.Entity<User>()
            .HasMany(u => u.ClientAppointments)
            .WithOne(a => a.Client)
            .HasForeignKey(a => a.ClientId)
            .OnDelete(DeleteBehavior.Restrict);

        // USER - APPOINTMENTS (Master)
        modelBuilder.Entity<User>()
            .HasMany(u => u.MasterAppointments)
            .WithOne(a => a.Master)
            .HasForeignKey(a => a.MasterId)
            .OnDelete(DeleteBehavior.Restrict);

        // USER - WORKSLOTS
        modelBuilder.Entity<User>()
            .HasMany(u => u.WorkSlots)
            .WithOne(ws => ws.Master)
            .HasForeignKey(ws => ws.MasterId)
            .OnDelete(DeleteBehavior.Restrict);

        // USER - REVIEWS (written)
```

```
        modelBuilder.Entity<User>()
            .HasMany(u => u.ReviewsWritten)
            .WithOne(r => r.Client)
            .HasForeignKey(r => r.ClientId)
            .OnDelete(DeleteBehavior.Restrict);

        // USER - REVIEWS (received)
        modelBuilder.Entity<User>()
            .HasMany(u => u.ReviewsReceived)
            .WithOne(r => r.Master)
            .HasForeignKey(r => r.MasterId)
            .OnDelete(DeleteBehavior.Restrict);

        // USER - NOTIFICATIONS (received)
        modelBuilder.Entity<User>()
            .HasMany(u => u.NotificationsReceived)
            .WithOne(n => n.User)
            .HasForeignKey(n => n.UserId)
            .OnDelete(DeleteBehavior.Restrict);

        // USER - NOTIFICATIONS (sent)
        modelBuilder.Entity<User>()
            .HasMany(u => u.NotificationsSent)
            .WithOne(n => n.Sender)
            .HasForeignKey(n => n.SenderId)
            .OnDelete(DeleteBehavior.Restrict);

        // USER - CARS
        modelBuilder.Entity<User>()
            .HasMany(u => u.Cars)
            .WithOne(c => c.Client)
            .HasForeignKey(c => c.ClientId)
            .OnDelete(DeleteBehavior.Restrict);
    }
}
```

Листинг В.12 – Файл WorkSlotsController.cs

```
namespace AutoServiceApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class WorkSlotsController : ControllerBase
    {
        [HttpGet("mine")]
        public async Task<IActionResult> GetMySlots()
        {
            var masterId = int.Parse(User.FindFirst("id")!.Value);
            var slots = await _context.WorkSlots
                .Where(ws => ws.MasterId == masterId)
                .OrderBy(ws => ws.StartDateTime)
                .ToListAsync();
            return Ok(slots);
        }
```

```
        /// <summary>
        /// Добавить рабочий слот
        /// </summary>
        [HttpPost]
        public async Task<IActionResult> AddSlot([FromBody]
WorkSlotCreateRequest req)
        {
            var masterId = int.Parse(User.FindFirst("id")!.Value);
            var slot = new WorkSlot
            {
                MasterId = masterId,
                StartDateTime = req.StartDateTime,
                EndDateTime = req.EndDateTime,
                IsBooked = false
            };
            _context.WorkSlots.Add(slot);
            await _context.SaveChangesAsync();
            return Ok(new { slot.Id });
        }

        /// <summary>
        /// Удалить слот
        /// </summary>
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteSlot(int id)
        {
            var masterId = int.Parse(User.FindFirst("id")!.Value);

            return Ok();
        }
    }

    public class WorkSlotCreateRequest
    {
        public DateTime StartDateTime { get; set; }
        public DateTime EndDateTime { get; set; }
    }
}
```

Листинг В.13 – Файл ServiceController.cs

```
public class ServiceController : ControllerBase
{
    public ServiceController(AppDbContext context)
    {
        _context = context;
    }

    // POST: api/service
    [HttpPost]
    [Authorize(Roles = "Admin")]
    public async Task<ActionResult<Service>> CreateService(Service
service)
```

```
    {
        _context.Services.Add(service);
        await _context.SaveChangesAsync();
        return CreatedAtAction(nameof(GetService), new { id =
service.Id }, service);
    }

    // PUT: api/service/5
    [HttpPut("{id}")]
    [Authorize(Roles = "Admin")]
    public async Task<IActionResult> UpdateService(int id, Service
updatedService)
    {
        if (id != updatedService.Id) return BadRequest();

        _context.Entry(updatedService).State =
EntityState.Modified;

        try
        {
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!_context.Services.Any(e => e.Id == id))
                return NotFound();
            throw;
        }

        return NoContent();
    }
}
```

Листинг В.14 – Файл ReviewsController.cs

```
public class ReviewsController : ControllerBase
    {
        /// <summary>
        /// Клиент оставляет отзыв по ремонту
        /// </summary>
        [HttpPost]
        [Authorize(Roles = "Client")]
        public async Task<IActionResult> AddReview([FromBody]
ReviewCreateRequest request)
        {
            var clientId = int.Parse(User.FindFirst("id")!.Value);
            var review = new Review
            {
                ClientId = clientId,
                MasterId = request.MasterId,
                Rating = request.Rating,
                Comment = request.Comment,
                CreatedAt = DateTime.UtcNow
            };
```

```
            _context.Reviews.Add(review);
            await _context.SaveChangesAsync();
            return Ok(new { review.Id });
        }

        /// <summary>
        /// Получить отзывы о мастере
        /// </summary>
        [HttpGet("master/{masterId}")]
        public async Task<IActionResult> GetReviewsForMaster(int
masterId)
        {
            var reviews = await _context.Reviews
                .Where(r => r.MasterId == masterId)
                .Select(r => new
                {
                    r.Id,
                    r.ClientId,
                    r.Rating,
                    r.Comment,
                    r.CreatedAt
                }).ToListAsync();

            return Ok(reviews);
        }
    }
}
```

Листинг В.15 – Файл NotificationsController.cs

```
public class NotificationsController : ControllerBase
    {
        [HttpGet]
        public async Task<IActionResult> GetMyNotifications()
        {
            var userId = int.Parse(User.FindFirst("id")!.Value);
            var notifs = await _context.Notifications
                .Where(n => n.UserId == userId)
                .OrderByDescending(n => n.SentAt)
                .ToListAsync();
            return Ok(notifs);
        }

        /// <summary>
        /// Отметить уведомление как прочитанное
        /// </summary>
        [HttpPost("{id}/read")]
        public async Task<IActionResult> MarkRead(int id)
        {
            var userId = int.Parse(User.FindFirst("id")!.Value);
            var n = await
_context.Notifications.FirstOrDefaultAsync(nf => nf.Id == id &&
nf.UserId == userId);
            if (n == null) return NotFound();
```

```
            n.IsRead = true;
            await _context.SaveChangesAsync();
            return Ok();
        }

        /// <summary>
        /// Отправить уведомление (Admin или Master)
        /// </summary>
        [HttpPost]
        [Authorize(Roles = "Admin,Master")]
        public async Task<IActionResult>
SendNotification([FromBody] NotificationCreateRequest req)
        {
            var senderId = int.Parse(User.FindFirst("id")!.Value);
            var n = new Notification
            {
                UserId = req.UserId,
                SenderId = senderId,
                Title = req.Title,
                Message = req.Message,
                SentAt = DateTime.UtcNow,
                IsRead = false
            };
            _context.Notifications.Add(n);
            await _context.SaveChangesAsync();
            return Ok(new { n.Id });
        }
    }
}
```

Листинг В.16 – Файл MastersController.cs

```
namespace AutoServiceApi.Controllers
{
    public class MastersController : ControllerBase
    {
        /// <summary>
        /// Получает список всех мастеров
        /// </summary>
        [HttpGet]
        public async Task<IActionResult> GetMasters()
        {
            var masters = await _context.Users
                .Where(u => u.Role == "Master")
                .Select(u => new User
                {
                    FullName = u.FullName,
                    Email = u.Email,
                    PhoneNumber = u.PhoneNumber
                })
                .ToListAsync();

            return Ok(masters);
        }
```

```
        /// <summary>
        /// Получает расписание мастера
        /// </summary>
        [HttpGet("{masterId}/schedule")]
        public async Task<IActionResult> GetMasterSchedule(int
masterId)
        {
            var slots = await _context.WorkSlots
                .Where(ws => ws.MasterId == masterId)
                .OrderBy(ws => ws.StartDateTime)
                .Select(ws => new
                {
                    ws.Id,
                    ws.StartDateTime,
                    ws.EndDateTime,
                    ws.IsBooked
                })
                .ToListAsync();

            return Ok(slots);
        }
    }
}
```

Листинг В.17 – Файл CarsController.cs

```
namespace AutoServiceApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    [Authorize(Roles = "Client")]
    public class CarsController : ControllerBase
    {
        /// <summary>
        /// Получить список всех автомобилей текущего клиента
        /// </summary>
        [HttpGet]
        public async Task<IActionResult> GetMyCars()
        {
            var userId = int.Parse(User.FindFirst("id")!.Value);

            var cars = await _context.Cars
                .Where(c => c.ClientId == userId)
                .Select(c => new
                {
                    c.Id,
                    c.Brand,
                    c.Model,
                    c.Number,
                    c.Year
                })
                .ToListAsync();
```

```
            return Ok(cars);
        }

        /// <summary>
        /// Добавить новый автомобиль
        /// </summary>
        [HttpPost]
        public async Task<IActionResult> AddCar([FromBody]
CarCreateRequest request)
        {
            var userId = int.Parse(User.FindFirst("id")!.Value);

            var car = new Car
            {
                ClientId = userId,
                Brand = request.Brand,
                Model = request.Model,
                Number = request.Number,
                Year = request.Year
            };

            _context.Cars.Add(car);
            await _context.SaveChangesAsync();

            return Ok(new { car.Id });
        }

        /// <summary>
        /// Удалить автомобиль по Id
        /// </summary>
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteCar(int id)
        {
            var userId = int.Parse(User.FindFirst("id")!.Value);

            var car = await _context.Cars.FirstOrDefaultAsync(c =>
c.Id == id && c.ClientId == userId);

            if (car == null)
                return NotFound();

            _context.Cars.Remove(car);
            await _context.SaveChangesAsync();

            return Ok();
        }

        /// <summary>
        /// Обновить данные автомобиля
        /// </summary>
        [HttpPut("{id}")]
        public async Task<IActionResult> UpdateCar(int id,
[FromBody] CarUpdateRequest request)
```

```
        {
            var userId = int.Parse(User.FindFirst("id")!.Value);
return Ok1 ();
        }
    }

    // DTO для создания машины
    public class CarCreateRequest
    {
        public string Brand { get; set; } = null!;
        public string Model { get; set; } = null!;
        public string Number { get; set; } = null!;
        public int? Year { get; set; }
    }
}
```

Листинг В.18 – Файл AuthController.cs

```
namespace AutoServiceApi.Controllers;

[ApiController]
[Route("api/[controller]")]
public class AuthController : ControllerBase
{
    public AuthController(AppDbContext db, IConfiguration config)
    {
        _db = db;
        _config = config;
    }

    [HttpPost("register")]
    public async Task<IActionResult> Register(RegisterRequest
request)
    {
        if (await _db.Users.AnyAsync(u => u.Email ==
request.Email))
            return BadRequest("Email already in use.");

        var user = new User
        {
            FullName = request.FullName,
            Email = request.Email,
            PasswordHash =
BCrypt.Net.BCrypt.HashPassword(request.Password),
            Role = request.Role,
            PhoneNumber = request.PhoneNumber
        };

        _db.Users.Add(user);
        await _db.SaveChangesAsync();

        return Ok(new { message = "User registered successfully"
});
    }
```

```
    private string GenerateJwtToken(User user)
    {


        var key = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_config["Jwt:Key"]!));
        var creds = new SigningCredentials(key,
SecurityAlgorithms.HmacSha256);
        return new JwtSecurityTokenHandler().WriteToken(token);
    }


    [HttpPost("login")]
    public async Task<IActionResult> Login(LoginRequest request)
    {
        var user = await _db.Users.FirstOrDefaultAsync(u =>
u.Email == request.Email);
        if (user == null ||
!BCrypt.Net.BCrypt.Verify(request.Password, user.PasswordHash))
            return Unauthorized("Invalid credentials.");

        var token = GenerateJwtToken(user);
        return Ok(new { token, user.Role });
    }

    [Authorize]
    [HttpGet("profile")]
    public async Task<IActionResult> GetProfile()
    {
        var userIdClaim =
User.FindFirst(ClaimTypes.NameIdentifier);
        if (userIdClaim == null)
            return Unauthorized("User ID not found in token.");

        if (!int.TryParse(userIdClaim.Value, out int userId))
            return BadRequest("Invalid user ID.");
    }
}
```

Листинг В.19 – Файл AppointmentController.cs

```
[ApiController]
[Route("api/[controller]")]
public class AppointmentController : ControllerBase
{
    private readonly AppDbContext _context;
    public async Task<ActionResult<IEnumerable<Appointment>>>
GetMyAppointments()
    {
        var userId =
int.Parse(User.FindFirstValue(ClaimTypes.NameIdentifier)!);
        var role = User.FindFirstValue(ClaimTypes.Role)!;
```

```
        var query = _context.Appointments
            .Include(a => a.Service)
            .Include(a => a.Master)
            .Include(a => a.Client)
            .AsQueryable();

        return role switch
        {
            "Client" => await query.Where(a => a.ClientId ==
userId).ToListAsync(),
            "Master" => await query.Where(a => a.MasterId ==
userId).ToListAsync(),
            _ => Forbid()
        };
    }

    /// <summary>
    /// Метод обработки создания новой записи на услугу.
    /// Доступен только для авторизованного клиента.
    /// </summary>
    /// <param name="appointment">Данные создаваемой
записи</param>
    /// <returns>Результат выполнения с созданной
записью</returns>
    [HttpPost]
    [Authorize(Roles = "Client")]
    public async Task<ActionResult<Appointment>>
CreateAppointment(Appointment appointment)
    {
        var userId =
int.Parse(User.FindFirstValue(ClaimTypes.NameIdentifier)!); //
Получение ID клиента из токена
        appointment.ClientId = userId; // Присвоение ID клиента
записи

        _context.Appointments.Add(appointment); // Добавление
записи в контекст
        await _context.SaveChangesAsync(); // Сохранение изменений
в базе

        // Возврат ответа 201 с указанием метода получения этой
записи
        return CreatedAtAction(nameof(GetAppointment), new { id =
appointment.Id }, appointment);
    }


    // GET: api/appointment/5
    [HttpGet("{id}")]
    [Authorize]
    public async Task<ActionResult<Appointment>>
GetAppointment(int id)
    {
```

```
        var appointment = await _context.Appointments
            .Include(a => a.Service)
            .Include(a => a.Client)
            .Include(a => a.Master)
            .FirstOrDefaultAsync(a => a.Id == id);

        if (appointment == null) return NotFound();

        var userId =
int.Parse(User.FindFirstValue(ClaimTypes.NameIdentifier)!);
        var role = User.FindFirstValue(ClaimTypes.Role)!;

        if (role == "Admin" || appointment.ClientId == userId ||
appointment.MasterId == userId)
            return appointment;

        return Forbid();
    }

    // DELETE: api/appointment/5
    [HttpDelete("{id}")]
    [Authorize]
    public async Task<IActionResult> DeleteAppointment(int id)
    {
        var appointment = await
_context.Appointments.FindAsync(id);
        if (appointment == null) return NotFound();

        var userId =
int.Parse(User.FindFirstValue(ClaimTypes.NameIdentifier)!);
        var role = User.FindFirstValue(ClaimTypes.Role)!;

        if (role != "Admin" && appointment.ClientId != userId)
            return Forbid();

        _context.Appointments.Remove(appointment);
        await _context.SaveChangesAsync();

        return NoContent();
    }
}
```

Приложение Г

(обязательное)

Исходный код мобильного приложения

Листинг Г.1 – Файл AppShell.xaml

```
<Shell
    x:Class="AutoServiceMobile.AppShell"
    >

    <!-- Маршруты -->
    <ShellContent Title="Login" ContentTemplate="{DataTemplate
views:LoginPage}" Route="LoginPage"/>
    <ShellContent Title="Register" ContentTemplate="{DataTemplate
views:RegisterPage}" Route="RegisterPage"/>


    <!-- Клиент -->
    <ShellContent Title="Client" ContentTemplate="{DataTemplate
views:ClientMainPage}" Route="ClientMainPage" />

    <!-- Мастер -->
    <ShellContent Title="Master" ContentTemplate="{DataTemplate
views:MasterMainPage}" Route="MasterMainPage" />

    <!-- Админ -->
    <ShellContent Title="Admin" ContentTemplate="{DataTemplate
views:AdminMainPage}" Route="AdminMainPage" />

</Shell>
```

Листинг Г.2 – Файл AppShell.xaml.cs

```
public partial class AppShell : Shell
    {
        public AppShell()
        {

            // Регистрируем маршрут для страницы добавления авто
            Routing.RegisterRoute("AddCarPage",
typeof(AddCarPage));
            Routing.RegisterRoute("GaragePage",
typeof(GaragePage));
            Routing.RegisterRoute("ClientAppointmentsPage",
typeof(ClientAppointmentsPage));
            Routing.RegisterRoute("ClientProfilePage",
typeof(ClientProfilePage));
            Routing.RegisterRoute("ClientServiceHistoryPage",
typeof(ClientServiceHistoryPage));
            Routing.RegisterRoute("ServiceBookingPage",
typeof(ServiceBookingPage));
            Routing.RegisterRoute("ClientAppointmentDetailPage",
typeof(ClientAppointmentDetailPage));

Routing.RegisterRoute("ClientServiceHistoryDetailPage",
typeof(ClientServiceHistoryDetailPage));
```

```
            Routing.RegisterRoute("MasterSchedulePage",
typeof(MasterSchedulePage));
            Routing.RegisterRoute("MasterProfilePage",
typeof(MasterProfilePage));
            Routing.RegisterRoute("AppointmentDetailPage",
typeof(AppointmentDetailPage));


            Routing.RegisterRoute("AdminStatisticsPage",
typeof(AdminStatisticsPage));
            Routing.RegisterRoute("AdminManagementPage",
typeof(AdminManagementPage));
            Routing.RegisterRoute("AdminProfilePage",
typeof(AdminProfilePage));
            Routing.RegisterRoute("ManageMastersPage",
typeof(ManageMastersPage));     // потом реализуешь страницу
            Routing.RegisterRoute("ManageServicesPage",
typeof(ManageServicesPage));   // потом реализуешь страницу


            // При необходимости, сюда можно добавить другие
страницы с деталями и т.д.
        }

        public void LoadClientShell()
        {
            Items.Clear(); // Удаляем всё из Shell

            // TabBar клиента
            var tabBar = new TabBar();

            var homeTab = new Tab { Title = "Главная", Icon =
"home_icon.png" };
            homeTab.Items.Add(new ShellContent
            {
                ContentTemplate = new
DataTemplate(typeof(ClientMainPage)),
                Route = "ClientMainPage"
            });

            var appointmentsTab = new Tab { Title = "Мои записи",
Icon = "calendar_icon.png" };
            appointmentsTab.Items.Add(new ShellContent
            {
                ContentTemplate = new
DataTemplate(typeof(ClientAppointmentsPage)),
                Route = "ClientAppointmentsPage"
            });

            var profileTab = new Tab { Title = "Профиль", Icon =
"profile_icon.png" };
```

```
            profileTab.Items.Add(new ShellContent
            {
                ContentTemplate = new
DataTemplate(typeof(ClientProfilePage)),
                Route = "ClientProfilePage"
            });

            tabBar.Items.Add(homeTab);
            tabBar.Items.Add(appointmentsTab);
            tabBar.Items.Add(profileTab);

            Items.Add(tabBar);
        }

        public void LoadMasterShell()
        {
            Items.Clear(); // Очистить текущее содержимое Shell

            var tabBar = new TabBar();

            // Главная страница мастера
            var homeTab = new Tab { Title = "Главная", Icon =
"home_icon.png" };
            homeTab.Items.Add(new ShellContent
            {
                ContentTemplate = new
DataTemplate(typeof(MasterMainPage)),
                Route = "MasterMainPage"
            });

            // Расписание
            var scheduleTab = new Tab { Title = "Расписание", Icon
= "calendar_icon.png" };
            scheduleTab.Items.Add(new ShellContent
            {
                ContentTemplate = new
DataTemplate(typeof(MasterSchedulePage)),
                Route = "MasterSchedulePage"
            });

            // Профиль
            var profileTab = new Tab { Title = "Профиль", Icon =
"profile_icon.png" };
            profileTab.Items.Add(new ShellContent
            {
                ContentTemplate = new
DataTemplate(typeof(MasterProfilePage)),
                Route = "MasterProfilePage"
            });

            tabBar.Items.Add(homeTab);
            tabBar.Items.Add(scheduleTab);
            tabBar.Items.Add(profileTab);
```

```
            Items.Add(tabBar);
        }

        public void LoadAdminShell()
        {
            Items.Clear();

            var tabBar = new TabBar();

            // 1. Статистика
            var statsTab = new Tab { Title = "Статистика", Icon =
"icon_stats.png" };
            statsTab.Items.Add(new ShellContent
            {
                ContentTemplate = new
DataTemplate(typeof(AdminStatisticsPage)),
                Route = "AdminStatisticsPage"
            });

            // 2. Управление
            var manageTab = new Tab { Title = "Управление", Icon =
"icon_manage.png" };

            manageTab.Items.Add(new ShellContent
            {
                ContentTemplate = new
DataTemplate(typeof(AdminManagementPage)),
                Route = "AdminManagementPage"
            });

            // 3. Профиль
            var profileTab = new Tab { Title = "Профиль", Icon =
"profile_icon.png" };

            profileTab.Items.Add(new ShellContent
            {
                ContentTemplate = new
DataTemplate(typeof(AdminProfilePage)),
                Route = "AdminProfilePage"
            });

            tabBar.Items.Add(statsTab);
            tabBar.Items.Add(manageTab);
            tabBar.Items.Add(profileTab);

            Items.Add(tabBar);
        }


    }
}
```

Листинг Г.3 – Файл App.xaml

```
<Application    x:Class="AutoServiceMobile.App">
    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary>
                    <utils:NullToBoolConverter
x:Key="NullToBoolConverter"/>
                    <utils:InverseBoolConverter
x:Key="InverseBoolConverter"/>
                    <utils:BoolToColorConverter
x:Key="BoolToColorConverter"/>
                    <utils:StringNotEmptyConverter
x:Key="StringNotEmptyConverter"/>
                    <utils:SelectedTimeToBoolConverter
x:Key="SelectedTimeToBoolConverter"/>
                </ResourceDictionary>

            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Application.Resources>
</Application>
```

Листинг Г.4 – Файл App.xaml.cs

```
namespace AutoServiceMobile
{
    public partial class App : Application
    {
        public App()
        {

            CheckAuthentication();
        }

        private async void CheckAuthentication()
        {

                await Shell.Current.GoToAsync("//LoginPage");

        }
    }

}
```

Листинг Г.5 – Файл ServiceBookingPage.xaml

```
<ContentPage
    x:Class="AutoServiceMobile.Views.ServiceBookingPage"
    Title="Бронирование услуги"
    BackgroundColor="Black">

    <VerticalStackLayout Padding="16" Spacing="16">
```

```
        <!-- Stepper с поддержкой возврата назад -->
        <Grid ColumnDefinitions="auto,*,auto,*,auto,*,auto"
Padding="0,10,0,18" HeightRequest="36">
            <!-- Шаг 1 -->
            <Frame Grid.Column="0" BackgroundColor="{Binding
Step1Color}" CornerRadius="18" HeightRequest="36"
WidthRequest="36">
            <BoxView Grid.Column="1" Color="{Binding Step2Color}"
HeightRequest="3" VerticalOptions="Center"/>

            <!-- Шаг 2 -->
            <Frame Grid.Column="2" BackgroundColor="{Binding
Step2Color}" CornerRadius="18">
                <Frame.GestureRecognizers>
                    <TapGestureRecognizer Command="{Binding
GoToStepCommand}" CommandParameter="2"/>
                </Frame.GestureRecognizers>
            </Frame>
            <BoxView Grid.Column="3" Color="{Binding Step3Color}"
HeightRequest="3" VerticalOptions="Center"/>
</ContentPage>
```

Листинг Г.6 – Файл ServiceBookingPage.xaml.cs

```
public partial class ServiceBookingPage : ContentPage
    {
        private readonly ServiceBookingViewModel viewModel;

        public ServiceBookingPage()
        {

            viewModel.PropertyChanged +=
ViewModel_PropertyChanged;
            LoadStepContent();
        }
        private void ViewModel_PropertyChanged(object sender,
PropertyChangedEventArgs e)
        {
            // Следим только за сменой шага
            if (e.PropertyName == nameof(viewModel.CurrentStep))
            {
                LoadStepContent();
            }
        }
    }
}
```

Листинг Г.7 – Файл RegisterPage.xaml

```
<ContentPage


    <ContentPage.BindingContext>
        <vm:RegisterViewModel />
```

```
    </ContentPage.BindingContext>

    <ScrollView>
        <VerticalStackLayout Padding="32,60,32,0" Spacing="22">
            <!-- Пароль -->
            <Entry Placeholder="Пароль"
                       IsPassword="True"
                       Text="{Binding Password}"
                       TextColor="White"
                       BackgroundColor="Transparent"
                       FontSize="17"
                       PlaceholderColor="#2B9DF5"/>

            <!-- Телефон -->
            <Entry Placeholder="Телефон"
                       Text="{Binding PhoneNumber}"
                       Keyboard="Telephone"
                       TextColor="White"
                       BackgroundColor="Transparent"
                       FontSize="17"
                       PlaceholderColor="#2B9DF5"/>



            <!-- Зарегистрироваться -->
            <Button Text="Зарегистрироваться"
                    Command="{Binding RegisterCommand}"
                    BackgroundColor="#232323"
                    BorderColor="#2B9DF5"
                    BorderWidth="1"
                    TextColor="#2B9DF5"
                    FontAttributes="Bold"
                    CornerRadius="14"
                    HeightRequest="48"
                    FontSize="18"
                    Margin="0,64,0,0"/>

            <!-- Войти -->
            <Button Text="Войти"
                    Command="{Binding GoToLoginCommand}"
                    BackgroundColor="Transparent"
                    BorderColor="Transparent"
                    BorderWidth="0"
                    TextColor="#2B9DF5"
                    FontAttributes="Bold"
                    />
</ContentPage>
```

Листинг Г.8 – Файл RegisterPage.xaml.cs

```
public partial class RegisterPage : ContentPage
{
     public RegisterPage()
     {
```

```
    }
}
```

Листинг Г.9 – Файл MasterSchedulePage.xaml

```
<ContentPage
             Title="Расписание">

    <ScrollView>
        <VerticalStackLayout Padding="22" Spacing="14">

            <!-- Понедельник -->
            <toolkit:Expander IsExpanded="False"
BindingContext="{Binding Monday}"
ExpandedChanged="OnExpandedChanged">
                <toolkit:Expander.Header>
                    <Frame BackgroundColor="#232323"
                           BorderColor="#2B9DF5"
                           CornerRadius="14"
                           Padding="0"
                           HasShadow="False"
                           Margin="0,10,0,0"
                           HorizontalOptions="Fill">
                        <Label Text="{Binding Day}"
                               FontAttributes="Bold"
                               FontSize="18"
                               TextColor="#2B9DF5"
                               HorizontalOptions="Center"
                               VerticalOptions="Center"
                               Margin="12,8"/>
                    </Frame>
                </toolkit:Expander.Header>
                <VerticalStackLayout>
                    <ActivityIndicator IsRunning="{Binding
IsLoading}" IsVisible="{Binding IsLoading}" Margin="0,0,0,4"/>
                    <CollectionView ItemsSource="{Binding
Intervals}">
                        <CollectionView.ItemTemplate>
                            <DataTemplate
x:DataType="models:WorkInterval">
                                <Frame Margin="2"
                                       BorderColor="#2B9DF5"
                                       BackgroundColor="{Binding
StatusColor}"
                                       Padding="0,8"
                                       CornerRadius="10"
                                       HasShadow="False">
                                    <Grid
ColumnDefinitions="*,*,*">
                                        <Label Text="{Binding
TimeRange}"
                                               FontSize="14"
                                               TextColor="White"
```

```
VerticalOptions="Center"
                                                    Grid.Column="0"
                                                    Margin="10,0,0,0"/>
                                            <Label Text="{Binding
Status}"
                                                    TextColor="White"

FontAttributes="Italic"
                                                    FontSize="14"

VerticalOptions="Center"
                                                    Grid.Column="1"/>
                                            <ActivityIndicator
                                                IsRunning="{Binding
IsBusy}"
                                                IsVisible="{Binding
IsBusy}"
                                                HeightRequest="16"
                                                WidthRequest="16"
                                                Color="#2B9DF5"
                                                Grid.Column="2"

VerticalOptions="Center"
                                                Margin="8,0,4,0"/>
                                        </Grid>
                                        <Frame.GestureRecognizers>
                                            <TapGestureRecognizer
Command="{Binding Source={x:Reference ThisPage},
Path=BindingContext.ToggleStatusCommand}"
CommandParameter="{Binding .}" />
                                        </Frame.GestureRecognizers>
                                    </Frame>
                                </DataTemplate>
                            </CollectionView.ItemTemplate>
                        </CollectionView>
                    </VerticalStackLayout>
                </toolkit:Expander>

                <!-- Вторник -->
                <toolkit:Expander IsExpanded="False"
BindingContext="{Binding Tuesday}"
ExpandedChanged="OnExpandedChanged">
                    <toolkit:Expander.Header>
                        <Frame BackgroundColor="#232323"
                               BorderColor="#2B9DF5"
                               CornerRadius="14"
                               Padding="0"
                               HasShadow="False"
                               Margin="0,10,0,0"
                               HorizontalOptions="Fill">
                            <Label Text="{Binding Day}"
                                   FontAttributes="Bold"
```

```
                                FontSize="18"
                                TextColor="#2B9DF5"
                                HorizontalOptions="Center"
                                VerticalOptions="Center"
                                Margin="12,8"/>
                    </Frame>
                </toolkit:Expander.Header>
                <VerticalStackLayout>
                    <ActivityIndicator IsRunning="{Binding
IsLoading}" IsVisible="{Binding IsLoading}" Margin="0,0,0,4"/>
                    <CollectionView ItemsSource="{Binding
Intervals}">
                        <CollectionView.ItemTemplate>
                            <DataTemplate
x:DataType="models:WorkInterval">
                                <Frame Margin="2"
                                       BorderColor="#2B9DF5"
                                       BackgroundColor="{Binding
StatusColor}"
                                       Padding="0,8"
                                       CornerRadius="10"
                                       HasShadow="False">
                                    <Grid
ColumnDefinitions="*,*,*">
                                        <Label Text="{Binding
TimeRange}"
                                               FontSize="14"
                                               TextColor="White"

VerticalOptions="Center"
                                               Grid.Column="0"
                                               Margin="10,0,0,0"/>
                                        <Label Text="{Binding
Status}"
                                               TextColor="White"

FontAttributes="Italic"
                                               FontSize="14"

VerticalOptions="Center"
                                               Grid.Column="1"/>
                                        <ActivityIndicator
                                            IsRunning="{Binding
IsBusy}"
                                            IsVisible="{Binding
IsBusy}"
                                            HeightRequest="16"
                                            WidthRequest="16"
                                            Color="#2B9DF5"
                                            Grid.Column="2"

VerticalOptions="Center"
                                            Margin="8,0,4,0"/>
```

```
                                    </Grid>
                                    <Frame.GestureRecognizers>
                                        <TapGestureRecognizer
Command="{Binding Source={x:Reference ThisPage},
Path=BindingContext.ToggleStatusCommand}"
CommandParameter="{Binding .}" />
                                    </Frame.GestureRecognizers>
                                </Frame>
                            </DataTemplate>
                        </CollectionView.ItemTemplate>
                    </CollectionView>
                </VerticalStackLayout>
            </toolkit:Expander>

            <!-- Среда -->
            <toolkit:Expander IsExpanded="False"
BindingContext="{Binding Wednesday}"
ExpandedChanged="OnExpandedChanged">
                <toolkit:Expander.Header>
                    <Frame BackgroundColor="#232323"
                           BorderColor="#2B9DF5"
                           CornerRadius="14"
                           Padding="0"
                           HasShadow="False"
                           Margin="0,10,0,0"
                           HorizontalOptions="Fill">
                        <Label Text="{Binding Day}"
                               FontAttributes="Bold"
                               FontSize="18"
                               TextColor="#2B9DF5"
                               HorizontalOptions="Center"
                               VerticalOptions="Center"
                               Margin="12,8"/>
                    </Frame>
                </toolkit:Expander.Header>
                <VerticalStackLayout>
                    <ActivityIndicator IsRunning="{Binding
IsLoading}" IsVisible="{Binding IsLoading}" Margin="0,0,0,4"/>
                    <CollectionView ItemsSource="{Binding
Intervals}">
                        <CollectionView.ItemTemplate>
                            <DataTemplate
x:DataType="models:WorkInterval">
                                <Frame Margin="2"
                                       BorderColor="#2B9DF5"
                                       BackgroundColor="{Binding
StatusColor}"
                                       Padding="0,8"
                                       CornerRadius="10"
                                       HasShadow="False">
                                    <Grid
ColumnDefinitions="*,*,*">
```

```
                                        <Label Text="{Binding
TimeRange}"
                                               FontSize="14"
                                               TextColor="White"

VerticalOptions="Center"
                                               Grid.Column="0"
                                               Margin="10,0,0,0"/>
                                        <Label Text="{Binding
Status}"
                                               TextColor="White"

FontAttributes="Italic"
                                               FontSize="14"

VerticalOptions="Center"
                                               Grid.Column="1"/>
                                        <ActivityIndicator
                                            IsRunning="{Binding
IsBusy}"
                                            IsVisible="{Binding
IsBusy}"
                                            HeightRequest="16"
                                            WidthRequest="16"
                                            Color="#2B9DF5"
                                            Grid.Column="2"

VerticalOptions="Center"
                                            Margin="8,0,4,0"/>
                                    </Grid>
                                    <Frame.GestureRecognizers>
                                        <TapGestureRecognizer
Command="{Binding Source={x:Reference ThisPage},
Path=BindingContext.ToggleStatusCommand}"
CommandParameter="{Binding .}" />
                                    </Frame.GestureRecognizers>
                                </Frame>
                            </DataTemplate>
                        </CollectionView.ItemTemplate>
                    </CollectionView>
                </VerticalStackLayout>
            </toolkit:Expander>

            <!-- Четверг -->
            <toolkit:Expander IsExpanded="False"
BindingContext="{Binding Thursday}"
ExpandedChanged="OnExpandedChanged">
                <toolkit:Expander.Header>
                    <Frame BackgroundColor="#232323"
                           BorderColor="#2B9DF5"
                           CornerRadius="14"
                           Padding="0"
                           HasShadow="False"
```

```
                            Margin="0,10,0,0"
                            HorizontalOptions="Fill">
                        <Label Text="{Binding Day}"
                               FontAttributes="Bold"
                               FontSize="18"
                               TextColor="#2B9DF5"
                               HorizontalOptions="Center"
                               VerticalOptions="Center"
                               Margin="12,8"/>
                    </Frame>
                </toolkit:Expander.Header>
                <VerticalStackLayout>
                    <ActivityIndicator IsRunning="{Binding
IsLoading}" IsVisible="{Binding IsLoading}" Margin="0,0,0,4"/>
                    <CollectionView ItemsSource="{Binding
Intervals}">
                        <CollectionView.ItemTemplate>
                            <DataTemplate
x:DataType="models:WorkInterval">
                                <Frame Margin="2"
                                       BorderColor="#2B9DF5"
                                       BackgroundColor="{Binding
StatusColor}"
                                       Padding="0,8"
                                       CornerRadius="10"
                                       HasShadow="False">
                                    <Grid
ColumnDefinitions="*,*,*">
                                        <Label Text="{Binding
TimeRange}"
                                               FontSize="14"
                                               TextColor="White"

VerticalOptions="Center"
                                               Grid.Column="0"
                                               Margin="10,0,0,0"/>
                                        <Label Text="{Binding
Status}"
                                               TextColor="White"

FontAttributes="Italic"
                                               FontSize="14"

VerticalOptions="Center"
                                               Grid.Column="1"/>
                                        <ActivityIndicator
                                            IsRunning="{Binding
IsBusy}"
                                            IsVisible="{Binding
IsBusy}"
                                            HeightRequest="16"
                                            WidthRequest="16"
                                            Color="#2B9DF5"
```

```
                                        Grid.Column="2"
VerticalOptions="Center"
                                        Margin="8,0,4,0"/>
                                </Grid>
                                <Frame.GestureRecognizers>
                                    <TapGestureRecognizer
Command="{Binding Source={x:Reference ThisPage},
Path=BindingContext.ToggleStatusCommand}"
CommandParameter="{Binding .}" />
                                </Frame.GestureRecognizers>
                            </Frame>
                        </DataTemplate>
                    </CollectionView.ItemTemplate>
                </CollectionView>
            </VerticalStackLayout>
        </toolkit:Expander>

        <!-- Пятница -->
        <toolkit:Expander IsExpanded="False"
BindingContext="{Binding Friday}"
ExpandedChanged="OnExpandedChanged">
            <toolkit:Expander.Header>
                <Frame BackgroundColor="#232323"
                       BorderColor="#2B9DF5"
                       CornerRadius="14"
                       Padding="0"
                       HasShadow="False"
                       Margin="0,10,0,0"
                       HorizontalOptions="Fill">
                    <Label Text="{Binding Day}"
                           FontAttributes="Bold"
                           FontSize="18"
                           TextColor="#2B9DF5"
                           HorizontalOptions="Center"
                           VerticalOptions="Center"
                           Margin="12,8"/>
                </Frame>
            </toolkit:Expander.Header>
            <VerticalStackLayout>
                <ActivityIndicator IsRunning="{Binding
IsLoading}" IsVisible="{Binding IsLoading}" Margin="0,0,0,4"/>
                <CollectionView ItemsSource="{Binding
Intervals}">
                    <CollectionView.ItemTemplate>
                        <DataTemplate
x:DataType="models:WorkInterval">
                            <Frame Margin="2"
                                   BorderColor="#2B9DF5"
                                   BackgroundColor="{Binding
StatusColor}"
                                   Padding="0,8"
                                   CornerRadius="10"
```

```
                                        HasShadow="False">
                                    <Grid
ColumnDefinitions="*,*,*">
                                        <Label Text="{Binding
TimeRange}"
                                               FontSize="14"
                                               TextColor="White"

VerticalOptions="Center"
                                               Grid.Column="0"
                                               Margin="10,0,0,0"/>
                                        <Label Text="{Binding
Status}"
                                               TextColor="White"

FontAttributes="Italic"
                                               FontSize="14"

VerticalOptions="Center"
                                               Grid.Column="1"/>
                                        <ActivityIndicator
                                            IsRunning="{Binding
IsBusy}"
                                            IsVisible="{Binding
IsBusy}"
                                            HeightRequest="16"
                                            WidthRequest="16"
                                            Color="#2B9DF5"
                                            Grid.Column="2"

VerticalOptions="Center"
                                            Margin="8,0,4,0"/>
                                    </Grid>
                                    <Frame.GestureRecognizers>
                                        <TapGestureRecognizer
Command="{Binding Source={x:Reference ThisPage},
Path=BindingContext.ToggleStatusCommand}"
CommandParameter="{Binding .}" />
                                    </Frame.GestureRecognizers>
                                </Frame>
                            </DataTemplate>
                        </CollectionView.ItemTemplate>
                    </CollectionView>
                </VerticalStackLayout>
            </toolkit:Expander>

            <!-- Суббота -->
            <toolkit:Expander IsExpanded="False"
BindingContext="{Binding Saturday}"
ExpandedChanged="OnExpandedChanged">
                <toolkit:Expander.Header>
                    <Frame BackgroundColor="#232323"
                           BorderColor="#2B9DF5"
```

```
                            CornerRadius="14"
                            Padding="0"
                            HasShadow="False"
                            Margin="0,10,0,0"
                            HorizontalOptions="Fill">
                        <Label Text="{Binding Day}"
                               FontAttributes="Bold"
                               FontSize="18"
                               TextColor="#2B9DF5"
                               HorizontalOptions="Center"
                               VerticalOptions="Center"
                               Margin="12,8"/>
                    </Frame>
                </toolkit:Expander.Header>
                <VerticalStackLayout>
                    <ActivityIndicator IsRunning="{Binding
IsLoading}" IsVisible="{Binding IsLoading}" Margin="0,0,0,4"/>
                    <CollectionView ItemsSource="{Binding
Intervals}">
                        <CollectionView.ItemTemplate>
                            <DataTemplate
x:DataType="models:WorkInterval">
                                <Frame Margin="2"
                                       BorderColor="#2B9DF5"
                                       BackgroundColor="{Binding
StatusColor}"
                                       Padding="0,8"
                                       CornerRadius="10"
                                       HasShadow="False">
                                    <Grid
ColumnDefinitions="*,*,*">
                                        <Label Text="{Binding
TimeRange}"
                                               FontSize="14"
                                               TextColor="White"

VerticalOptions="Center"
                                               Grid.Column="0"
                                               Margin="10,0,0,0"/>
                                        <Label Text="{Binding
Status}"
                                               TextColor="White"

FontAttributes="Italic"
                                               FontSize="14"

VerticalOptions="Center"
                                               Grid.Column="1"/>
                                        <ActivityIndicator
                                            IsRunning="{Binding
IsBusy}"
                                            IsVisible="{Binding
IsBusy}"
```

```
                                                HeightRequest="16"
                                                WidthRequest="16"
                                                Color="#2B9DF5"
                                                Grid.Column="2"
VerticalOptions="Center"
                                                Margin="8,0,4,0"/>
                                        </Grid>
                                        <Frame.GestureRecognizers>
                                            <TapGestureRecognizer
Command="{Binding Source={x:Reference ThisPage},
Path=BindingContext.ToggleStatusCommand}"
CommandParameter="{Binding .}" />
                                        </Frame.GestureRecognizers>
                                    </Frame>
                                </DataTemplate>
                            </CollectionView.ItemTemplate>
                        </CollectionView>
                    </VerticalStackLayout>
                </toolkit:Expander>

                <!-- Воскресенье -->
                <toolkit:Expander IsExpanded="False"
BindingContext="{Binding Sunday}"
ExpandedChanged="OnExpandedChanged">
                    <toolkit:Expander.Header>
                        <Frame BackgroundColor="#232323"
                               BorderColor="#2B9DF5"
                               CornerRadius="14"
                               Padding="0"
                               HasShadow="False"
                               Margin="0,10,0,0"
                               HorizontalOptions="Fill">
                            <Label Text="{Binding Day}"
                                   FontAttributes="Bold"
                                   FontSize="18"
                                   TextColor="#2B9DF5"
                                   HorizontalOptions="Center"
                                   VerticalOptions="Center"
                                   Margin="12,8"/>
                        </Frame>
                    </toolkit:Expander.Header>
                    <VerticalStackLayout>
                        <ActivityIndicator IsRunning="{Binding
IsLoading}" IsVisible="{Binding IsLoading}" Margin="0,0,0,4"/>
                        <CollectionView ItemsSource="{Binding
Intervals}">
                            <CollectionView.ItemTemplate>
                                <DataTemplate
x:DataType="models:WorkInterval">
                                    <Frame Margin="2"
                                           BorderColor="#2B9DF5"
```

```
                                        BackgroundColor="{Binding
StatusColor}"
                                        Padding="0,8"
                                        CornerRadius="10"
                                        HasShadow="False">
                                    <Grid
ColumnDefinitions="*,*,*">
                                        <Label Text="{Binding
TimeRange}"
                                               FontSize="14"
                                               TextColor="White"

VerticalOptions="Center"
                                               Grid.Column="0"
                                               Margin="10,0,0,0"/>
                                        <Label Text="{Binding
Status}"
                                               TextColor="White"

FontAttributes="Italic"
                                               FontSize="14"

VerticalOptions="Center"
                                               Grid.Column="1"/>
                                        <ActivityIndicator
                                            IsRunning="{Binding
IsBusy}"
                                            IsVisible="{Binding
IsBusy}"

                                            HeightRequest="16"
                                            WidthRequest="16"
                                            Color="#2B9DF5"
                                            Grid.Column="2"

VerticalOptions="Center"
                                            Margin="8,0,4,0"/>
                                    </Grid>
                                    <Frame.GestureRecognizers>
                                        <TapGestureRecognizer
Command="{Binding Source={x:Reference ThisPage},
Path=BindingContext.ToggleStatusCommand}"
CommandParameter="{Binding .}" />

                                    </Frame.GestureRecognizers>
                                </Frame>
                            </DataTemplate>
                        </CollectionView.ItemTemplate>
                    </CollectionView>
                </VerticalStackLayout>
            </toolkit:Expander>
</ContentPage>
```

Листинг Г.10 – Файл MasterProfilePage.xaml

```
<ContentPage
    Title="Профиль мастера"

    <ScrollView>

            <Frame
                   HasShadow="True"
                   Margin="0,0,0,12">
                <VerticalStackLayout Spacing="12">
                    <Label Text="ФИО:" FontAttributes="Bold"
TextColor="#2B9DF5"/>
                    <Label Text="{Binding FullName}" FontSize="17"
TextColor="White"/>
                    <Label Text="Email:" FontAttributes="Bold"
</ContentPage>
```

Листинг Г.11 – Файл MasterProfilePage.xaml.cs

```
    public partial class MasterProfilePage : ContentPage
    {

        public MasterProfilePage()
        {
            BindingContext = new MasterProfileViewModel();
        }
    }
```

Листинг Г.12 – Файл MasterMainPage.xaml

```
<ContentPage

    <VerticalStackLayout Padding="22" Spacing="18">

        <Label Text="Ваши записи"
               FontSize="26"
               FontAttributes="Bold"
               TextColor="#2B9DF5"
               HorizontalOptions="Center"
               Margin="0,0,0,12"/>

        <DatePicker Date="{Binding SelectedDate}"
                    TextColor="#2B9DF5"
                    FontSize="16"
                    Margin="0,0,0,10"/>

        <CollectionView ItemsSource="{Binding Appointments}"
VerticalOptions="FillAndExpand">
            <CollectionView.ItemTemplate>
                <DataTemplate
x:DataType="models:AppointmentModel">
       />
```

Листинг Г.13 – Файл ManageServicesPage.xaml

```
<ContentPage
             Title="Услуги">
    <ScrollView>
        <
                    BackgroundColor="#232323"
BorderColor="#2B9DF5" BorderWidth="1"
                    TextColor="#2B9DF5" FontAttributes="Bold"
CornerRadius="14" HeightRequest="44"/>

            <BoxView HeightRequest="1" Color="#2B9DF5"
Margin="0,10,0,10"/>
                    <Entry Placeholder="Длительность (минут)"
TextColor="White" Text="{Binding EditServiceDurationMinutes}"
Keyboard="Numeric" BackgroundColor="Transparent"/>
                    <HorizontalStackLayout Spacing="10">
                        <Button Text="Сохранить" Command="{Binding
SaveEditServiceCommand}" BackgroundColor="#2B9DF5"
TextColor="White" CornerRadius="12"/>
                        <Button Text="Отмена" Command="{Binding
CancelEditServiceCommand}" BackgroundColor="#232323"
TextColor="White" CornerRadius="12"/>
                    </HorizontalStackLayout>
                </VerticalStackLayout>
            </Frame>
            <!-- Список услуг -->
            <CollectionView ItemsSource="{Binding Services}">
                <CollectionView.ItemTemplate>
                    <DataTemplate
x:DataType="models:ServiceModel">
                        <Frame BackgroundColor="#232323"
BorderColor="#2B9DF5" CornerRadius="14" Padding="12" Margin="0,5">
                            <Grid>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="*"/>
                                    <ColumnDefinition Width="36"/>
                                    <ColumnDefinition Width="36"/>
                                </Grid.ColumnDefinitions>
                                <!-- Инфо -->
                                <VerticalStackLayout
Grid.Column="0">
                                    <Label Text="{Binding Name}"
TextColor="White" FontSize="18" FontAttributes="Bold"/>
                                    <Label Text="{Binding
Description}" TextColor="#aaa" FontSize="14"/>
                                    <Label Text="{Binding Price,
StringFormat='Цена: {0} ₽'}" TextColor="#2B9DF5" FontSize="14"/>
                                    <Label Text="{Binding
DurationMinutes, StringFormat='Длительность: {0} мин.'}"
TextColor="#aaa" FontSize="14"/>
                                </VerticalStackLayout>
                    </DataTemplate>
```

```
                </CollectionView.ItemTemplate>
            </CollectionView>



        </VerticalStackLayout>
    </ScrollView>
</ContentPage>
```

Листинг Г.14 – Файл ManageServicesPage.xaml.cs

```
    public partial class ManageServicesPage : ContentPage
    {
        public ManageServicesPage()
        {
            BindingContext = new ManageServicesViewModel();
        }
    }
```

Листинг Г.15 – Файл ManageMastersPage.xaml

```
<ContentPage
             Title="Мастера">

    <ScrollView>
        <VerticalStackLayout Padding="22" Spacing="14">

            <!-- Блок редактирования -->
            <Frame BackgroundColor="#181a20" BorderColor="#2B9DF5"
CornerRadius="14" Padding="14" Margin="0,10,0,0"
       IsVisible="{Binding EditingMaster,
Converter={StaticResource NullToBoolConverter}}">
                <VerticalStackLayout Spacing="8">

                        <Button Text="Отмена" Command="{Binding
CancelEditMasterCommand}" BackgroundColor="#232323"
TextColor="White" CornerRadius="12"/>
                    </HorizontalStackLayout>
                </VerticalStackLayout>
            </Frame>
            <!-- Список мастеров -->
            <CollectionView ItemsSource="{Binding Masters}">
                <CollectionView.ItemTemplate>
                    <DataTemplate x:DataType="models:MasterModel">
                        <Frame >
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="*"/>
                                    <ColumnDefinition Width="36"/>
                                    <ColumnDefinition Width="36"/>
                                </Grid.ColumnDefinitions>

                                </Frame>
                </CollectionView.ItemTemplate>
  </CollectionView>
```

Листинг Г.16 – Файл ManageMastersPage.xaml.cs

```
using AutoServiceMobile.ViewModels;

namespace AutoServiceMobile.Views
{
    public partial class ManageMastersPage : ContentPage
    {
        public ManageMastersPage()
        {
            BindingContext = new ManageMastersViewModel();
        }
    }
}
```

Листинг Г.17 – Файл LoginPage.xaml

```
<ContentPage
    Title=""
    Shell.NavBarIsVisible="False">

    <VerticalStackLayout Padding="32,80,32,0" Spacing="28">

        <Label Text="Вход"
               FontSize="28"
               FontAttributes="Bold"
               TextColor="#2B9DF5"
               HorizontalOptions="Center"
               Margin="0,0,0,34"/>


        <!-- Войти -->
        <Button Text="Войти"
                Command="{Binding LoginCommand}"
                BackgroundColor="#232323"
                BorderColor="#2B9DF5"
                BorderWidth="1"
                TextColor="#2B9DF5"
                FontAttributes="Bold"
                CornerRadius="14"
                HeightRequest="48"
                FontSize="18"
                Margin="0,20,0,0"/>

        <!-- Регистрация -->
        <Button Text="Регистрация"
                Command="{Binding GoToRegisterCommand}"
                BackgroundColor="Transparent"
                BorderColor="Transparent"
                BorderWidth="0"
                TextColor="#2B9DF5"
                FontAttributes="Bold"
                FontSize="16"
                Margin="0,8,0,0"/>
```

```
    </VerticalStackLayout>
</ContentPage>
```

Листинг Г.18 – Файл LoginPage.xaml.cs

```
using AutoServiceMobile.ViewModels;

namespace AutoServiceMobile.Views;

public partial class LoginPage : ContentPage
{
     public LoginPage()
     {

         BindingContext = new LoginViewModel();
    }
}
```

Листинг Г.19 – Файл GaragePage.xaml

```
<ContentPage
    Title="Мой гараж"
    >

    <VerticalStackLayout Padding="18" Spacing="18">
        <!-- Список автомобилей -->
        <CollectionView ItemsSource="{Binding Cars}"
SelectionMode="None" MinimumHeightRequest="100"
VerticalOptions="FillAndExpand">
            <CollectionView.ItemsLayout>
                <LinearItemsLayout Orientation="Vertical"
ItemSpacing="12"/>
            </CollectionView.ItemsLayout>
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <Frame BackgroundColor="#232323"
CornerRadius="14" BorderColor="#2B9DF5" Padding="14"
HasShadow="True">
                        <Grid ColumnDefinitions="auto,*,auto">
                            <!-- Иконка -->
                            <Image Source="{Binding ImagePath}"
HeightRequest="56" WidthRequest="56" VerticalOptions="Center"/

        <Button Text="Добавить автомобиль"
                Command="{Binding AddCarCommand}"
                BackgroundColor="#232323"
                BorderColor="#2B9DF5"
                BorderWidth="1"
                TextColor="#2B9DF5"
                />
```

Листинг Г.20 – Файл GaragePage.xaml.cs

```
using AutoServiceMobile.ViewModels;
```

```
namespace AutoServiceMobile.Views;

public partial class GaragePage : ContentPage
{
    public GaragePage()
    {
        InitializeComponent();
        BindingContext = new GarageViewModel();
    }

}
```

Листинг Г.21 – Файл ClientServiceHistoryPage.xaml

```
<ContentPage
        Title="История обслуживания"

    <VerticalStackLayout Padding="18" Spacing="18">

        <CollectionView ItemsSource="{Binding History}"
SelectionMode="None">
            <CollectionView.ItemsLayout>
                <LinearItemsLayout Orientation="Vertical"
ItemSpacing="10"/>
            </CollectionView.ItemsLayout>

                        <Frame.GestureRecognizers>
                            <TapGestureRecognizer
                                Command="{Binding
Source={x:Reference ThisPage},
Path=BindingContext.OpenDetailCommand}"
                                CommandParameter="{Binding .}" />
                        </Frame.GestureRecognizers>

</ContentPage>
```

Листинг Г.22 – Файл ClientServiceHistoryPage.xaml.cs

```
    public partial class ClientServiceHistoryPage : ContentPage
    {
        public ClientServiceHistoryPage()
        {
            BindingContext = new ClientServiceHistoryViewModel();
        }
    }
```

Листинг Г.23 – Файл ClientServiceHistoryDetailPage.xaml.cs

```
using AutoServiceMobile.ViewModels;

namespace AutoServiceMobile.Views
{
    public partial class ClientServiceHistoryDetailPage :
ContentPage
```

```
    {
        public ClientServiceHistoryDetailPage()
        {
            InitializeComponent();
            BindingContext = new
ClientServiceHistoryDetailViewModel();
        }
    }
}
```

Листинг Г.24 – Файл ClientProfilePage.xaml

```
<ContentPage
        Title="Профиль">

    <VerticalStackLayout Padding="18" Spacing="18"
VerticalOptions="FillAndExpand">

        <Frame Style="{StaticResource CardFrameStyle}">
            <VerticalStackLayout Spacing="10">
                <Label Text="ФИО:" TextColor="#2B9DF5"
FontAttributes="Bold" FontSize="16"/>
                <Label Text="{Binding FullName}" TextColor="White"
FontSize="16"/>
                <BoxView HeightRequest="1" Color="#2B9DF5"
Margin="0,2"/>

                <Label Text="Email:" TextColor="#2B9DF5"
FontAttributes="Bold" FontSize="16"/>
                <Label Text="{Binding Email}" TextColor="White"
FontSize="16"/>
                <BoxView HeightRequest="1" Color="#2B9DF5"
Margin="0,2"/>

                <Label Text="Телефон:" TextColor="#2B9DF5"
FontAttributes="Bold" FontSize="16"/>
                <Label Text="{Binding Phone}" TextColor="White"
FontSize="16"/>
            </VerticalStackLayout>
        </Frame>

        <Grid RowDefinitions="Auto,*,Auto" Padding="18">
            <BoxView Grid.Row="0" HeightRequest="230"
BackgroundColor="Black"/>

        <Button Text="Выйти"
                Command="{Binding LogoutCommand}"
                BackgroundColor="#232323"
                BorderColor="Red"
                BorderWidth="1"
                TextColor="Red"
                FontAttributes="Bold"
                CornerRadius="14"
                />
```

Листинг Г.25 – Файл ClientProfilePage.xaml.cs

```
using AutoServiceMobile.ViewModels;

namespace AutoServiceMobile.Views
{
    public partial class ClientProfilePage : ContentPage
    {
        public ClientProfilePage()
        {
            InitializeComponent();
            BindingContext = new ClientProfileViewModel();
        }
    }
}
```

Листинг Г.26 – Файл ClientAppointmentsPage.xaml

```
<ContentPage
    Title="Мои записи"
    x:Name="ThisPage">

    <VerticalStackLayout Padding="18" Spacing="18">

        <CollectionView ItemsSource="{Binding Appointments}"
SelectionMode="None">
                <DataTemplate x:DataType="models:AppointmentItem">
                    <Frame Style="{StaticResource CardFrameStyle}"
Margin="0,0,0,2">
                        <!-- ВЕСЬ Frame кликабельный -->
                        <Frame.GestureRecognizers>
```

Листинг Г.27 – Файл ClientAppointmentDetailPage.xaml

```
<ContentPage
    Title="Детали записи">

    <VerticalStackLayout Padding="18" Spacing="20">



        <Frame Style="{StaticResource CardFrameStyle}">
            <VerticalStackLayout Spacing="8">
                <Label Text="{Binding Appointment.ServiceName,
StringFormat='Услуга: {0}'}" TextColor="White" FontSize="17"/>
                <Label Text="{Binding Appointment.DateTime,
StringFormat='Дата: {0:dd.MM.yyyy HH:mm}'}" TextColor="White"
FontSize="17"/>
                <Label Text="{Binding Appointment.Status,
StringFormat='Статус: {0}'}" TextColor="#2B9DF5" FontSize="17"
FontAttributes="Bold"/>
                        <Label Text="{Binding}" TextColor="#aaa"
FontSize="14"/>
                    </Frame>
```

```
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>
    </VerticalStackLayout>
</ContentPage>
```

Листинг Г.28 – Файл ClientAppointmentDetailPage.xaml.cs

```
    public partial class ClientAppointmentDetailPage : ContentPage
    {
        public ClientAppointmentDetailPage()
        {
            InitializeComponent();
            BindingContext = new
ClientAppointmentDetailViewModel();
        }
    }
}
```

Листинг Г.29 – Файл AppointmentDetailPage.xaml

```
<ContentPage
    Title="Детали заказа">

    <VerticalStackLayout Padding="22" Spacing="22">


        <Frame BackgroundColor="#232323" BorderColor="#2B9DF5"
CornerRadius="14" Padding="16" HasShadow="True">
            <VerticalStackLayout Spacing="9">
                <Label Text="{Binding Appointment.Time,
StringFormat='Время: {0}'}" TextColor="White" FontSize="17"/>
                <Label Text="{Binding Appointment.ServiceType,
StringFormat='Услуга: {0}'}" TextColor="White" FontSize="17"/>
                <Label Text="{Binding Appointment.Car,
StringFormat='Автомобиль: {0}'}" TextColor="White" FontSize="17"/>
            </VerticalStackLayout>
        </Frame>

        <Button Text="Отправить уведомление"
                Command="{Binding SendMessageCommand}"
                BackgroundColor="#232323"
                BorderColor="#2B9DF5"
                BorderWidth="1"
                TextColor="#2B9DF5"
                FontAttributes="Bold"
                CornerRadius="14"
                HeightRequest="48"/>
    </VerticalStackLayout>
</ContentPage>
```

Г.30 – Файл AdminStatisticsPage.xaml

```
<
x:Class="AutoServiceMobile.Views.AdminStatisticsPage"
```

```
             x:DataType="viewModels:AdminStatisticsViewModel"
             Title="Статистика"

    <ScrollView>
        <VerticalStackLayout Padding="18" Spacing="15">

            <Label Text="Статистика автосервиса" FontSize="26"
FontAttributes="Bold" HorizontalOptions="Center"
Margin="0,20,0,20"/>

                        <Label Text="Выручка" TextColor="Black"
FontSize="14"/>
                        <Label Text="{Binding TotalRevenue,
StringFormat='{0:N0} ₽'}" FontSize="20" FontAttributes="Bold"
TextColor="#952BF5"/>
                    </VerticalStackLayout>
                </Frame>
                <Frame Grid.Column="0" Grid.Row="1" Padding="14"
CornerRadius="14" BackgroundColor="#fff8ec">
                    <VerticalStackLayout Spacing="3">
                        <Label Text="Средний чек"
TextColor="Black" FontSize="14"/>
                        <Label Text="{Binding AverageCheck,
StringFormat='{0:N0} ₽'}" FontSize="20" FontAttributes="Bold"
TextColor="#F59B2B"/>
                    </VerticalStackLayout>
                </Frame>


            <!-- Топ мастера -->
            <Label Text="Топ-3 мастера" FontAttributes="Bold"
FontSize="18" Margin="0,15,0,5"/>
            <CollectionView ItemsSource="{Binding TopMasters}"
SelectionMode="None">
                <CollectionView.ItemTemplate>
                    <DataTemplate
x:DataType="viewModels:MasterStat">
                        <Grid ColumnDefinitions="*,auto"
Padding="6">

                            <Label Text="{Binding Name}"
FontSize="15" TextColor="White"/>

                            <Label Grid.Column="1" Text="{Binding
Count, StringFormat='×{0}'}" FontSize="15" TextColor="#28B153"/>
                        </Grid>
                    </DataTemplate>
                </CollectionView.ItemTemplate>
            </CollectionView>
        </VerticalStackLayout>
    </ScrollView>
</ContentPage>
```

Листинг Г.31 – Файл AdminProfilePage.xaml

```
<ContentPage
    Title="Профиль">

    <VerticalStackLayout Padding="18" Spacing="18"
VerticalOptions="FillAndExpand">

        <Frame Style="{StaticResource CardFrameStyle}">
            <VerticalStackLayout Spacing="10">
                <Label Text="ФИО:" TextColor="#2B9DF5"
FontAttributes="Bold" FontSize="16"/>
                <Label Text="{Binding FullName}" TextColor="White"
FontSize="16"/>
                <BoxView HeightRequest="1" Color="#2B9DF5"
Margin="0,2"/>

                <Label Text="Email:" TextColor="#2B9DF5"
FontAttributes="Bold" FontSize="16"/>
                <Label Text="{Binding Email}" TextColor="White"
FontSize="16"/>
                <BoxView HeightRequest="1" Color="#2B9DF5"
Margin="0,2"/>

                <Label Text="Телефон:" TextColor="#2B9DF5"
FontAttributes="Bold" FontSize="16"/>
                <Label Text="{Binding Phone}" TextColor="White"
FontSize="16"/>
            </VerticalStackLayout>
        </Frame>
            Command="{Binding LogoutCommand}"
            BackgroundColor="#232323"
            BorderColor="Red"
            BorderWidth="1"
            TextColor="Red"
            FontAttributes="Bold"
            CornerRadius="14">
```

Листинг Г.32 – Файл AddCarPage.xaml

```
<ContentPage
    Title="Добавить автомобиль"
    x:Name="ThisPage">

    <ScrollView>
        <VerticalStackLayout Padding="24,24,24,24" Spacing="26">

            <Button Text="Сохранить"
                    Command="{Binding SaveCommand}"
                    BackgroundColor="#232323"
                    BorderColor="#2B9DF5"
                    BorderWidth="1"
                    TextColor="#2B9DF5"
```

```
                    FontAttributes="Bold"
                    CornerRadius="14"
                    HeightRequest="48"
                    FontSize="18"
                    Margin="0,8"/>
        </VerticalStackLayout>
    </ScrollView>
</ContentPage>
```

Листинг Г.33 – Файл Step4_Confirmation.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
        <Label Text="{Binding BookingVM.SelectedService,
StringFormat='Услуга: {0}'}"
               FontSize="17" TextColor="White" Margin="16,0"/>
        <BoxView HeightRequest="1" Color="White" Margin="16,8"/>

        <Label Text="{Binding BookingVM.SelectedCar,
StringFormat='Автомобиль: {0}'}"
               FontSize="17" TextColor="White" Margin="16,0"/>
        <BoxView HeightRequest="1" Color="White" Margin="16,8"/>

        <Label Text="{Binding BookingVM.SelectedMaster,
StringFormat='Мастер: {0}'}"
               FontSize="17" TextColor="White" Margin="16,0"/>
        <BoxView HeightRequest="1" Color="White" Margin="16,8"/>

        <Label Text="{Binding BookingVM.SelectedTime,
StringFormat='Время: {0}'}"
               FontSize="17" TextColor="White" Margin="16,0"/>
        <BoxView HeightRequest="1" Color="White" Margin="16,8"/>

        <Button Text="Подтвердить"
                BackgroundColor="Transparent"
                TextColor="#28B153"
                BorderColor="#28B153"
                BorderWidth="1"
                CornerRadius="14"
                FontSize="18"
                FontAttributes="Bold"
                Margin="16,32,16,0"
                Command="{Binding ConfirmCommand}"/>
    </VerticalStackLayout>
</ContentView>
```

Листинг Г.34 – Файл Step4_Confirmation.xaml.cs

```
using AutoServiceMobile.ViewModels;

namespace AutoServiceMobile.Views.Partials
{
    public partial class Step4_Confirmation : ContentView
    {
        public Step4_Confirmation(ServiceBookingViewModel vm)
```

```
        {
            InitializeComponent();
            BindingContext = new Step4_ConfirmationViewModel(vm);
        }
    }
}
```

Листинг Г.35 – Файл Step3_TimeAndMaster.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentView
    x:Class="AutoServiceMobile.Views.Partials.Step3_TimeAndMaster"
    BackgroundColor="Black">

    <VerticalStackLayout Padding="0,16" Spacing="20">
        <Label Text="Выберите мастера" FontSize="22"
FontAttributes="Bold" TextColor="#2B9DF5" Margin="16,0,0,12"/>
        <Picker ItemsSource="{Binding Masters}"
SelectedItem="{Binding SelectedMaster}" Title="Выберите мастера"
TitleColor="White" ItemDisplayBinding="{Binding FullName}"
TextColor="White"/>

        <Label Text="Выберите время" FontSize="18"
TextColor="#2B9DF5" Margin="16,18,0,6"/>

                        <Label Text="{Binding}" TextColor="White"
FontSize="17"/>
                        <Image Source="check_green.png"
WidthRequest="20" HeightRequest="20"
                               HorizontalOptions="End"
                               IsVisible="{Binding IsSelected}"
                               Grid.Column="1"/>

                        <BoxView Grid.Row="1" HeightRequest="1"
Color="#2B9DF5" Margin="0,6"/>
                    </Grid>

                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>
    </VerticalStackLayout>
</ContentView>
```

Листинг Г.36 – Файл Step3_TimeAndMaster.xaml.cs

```
using AutoServiceMobile.ViewModels;

namespace AutoServiceMobile.Views.Partials
{
    public partial class Step3_TimeAndMaster : ContentView
    {
        public Step3_TimeAndMaster(ServiceBookingViewModel vm)
        {
            InitializeComponent();
```

```
            BindingContext = new Step3_TimeAndMasterViewModel(vm);
        }
    }
}
```

Листинг Г.37 – Файл Step2_CarSelection.xaml.cs

```
namespace AutoServiceMobile.Views.Partials
{

            InitializeComponent();
            BindingContext = new Step2_CarSelectionViewModel(vm);
        }
    }
}
```

Листинг Г.38 – Файл Step1_ServiceSelection.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentView

x:Class="AutoServiceMobile.Views.Partials.Step1_ServiceSelection"
    BackgroundColor="Black">

    <VerticalStackLayout Padding="0,16"
BackgroundColor="Transparent" Spacing="0">
        <Label Text="Выберите услугу"
               FontSize="22"
               FontAttributes="Bold"
               TextColor="#2B9DF5"
               Margin="16,0,0,12"/>
                           <Label Text="{Binding DurationMinutes,
StringFormat='Длительность: {0} мин.'}" FontSize="14"
TextColor="#aaa"/>
                       </VerticalStackLayout>
```

Листинг Г.39 – Файл Step1_ServiceSelection.cs

```
    public partial class Step1_ServiceSelection : ContentView
    {
        public Step1_ServiceSelection(ServiceBookingViewModel vm)
        {
            InitializeComponent();
            BindingContext = new
Step1_ServiceSelectionViewModel(vm);
        }
    }
```

Листинг Г.40 – Файл Step4_ConfirmationViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
```

```
    public partial class Step4_ConfirmationViewModel :
ObservableObject
    {
        public ServiceBookingViewModel BookingVM { get; }

        public Step4_ConfirmationViewModel(ServiceBookingViewModel
vm)
        {
            BookingVM = vm;
            ConfirmCommand = new AsyncRelayCommand(OnConfirm);
        }

        public AsyncRelayCommand ConfirmCommand { get; }

        private async Task OnConfirm()
        {
            await App.Current.MainPage.DisplayAlert("Запись
подтверждена!", "Ваша запись успешно оформлена.", "Ок");
            await Shell.Current.GoToAsync("//ClientMainPage");
        }
    }
}
```

Листинг Г.41 – Файл Step3_TimeAndMasterViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public partial class Step3_TimeAndMasterViewModel :
ObservableObject
    {
        private readonly ServiceBookingViewModel bookingVM;
        [ObservableProperty] ObservableCollection<MasterModel>
masters;
        [ObservableProperty] MasterModel selectedMaster;
        [ObservableProperty] ObservableCollection<string>
availableTimes;
        [ObservableProperty] string selectedTime;

        public
Step3_TimeAndMasterViewModel(ServiceBookingViewModel vm)
        {
            bookingVM = vm;

        }

        partial void OnSelectedMasterChanged(MasterModel value)
        {
            AvailableTimes = value != null
                ? new ObservableCollection<string> { "10:00",
"11:00", "12:00", "13:00" }
                : null;
            SelectedTime = null;
        }
```

```
        partial void OnSelectedTimeChanged(string value)
        {
            bookingVM.SelectedMaster = SelectedMaster?.FullName;
            bookingVM.SelectedTime = value;
        }
    }
}
```

Листинг Г.42 – Файл Step2_CarSelectionViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public partial class Step2_CarSelectionViewModel :
ObservableObject
    {
        private readonly ServiceBookingViewModel bookingVM;
        [ObservableProperty] ObservableCollection<CarModel> cars;
        [ObservableProperty] CarModel selectedCar;
        [ObservableProperty] string newCarName;

        public IRelayCommand AddCarCommand { get; }

        public Step2_CarSelectionViewModel(ServiceBookingViewModel
vm)
        {
            bookingVM = vm;
            Cars = new ObservableCollection<CarModel>
        {
            new CarModel { Name = "Toyota Camry", ImagePath =
"car_icon.png" },
            new CarModel { Name = "Mazda CX-5", ImagePath =
"car_icon.png" }
        };
            AddCarCommand = new RelayCommand(AddCar);
        }

        partial void OnSelectedCarChanged(CarModel value)
        {
            bookingVM.SelectedCar = value?.Name;
        }

        private void AddCar()
        {
            if (!string.IsNullOrWhiteSpace(NewCarName))
            {
                var car = new CarModel { Name = NewCarName,
ImagePath = "car_icon.png" };
                Cars.Add(car);
                SelectedCar = car;
                NewCarName = "";
            }
        }
    }
}
```

Листинг Г.43 – Файл Step1_ServiceSelectionViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{

    public partial class Step1_ServiceSelectionViewModel :
ObservableObject
    {
        private readonly ServiceBookingViewModel bookingVM;
        [ObservableProperty] ObservableCollection<ServiceModel>
services;
        [ObservableProperty] ServiceModel selectedServiceModel;

        public
Step1_ServiceSelectionViewModel(ServiceBookingViewModel vm)
        {
            bookingVM = vm;
            Services = new ObservableCollection<ServiceModel>
        {
            new ServiceModel { Name = "Замена масла", Description
= "Быстрая замена масла", Price = 1200, DurationMinutes = 30 },
            new ServiceModel { Name = "Диагностика", Description =
"Комплексная диагностика", Price = 1500, DurationMinutes = 45 }
        };
        }

        partial void OnSelectedServiceModelChanged(ServiceModel
value)
        {
            foreach (var s in Services)
                s.IsSelected = (s == value);
            bookingVM.SelectedService = value?.Name;
            // Можно сохранить и другие параметры, если
потребуется!
        }
    }


}
```

Листинг Г.44 – Файл ServiceBookingViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public partial class ServiceBookingViewModel :
ObservableObject
    {
        [ObservableProperty] private int currentStep = 1;

        public Color Step1Color => GetColorForStep(1);
        public Color Step2Color => GetColorForStep(2);
        public Color Step3Color => GetColorForStep(3);
        public Color Step4Color => GetColorForStep(4);
```

```
        public bool CanGoToStep2 => CurrentStep > 1;
        public bool CanGoToStep3 => CurrentStep > 2;
        public bool CanGoToStep4 => CurrentStep > 3;

        public IRelayCommand<object> GoToStepCommand { get; }

        public ServiceBookingViewModel()
        {
            GoToStepCommand = new
RelayCommand<object>(OnGoToStep);
        }

        private Color GetColorForStep(int step)
        {
            if (CurrentStep == step) return Colors.Gold;
            if (step < CurrentStep) return Colors.Green;
            return Colors.Gray;
        }

        public void NextStep()
        {
            if (CurrentStep < 4)
                CurrentStep++;
            OnStepChanged();
        }

        public void GoToStep(int step)
        {
            if (step >= 1 && step < CurrentStep)
            {
                CurrentStep = step;
                OnStepChanged();
            }
        }

        private void OnGoToStep(object param)
        {
            if (param is string s && int.TryParse(s, out int
step))
                GoToStep(step);
        }

        private void OnStepChanged()
        {
            OnPropertyChanged(nameof(CurrentStep));
            OnPropertyChanged(nameof(Step1Color));
            OnPropertyChanged(nameof(Step2Color));
            OnPropertyChanged(nameof(Step3Color));
            OnPropertyChanged(nameof(Step4Color));
            OnPropertyChanged(nameof(CanGoToStep2));
            OnPropertyChanged(nameof(CanGoToStep3));
            OnPropertyChanged(nameof(CanGoToStep4));
        }
```

```
        public View GetCurrentStepContent()
        {
            return CurrentStep switch
            {
                1 => new Step1_ServiceSelection(this),
                2 => new Step2_CarSelection(this),
                3 => new Step3_TimeAndMaster(this),
                4 => new Step4_Confirmation(this),
                _ => new Label { Text = "Неизвестный этап" }
            };
        }
    }
}
```

Листинг Г.45 – Файл RegisterViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public partial class RegisterViewModel : ObservableObject
    {
        private readonly AuthService _authService = new();

        [RelayCommand]
        private async Task RegisterAsync()
        {
            var success = await _authService.RegisterAsync(new
RegisterRequest
            {
                FullName = FullName,
                Email = Email,
                Password = Password,
                PhoneNumber = PhoneNumber,
                Role = Role
            )

        [RelayCommand]
        private async Task GoToLogin()
        {
            await Shell.Current.GoToAsync("//LoginPage");
        }
    }
}
```

Листинг Г.46 – Файл MasterScheduleViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public class MasterScheduleViewModel : INotifyPropertyChanged
    {

        public MasterScheduleViewModel()
        {
            InitializeWeek();
```

```
        }


        public async Task LoadScheduleForDayAsync(GroupedSchedule
schedule)
        {
            if (schedule == null || schedule.IsLoaded)
                return;

            schedule.IsLoading = true;


            schedule.Intervals = GenerateDailyIntervals();
            schedule.IsLoaded = true;
            schedule.IsLoading = false;
        }

        private ObservableCollection<WorkInterval>
GenerateDailyIntervals()
        {
            var intervals = new
ObservableCollection<WorkInterval>();

            for (int h = 0; h < 24; h++)
            {
                var start = new TimeSpan(h, 0, 0);
                var end = start.Add(TimeSpan.FromHours(1));
                intervals.Add(new WorkInterval
                {
                    TimeRange = $"{start:hh\\:mm}-{end:hh\\:mm}",
                    IsWorking = false
                });
            }

            return intervals;
        }

        public ICommand ToggleStatusCommand => new
Command<WorkInterval>(async (interval) =>
        {
            if (interval != null && !interval.IsBusy)
            {
                interval.IsBusy = true;

                // Меняем статус локально
                interval.IsWorking = !interval.IsWorking;


                interval.IsBusy = false;
            }
        });
```

```
        public event PropertyChangedEventHandler PropertyChanged;
        private void OnPropertyChanged([CallerMemberName] string
name = "") =>
            PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(name));
    }
}
```

Листинг Г.47 – Файл MasterProfileViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public partial class MasterProfileViewModel : ObservableObject
    {


        public ICommand LogoutCommand { get; }

        public MasterProfileViewModel()
        {
            LoadProfile();
            LogoutCommand = new AsyncRelayCommand(OnLogout);
        }
        public void LoadProfile()
        {
            var user = AuthService.CurrentUser;
            if (user != null)
            {
                FullName = user.FullName;
                Email = user.Email;
                Phone = user.PhoneNumber;
            }
        }

        private async Task OnLogout()
        {
            // Очистка токена
            await SecureStorage.SetAsync("auth_token",
string.Empty);
            Application.Current.MainPage = new AppShell();
            await Shell.Current.GoToAsync("///LoginPage");

        }
    }
}
```

Листинг Г.48 – Файл MasterMainViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public class MasterMainViewModel : INotifyPropertyChanged
    {

        private void LoadAppointmentsForDate(DateTime date)
```

```
        {
            Appointments.Clear();

            var allAppointments = new List<AppointmentModel>
            {
                new() { Time = "09:00", ServiceType = "Замена
масла", Car = "BMW X5", Status = "Ожидает" },
                new() { Time = "11:30", ServiceType =
"Диагностика", Car = "Audi A6", Status = "В работе" },
                new() { Time = "14:00", ServiceType = "Ремонт
подвески", Car = "Toyota Camry", Status = "Завершено" }
            };

            foreach (var appt in allAppointments)
            {
                Appointments.Add(appt);
            }
        }

        private void OnPropertyChanged([CallerMemberName] string
name = null)
        {
            PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(name));
        }
    }
}
```

Листинг Г.49 – Файл ManageServicesViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public partial class ManageServicesViewModel :
ObservableObject
    {

        public ManageServicesViewModel()
        {
            Services = new ObservableCollection<ServiceModel>
        {
            new ServiceModel { Name = "Замена масла", Description
= "Быстрая и качественная замена масла", Price = 1200,
DurationMinutes = 30 },
            new ServiceModel { Name = "Диагностика двигателя",
Description = "Комплексная диагностика", Price = 1500,
DurationMinutes = 45 }
        };

            AddServiceCommand = new RelayCommand(AddService);
            RemoveServiceCommand = new
RelayCommand<ServiceModel>(RemoveService);
            EditServiceCommand = new
RelayCommand<ServiceModel>(BeginEditService);
```

```
            SaveEditServiceCommand = new
RelayCommand(SaveEditService);
            CancelEditServiceCommand = new
RelayCommand(CancelEditService);
        }

        private void AddService()
        {
            if (string.IsNullOrWhiteSpace(NewServiceName)) return;

            decimal price = 0;
            int duration = 0;
            decimal.TryParse(NewServicePrice, out price);
            int.TryParse(NewServiceDurationMinutes, out duration);

            Services.Add(new ServiceModel
            {
                Name = NewServiceName,
                Description = NewServiceDescription,
                Price = price,
                DurationMinutes = duration
            });
            NewServiceName = NewServiceDescription =
NewServicePrice = NewServiceDurationMinutes = string.Empty;
        }


        private void RemoveService(ServiceModel service)
        {
            if (service == null) return;
            Services.Remove(service);
        }

        private void BeginEditService(ServiceModel service)
        {
            if (service == null) return;

            decimal price = 0;
            int duration = 0;
            decimal.TryParse(EditServicePrice, out price);
            int.TryParse(EditServiceDurationMinutes, out
duration);

            EditingService = service;
            EditServiceName = service.Name;
            EditServiceDescription = service.Description;
            EditServicePrice = price.ToString();
            EditServiceDurationMinutes = duration.ToString();
        }

        private void SaveEditService()
        {
            if (EditingService == null) return;
```

```
            decimal price = 0;
            int duration = 0;
            decimal.TryParse(EditServicePrice, out price);
            int.TryParse(EditServiceDurationMinutes, out
duration);

            EditingService.Name = EditServiceName;
            EditingService.Description = EditServiceDescription;
            EditingService.Price = price;
            EditingService.DurationMinutes = duration;

            // Force UI refresh
            var i = Services.IndexOf(EditingService);
            if (i >= 0)
            {
                Services.RemoveAt(i);
                Services.Insert(i, EditingService);
            }
            CancelEditService();
        }

        private void CancelEditService()
        {
            EditingService = null;
            EditServiceName = EditServiceDescription =
string.Empty;
            EditServicePrice = "0";
            EditServiceDurationMinutes = "0";
        }
    }

}
```

Листинг Г.50 – Файл ManageMastersViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{

    public partial class ManageMastersViewModel : ObservableObject
    {
        [ObservableProperty]
        private ObservableCollection<MasterModel> masters;


        public ManageMastersViewModel()
        {
            Masters = new ObservableCollection<MasterModel>
        {
            new MasterModel { FullName = "Иванов Иван", Email =
"ivanov@mail.com", Phone = "+7 900 123-45-67" },
            new MasterModel { FullName = "Петров Петр", Email =
"petrov@mail.com", Phone = "+7 901 456-78-90" }
        };
```

```
            AddMasterCommand = new RelayCommand(AddMaster);
            RemoveMasterCommand = new
RelayCommand<MasterModel>(RemoveMaster);
            EditMasterCommand = new
RelayCommand<MasterModel>(BeginEditMaster);
            SaveEditMasterCommand = new
RelayCommand(SaveEditMaster);
            CancelEditMasterCommand = new
RelayCommand(CancelEditMaster);
        }

        private void AddMaster()
        {
            if (!string.IsNullOrWhiteSpace(NewFullName) &&
                !string.IsNullOrWhiteSpace(NewEmail) &&
                !string.IsNullOrWhiteSpace(NewPhone))
            {
                Masters.Add(new MasterModel { FullName =
NewFullName, Email = NewEmail, Phone = NewPhone });
                NewFullName = NewEmail = NewPhone = string.Empty;
            }
        }

        private void RemoveMaster(MasterModel master)
        {
            if (master == null) return;
            Masters.Remove(master);
        }

        private void BeginEditMaster(MasterModel master)
        {
            if (master == null) return;
            EditingMaster = master;
            EditFullName = master.FullName;
            EditEmail = master.Email;
            EditPhone = master.Phone;
        }

        private void SaveEditMaster()
        {
            if (EditingMaster == null) return;
            EditingMaster.FullName = EditFullName;
            EditingMaster.Email = EditEmail;
            EditingMaster.Phone = EditPhone;
            // Force UI refresh
            var i = Masters.IndexOf(EditingMaster);
            if (i >= 0)
            {
                Masters.RemoveAt(i);
                Masters.Insert(i, EditingMaster);
            }
            CancelEditMaster();
```

```
        }

        private void CancelEditMaster()
        {
            EditingMaster = null;
            EditFullName = EditEmail = EditPhone = string.Empty;
        }
    }
}
```

Листинг Г.51 – Файл LoginViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{


        public LoginViewModel()
        {
            _authService = new AuthService(new HttpClient
            {
                BaseAddress = new Uri(AppSettings.BaseApiUrl)
            });
        }

        [RelayCommand]
        private async Task Login()
        {
            var success = await _authService.LoginAsync(Email,
Password);

            if (success)
            {
                var role = await
SecureStorage.Default.GetAsync("user_role");

                if (role.Equals("Client"))
                {
                    ((AppShell)Shell.Current).LoadClientShell();
                    await
Shell.Current.GoToAsync("//ClientMainPage");
                }
                else if (role.Equals("Master"))
                {
                    ((AppShell)Shell.Current).LoadMasterShell();
                    await
Shell.Current.GoToAsync("//MasterMainPage");
                }
                else if (role.Equals("Admin"))
                {
                    ((AppShell)Shell.Current).LoadAdminShell();

            }
        }
```

```
        [RelayCommand]
        private async Task GoToRegister()
        {
            await Shell.Current.GoToAsync("//RegisterPage");
        }
    }

}
```

Листинг Г.52 – Файл GarageViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public partial class GarageViewModel : ObservableObject
    {
        [ObservableProperty]
        ObservableCollection<CarModel> cars;

        public IRelayCommand AddCarCommand { get; }
        public IRelayCommand<CarModel> RemoveCarCommand { get; }
        public IRelayCommand GoBackCommand { get; }

        public GarageViewModel()
        {
            Cars = new ObservableCollection<CarModel>
        {
            new CarModel { Name = "Toyota Camry", ImagePath =
"car_icon.png", Status = "Записан на обслуживание",
IsStatusVisible = true },
            new CarModel { Name = "Mazda CX-5", ImagePath =
"car_icon.png", Status = "", IsStatusVisible = false }
        };

            AddCarCommand = new RelayCommand(AddCar);
            RemoveCarCommand = new
RelayCommand<CarModel>(RemoveCar);
            GoBackCommand = new RelayCommand(GoBack);

            // Подписка на событие добавления автомобиля
            MessagingCenter.Subscribe<AddCarViewModel,
CarModel>(this, "CarAdded", (sender, car) =>
            {
                Cars.Add(car);
            });
        }

        private async void AddCar()
        {
            await Shell.Current.GoToAsync("AddCarPage");
        }

        private async void RemoveCar(CarModel car)
        {
            if (car == null) return;
```

```
            var confirm = await
Application.Current.MainPage.DisplayAlert(
                "Удалить автомобиль?", $"Удалить {car.Name} из
гаража?", "Удалить", "Отмена");
            if (confirm)
                Cars.Remove(car);
        }

        private async void GoBack()
        {
            await Shell.Current.GoToAsync("..");
        }
    }
}
```

Листинг Г.53 – Файл ClientServiceHistoryViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public partial class ClientServiceHistoryViewModel :
ObservableObject
    {
        [ObservableProperty]
        private ObservableCollection<ServiceHistoryItem> history;

        public IRelayCommand<ServiceHistoryItem> OpenDetailCommand
{ get; }

        public ClientServiceHistoryViewModel()
        {
            History = new ObservableCollection<ServiceHistoryItem>
            {
                new ServiceHistoryItem
                {
                    Service = "Замена масла",
                    Car = "Toyota Camry",
                    Date = "12.03.2024",
                    Status = "Обслужен",
                    NotificationHistory = new
ObservableCollection<string>
                    {
                        "10.03.2024 12:00 — Запись создана",
                        "11.03.2024 14:21 — Мастер подтвердил",
                        "12.03.2024 10:10 — Выполнено"
                    }
                },
                new ServiceHistoryItem
                {
                    Service = "Диагностика подвески",
                    Car = "Mazda CX-5",
                    Date = "05.02.2024",
                    Status = "Обслужен",
                    NotificationHistory = new
ObservableCollection<string>
```

```
                    {
                        "03.02.2024 15:20 — Запись создана",
                        "04.02.2024 13:45 — Подтверждено",
                        "05.02.2024 17:00 — Выполнено"
                    }
                }
            };

            OpenDetailCommand = new
RelayCommand<ServiceHistoryItem>(OpenDetail);
        }

        private async void OpenDetail(ServiceHistoryItem item)
        {
            if (item == null) return;
            await
Shell.Current.GoToAsync("ClientServiceHistoryDetailPage",
                new Dictionary<string, object> { {
"ServiceHistory", item } });
        }
    }
}
```

Листинг Г.54 – Файл ClientServiceHistoryDetailViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    [QueryProperty(nameof(ServiceHistory), "ServiceHistory")]
    public partial class ClientServiceHistoryDetailViewModel :
ObservableObject
    {
        [ObservableProperty]
        private ServiceHistoryItem serviceHistory;

        public IRelayCommand GoBackCommand { get; }

        public ClientServiceHistoryDetailViewModel()
        {
            GoBackCommand = new RelayCommand(GoBack);
        }

        private async void GoBack()
        {
            await Shell.Current.GoToAsync("..");
        }
    }
}
```

Листинг Г.55 – Файл ClientProfileViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public partial class ClientProfileViewModel : ObservableObject
    {
```

```
        public ICommand LogoutCommand { get; }

        public ClientProfileViewModel()
        {
            LoadProfile();
            LogoutCommand = new AsyncRelayCommand(OnLogout);
        }
        public void LoadProfile()
        {
            var user = AuthService.CurrentUser;
            if (user != null)
            {
                FullName = user.FullName;
                Email = user.Email;
                Phone = user.PhoneNumber;
            }
        }

        private async Task OnLogout()
        {
            // Очистка токена
            await SecureStorage.SetAsync("auth_token",
string.Empty);
            Application.Current.MainPage = new AppShell();
            await Shell.Current.GoToAsync("///LoginPage");

        }
    }
}
```

Листинг Г.56 – Файл ClientMainPageViewModel.cs

```
public partial class ClientMainPageViewModel : ObservableObject
{
    [ObservableProperty] public string userName =
AuthService.CurrentUser.FullName;
    [ObservableProperty] private
ObservableCollection<ServiceHistoryItem> serviceHistory;
    public IRelayCommand BookServiceCommand { get; }
    public IRelayCommand OpenGarageCommand { get; }
    public IRelayCommand OpenFullHistoryCommand { get; }

    public ClientMainPageViewModel()
    {
        ServiceHistory = new
ObservableCollection<ServiceHistoryItem>
        {
            new ServiceHistoryItem { Service = "Замена масла", Car
= "Mazda CX-5", Date = DateTime.Today.AddDays(-2).ToString(),
Status = "Обслужен" },
```

```
            new ServiceHistoryItem { Service = "Диагностика", Car
= "Toyota Camry", Date = DateTime.Today.AddDays(-15).ToString(),
Status = "Обслужен" }
        };
        BookServiceCommand = new RelayCommand(OpenBooking);
        OpenGarageCommand = new RelayCommand(OpenGarage);
        OpenFullHistoryCommand = new
RelayCommand(OpenFullHistory);
    }

    private async void OpenBooking()
    {
        await Shell.Current.GoToAsync("ServiceBookingPage");
    }

    private async void OpenGarage()
    {
        await Shell.Current.GoToAsync("GaragePage");
    }

    private async void OpenFullHistory()
    {
        await Shell.Current.GoToAsync("ClientServiceHistoryPage");
    }
}
```

Листинг Г.57 – Файл ClientAppointmentsViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public partial class ClientAppointmentsViewModel :
ObservableObject
    {
        [ObservableProperty]
        private ObservableCollection<AppointmentItem>
appointments;

        public IRelayCommand<AppointmentItem> OpenDetailCommand {
get; }

        public ClientAppointmentsViewModel()
        {
            Appointments = new
ObservableCollection<AppointmentItem>
            {
                new AppointmentItem
                {
                    ServiceName = "Замена масла",
                    DateTime = new DateTime(2025, 5, 20, 10, 0,
0),
                    Status = "Ожидает подтверждения",
                    NotificationHistory = new
ObservableCollection<string>
                    {
```

```
                        "18.05.2025 12:00 — Запись создана"
                    }
                },
                new AppointmentItem
                {
                    ServiceName = "Диагностика двигателя",
                    DateTime = new DateTime(2025, 5, 25, 14, 0,
0),
                    Status = "Подтверждена",
                    NotificationHistory = new
ObservableCollection<string>
                    {
                        "24.05.2025 09:10 — Запись создана",
                        "24.05.2025 13:45 — Подтверждено"
                    }
                },
                new AppointmentItem
                {
                    ServiceName = "Ремонт подвески",
                    DateTime = new DateTime(2025, 6, 1, 9, 0, 0),
                    Status = "Выполнена",
                    NotificationHistory = new
ObservableCollection<string>
                    {
                        "30.05.2025 17:20 — Запись создана",
                        "31.05.2025 13:10 — Подтверждено",
                        "01.06.2025 12:00 — Выполнено"
                    }
                }
            };

            OpenDetailCommand = new
RelayCommand<AppointmentItem>(OpenDetail);
        }

        private async void OpenDetail(AppointmentItem item)
        {
            if (item == null) return;
        }
    }
}
```

Листинг Г.58 – Файл ClientAppointmentDetailViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    [QueryProperty(nameof(Appointment), "Appointment")]
    public partial class ClientAppointmentDetailViewModel :
ObservableObject
    {
        [ObservableProperty]
        private AppointmentItem appointment;

        public IRelayCommand GoBackCommand { get; }
```

```
        public ClientAppointmentDetailViewModel()
        {
            GoBackCommand = new RelayCommand(GoBack);
        }

        private async void GoBack()
        {
            await Shell.Current.GoToAsync("..");
        }
    }
}
```

Листинг Г.59 – Файл BookingViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public partial class BookingViewModel : ObservableObject
    {
        [ObservableProperty]
        private int currentStep = 1;

        [ObservableProperty]
        private BookingData bookingData = new();

        [RelayCommand]
        private void NextStep()
        {
            if (CurrentStep < 4) CurrentStep++;
        }

        [RelayCommand]
        private void PreviousStep()
        {
            if (CurrentStep > 1) CurrentStep--;
        }

        [RelayCommand]
        private async Task SubmitBooking()
        {
            await BookingService.SubmitBookingAsync(BookingData);
            await Shell.Current.GoToAsync("//ClientMainPage");
        }
    }
}
```

Листинг Г.60 – Файл AppointmentDetailViewModel.cs

```
n amespace AutoServiceMobile.ViewModels
{
    [QueryProperty(nameof(Appointment), "Appointment")]
    public partial class AppointmentDetailViewModel :
ObservableObject
    {
```

```
        [ObservableProperty]
        private AppointmentModel appointment;

        public ObservableCollection<string> Statuses { get; } =
new()
        {
            "Ожидает",
            "В работе",
            "Завершено"
        };

        [ObservableProperty]
        private string message;

        public ICommand SendMessageCommand { get; }

        public AppointmentDetailViewModel()
        {
            SendMessageCommand = new RelayCommand(SendMessage);
        }

        private void SendMessage()
        {
            App.Current.MainPage.DisplayAlert("Уведомление",
$"Сообщение отправлено клиенту:\n{Message}\nТекущий статус:
{Appointment?.Status}", "OK");

        }
    }
}
```

Листинг Г.61 – Файл AdminStatisticsViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public partial class AdminStatisticsViewModel :
ObservableObject
    {
        [ObservableProperty] private int totalOrders;
        [ObservableProperty] private decimal totalRevenue;
        [ObservableProperty] private decimal averageCheck;
        [ObservableProperty] private int newClients;

        // Динамика по дням
        public ObservableCollection<DailyStat> OrdersByDay { get;
} = new()
        {
            new DailyStat { Day = "Пн", Count = 12 },
            new DailyStat { Day = "Вт", Count = 10 },
            new DailyStat { Day = "Ср", Count = 15 },
            new DailyStat { Day = "Чт", Count = 8 },
            new DailyStat { Day = "Пт", Count = 20 },
            new DailyStat { Day = "Сб", Count = 25 },
            new DailyStat { Day = "Вс", Count = 7 }
```

```
        };

        // Топ-услуги
        public ObservableCollection<ServiceStat> TopServices {
get; } = new()
        {
            new ServiceStat { Name = "Замена масла", Count = 33 },
            new ServiceStat { Name = "Диагностика двигателя",
Count = 24 },
            new ServiceStat { Name = "Ремонт подвески", Count = 17
}
        };

        // Топ-мастера
        public ObservableCollection<MasterStat> TopMasters { get;
} = new()
        {
            new MasterStat { Name = "Иванов Иван", Count = 26 },
            new MasterStat { Name = "Петров Петр", Count = 21 },
            new MasterStat { Name = "Сидоров Сидор", Count = 18 }
        };

        public AdminStatisticsViewModel()
        {
            // Мок-данные, можешь менять
            TotalOrders = 107;
            TotalRevenue = 155000;
            AverageCheck = 1449;
            NewClients = 12;
        }
    }
```

Листинг Г.62 – Файл AdminProfileViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public partial class AdminProfileViewModel : ObservableObject
    {

        public ICommand LogoutCommand { get; }

        public AdminProfileViewModel()
        {
            LoadProfile();
            LogoutCommand = new AsyncRelayCommand(OnLogout);
        }
        public void LoadProfile()
        {
            var user = AuthService.CurrentUser;
            if (user != null)
            {
                FullName = user.FullName;
                Email = user.Email;
                Phone = user.PhoneNumber;
```

```
            }
        }

        private async Task OnLogout()
        {
            // Очистка токена
            await SecureStorage.SetAsync("auth_token",
string.Empty);
            Application.Current.MainPage = new AppShell();
            await Shell.Current.GoToAsync("///LoginPage");

        }
    }
}
```

Листинг Г.63 – Файл AddCarViewModel.cs

```
namespace AutoServiceMobile.ViewModels
{
    public partial class AddCarViewModel : ObservableObject
    {
        [ObservableProperty] private string carName;
        [ObservableProperty] private string selectedImagePath;

        public ICommand SelectImageCommand { get; }
        public ICommand SaveCommand { get; }

        public bool HasPhoto =>
!string.IsNullOrEmpty(SelectedImagePath) && SelectedImagePath !=
"car_icon.png";

        public AddCarViewModel()
        {
            SelectImageCommand = new
AsyncRelayCommand(OnSelectImage);
            SaveCommand = new AsyncRelayCommand(OnSave);

            SelectedImagePath = "car_icon.png"; // если нет фото —
показываем "Выбрать фото"
        }

        private async Task OnSelectImage()
        {
            try
            {
                var file = await MediaPicker.PickPhotoAsync(new
MediaPickerOptions
                {
                    Title = "Выберите фото автомобиля"
                });
                if (file != null)
                {
                    SelectedImagePath = file.FullPath;
```

```
                    OnPropertyChanged(nameof(HasPhoto));
                }
            }
            catch (Exception ex)
            {
                await App.Current.MainPage.DisplayAlert("Ошибка",
ex.Message, "Ок");
            }
        }

        private async Task OnSave()
        {
            if (string.IsNullOrWhiteSpace(CarName))
            {
                await App.Current.MainPage.DisplayAlert("Ошибка",
"Введите название автомобиля", "Ок");
                return;
            }

            var newCar = new CarModel
            {
                Name = CarName,
                ImagePath =
string.IsNullOrEmpty(SelectedImagePath) ? "car_icon.png" :
SelectedImagePath,
                Status = "",
                IsStatusVisible = false
            };

            // Отправляем машину подписчикам (GarageViewModel)
            MessagingCenter.Send(this, "CarAdded", newCar);

            await App.Current.MainPage.DisplayAlert("Успех",
$"Автомобиль '{CarName}' добавлен", "Ок");
            await Shell.Current.GoToAsync("..");
        }
    }
}
```

Листинг Г.64 – Файл UIMessageService.cs

```
namespace AutoServiceMobile.Utils
{
    public static class UIMessageService
    {
        public static async Task ShowInfo(string message, string
title = "Информация")
        {
            await Application.Current.MainPage.DisplayAlert(title,
message, "OK");
        }

        public static async Task<bool> ShowConfirm(string message,
string title = "Подтвердите")
```

```
        {
            return await
Application.Current.MainPage.DisplayAlert(title, message, "Да",
"Нет");
        }

        public static async Task ShowError(string message)
        {
            await
Application.Current.MainPage.DisplayAlert("Ошибка", message,
"OK");
        }
    }

}
```

Г.65 – Файл WorkSlotService.cs

```
namespace AutoServiceMobile.Services
{
    public class WorkSlotService
    {
        private readonly HttpClient _httpClient;

        public WorkSlotService()
        {
            _httpClient = new HttpClient
            {
                BaseAddress = new Uri(AppSettings.BaseApiUrl)
            };
        }

        private async Task<bool> SetAuthHeaderAsync()
        {
            var token = await
SecureStorage.GetAsync("auth_token");
            if (string.IsNullOrEmpty(token)) return false;

            _httpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
            return true;
        }

        /// <summary>
        /// Получить рабочие слоты текущего мастера.
        /// </summary>
        public async Task<List<WorkSlotModel>?>
GetMyWorkSlotsAsync()
        {
            if (!await SetAuthHeaderAsync()) return null;
                return await
response.Content.ReadFromJsonAsync<List<WorkSlotModel>>();
            }
            catch
```

```
            {
                return null;
            }
        }

        /// <summary>
        /// Создать рабочий слот.
        /// </summary>
        public async Task<bool> CreateWorkSlotAsync(WorkSlotModel
slot)
        {
            if (!await SetAuthHeaderAsync()) return false;

            try
            {
                var response = await
_httpClient.PostAsJsonAsync("api/workslots", slot);
                return response.IsSuccessStatusCode;
            }
            catch
            {
                return false;
            }
        }

        /// <summary>
        /// Удалить рабочий слот.
        /// </summary>
        public async Task<bool> DeleteWorkSlotAsync(int slotId)
        {
            if (!await SetAuthHeaderAsync()) return false;
```

Листинг Г.66 – Файл UserService.cs

```
n amespace AutoServiceMobile.Services
{
    public class UserService
    {
        private readonly HttpClient _httpClient;

        public UserService()
        {
            _httpClient = new HttpClient
            {
                BaseAddress = new Uri(AppSettings.BaseApiUrl)
            };
        }

        private async Task<bool> SetAuthHeaderAsync()
        {
            var token = await
SecureStorage.GetAsync("auth_token");
            if (string.IsNullOrEmpty(token)) return false;
```

```
            _httpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
            return true;
        }

        public async Task<List<UserProfile>?> GetAllUsersAsync()
        {
            if (!await SetAuthHeaderAsync()) return null;



        public async Task<UserProfile?> GetUserByIdAsync(int
userId)
        {
            if (!await SetAuthHeaderAsync()) return null;

            try
            {
                var response = await
_httpClient.GetAsync($"api/users/{userId}");
                if (!response.IsSuccessStatusCode) return null;

                return await
response.Content.ReadFromJsonAsync<UserProfile>();
            }
            catch
            {
                return null;
            }

        }

        public async Task<bool> DeleteUserAsync(int userId)
        {
            if (!await SetAuthHeaderAsync()) return false;

            try
            {
                var response = await
_httpClient.DeleteAsync($"api/users/{userId}");
                return response.IsSuccessStatusCode;
            }
            catch
            {
                return false;
            }
        }

        public async Task<bool> UpdateUserAsync(UserProfile
updatedUser)
        {
            if (!await SetAuthHeaderAsync()) return false;
```

Листинг Г.67 – Файл ReviewService.cs

```
namespace AutoServiceMobile.Services
{
    public class ReviewService
    {
        private readonly HttpClient _httpClient;

        public ReviewService()
        {
            _httpClient = new HttpClient
            {
                BaseAddress = new Uri(AppSettings.BaseApiUrl)
            };
        }

        private async Task<bool> SetAuthHeaderAsync()
        {
            var token = await
SecureStorage.GetAsync("auth_token");
            if (string.IsNullOrEmpty(token)) return false;

            _httpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
            return true;
        }

        public async Task<List<ReviewModel>?>
GetReviewsByMasterIdAsync(int masterId)
        {
            if (!await SetAuthHeaderAsync()) return null;

            try
            {
                var response = await
_httpClient.GetAsync($"api/reviews/master/{masterId}");
            catch
            {
                return null;
            }
        }

        public async Task<bool> SubmitReviewAsync(ReviewModel
review)
        {
            if (!await SetAuthHeaderAsync()) return false;
```

Листинг Г.68 – Файл NotificationService.cs

```
namespace AutoServiceMobile.Services
{
    public class NotificationService
    {
        private readonly HttpClient _httpClient;
```

```
        public NotificationService()
        {
            _httpClient = new HttpClient
            {
                BaseAddress = new Uri(AppSettings.BaseApiUrl)
            };
        }

        public NotificationService(HttpClient httpClient)
        {
            _httpClient = httpClient;
        }

        /// <summary>
        /// Загружает список уведомлений для текущего пользователя
        /// </summary>
        public async Task<List<Notification>?>
GetNotificationsAsync()
        {
            var token = await
SecureStorage.GetAsync("auth_token");
            if (string.IsNullOrEmpty(token)) return null;

            _httpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);

            try
            {
                var response = await
_httpClient.GetAsync("/api/notifications");
                if (!response.IsSuccessStatusCode) return null;

                var json = await
response.Content.ReadAsStringAsync();
                return
JsonSerializer.Deserialize<List<Notification>>(json, new
JsonSerializerOptions { PropertyNameCaseInsensitive = true });
            }
            catch
            {
                return null;
            }
        }

        /// <summary>
        /// Помечает уведомление как прочитанное
        /// </summary>
        public async Task<bool> MarkAsReadAsync(int
notificationId)
        {
            var token = await
SecureStorage.GetAsync("auth_token");
```

```
            if (string.IsNullOrEmpty(token)) return false;

            _httpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
```

Листинг Г.69 – Файл MasterService.cs

```
        public MasterService()
        {
            _httpClient = new HttpClient
            {
                BaseAddress = new Uri(AppSettings.BaseApiUrl)
            };
        }

        private async Task<bool> SetAuthHeaderAsync()
        {
            var token = await
SecureStorage.GetAsync("auth_token");
            if (string.IsNullOrEmpty(token)) return false;


            _

        public async Task<List<MasterModel>?> GetAllMastersAsync()
        {
            if (!await SetAuthHeaderAsync()) return null;

            try
            {
                var response = await
_httpClient.GetAsync("api/masters");

await response.Content.ReadFromJsonAsync<List<MasterModel>>();
            }
            catch
            {
                return null;
            }
        }

        public async Task<MasterModel?> GetMasterByIdAsync(int
masterId)
        {
            if (!await SetAuthHeaderAsync()) return null;

            try
            {
                var response = await
_httpClient.GetAsync($"api/masters/{masterId}");

            catch
            {
                return null;
            }
```

```
        }
    }
}
```

Листинг Г.70 – Файл CarService.cs

```
namespace AutoServiceMobile.Services
{
    public class CarService
    {
        private readonly HttpClient _httpClient;

        public CarService()
        {
            _httpClient = new HttpClient
            {
                BaseAddress = new Uri(AppSettings.BaseApiUrl)
            };
        }

        private async Task<bool> SetAuthHeaderAsync()
        {
            var token = await
SecureStorage.GetAsync("auth_token");
            if (string.IsNullOrEmpty(token)) return false;

            _httpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
            return true;
        }

        /// <summary>
        /// Получить список автомобилей текущего пользователя.
        /// </summary>
        public async Task<List<CarModel>?> GetMyCarsAsync()
        {
            if (!await SetAuthHeaderAsync()) return null;

            try
            {
                var response = await
_httpClient.GetAsync("api/cars/my");

await response.Content.ReadFromJsonAsync<List<CarModel>>();
            }
            catch
            {
                return null;
            }
        }

        /// <summary>
        /// Добавить новый автомобиль.
        /// </summary>
```

```
        public async Task<bool> AddCarAsync(CarModel car)
        {
            if (!await SetAuthHeaderAsync()) return false;
```

Листинг Г.71 – Файл AuthService.cs

```
namespace AutoServiceMobile.Services
{
    public class AuthService
    {
        public static UserProfile? CurrentUser { get; private set;
}

        public AuthService()
        {
            _httpClient = new HttpClient
            {
                BaseAddress = new Uri(AppSettings.BaseApiUrl)
            };
        }

        public AuthService(HttpClient httpClient)
        {
            _httpClient = httpClient;
        }

        public async Task<bool> LoginAsync(string email, string
password)
        {
            var response = await
_httpClient.PostAsJsonAsync("/api/Auth/login", new
            {
                Email = email,
                Password = password
            });

            if (result != null)
            {
                await SecureStorage.Default.SetAsync("auth_token",
result.Token);
                await SecureStorage.Default.SetAsync("user_role",
result.Role);
                await LoadUserProfileAsync();
                return true;
            }

            return false;
        }

        public async Task<bool> LoadUserProfileAsync()
        {
            var token = await
SecureStorage.GetAsync("auth_token");
```

```
            if (string.IsNullOrEmpty(token)) return false;

            _httpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);

            try
            {
                var response = await
_httpClient.GetAsync("/api/auth/profile");
                if (!response.IsSuccessStatusCode) return false;

                var json = await
response.Content.ReadAsStringAsync();
                CurrentUser =
JsonSerializer.Deserialize<UserProfile>(json, new
JsonSerializerOptions { PropertyNameCaseInsensitive = true });

                return true;
            }
            catch
            {
                return false;
            }
        }

        public async Task<bool> RegisterAsync(RegisterRequest
request)
        {
            var response = await
_httpClient.PostAsJsonAsync("api/Auth/register", request);
            return response.IsSuccessStatusCode;
        }
    }
}
```

Листинг Г.72 – Файл WorkInterval.cs

```
namespace AutoServiceMobile.Models
{
    public class WorkInterval : INotifyPropertyChanged
    {
        private bool isWorking;
        public bool IsWorking
        {
            get => isWorking;
            set
            {
                if (isWorking != value)
                {
                    isWorking = value;
                    OnPropertyChanged();
                    OnPropertyChanged(nameof(Status));
                    OnPropertyChanged(nameof(StatusColor));
                }
```

```
            }
        }

        public string TimeRange { get; set; }

        public string Status => IsWorking ? "Рабочее время" :
"Выходной";
        public Color StatusColor => IsWorking ? Colors.LightGreen
: Colors.LightGray;

        public event PropertyChangedEventHandler PropertyChanged;
        private void OnPropertyChanged([CallerMemberName] string
name = "") =>
            PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(name));
    }
}
```

Листинг Г.73 – Файл GroupedSchedule.cs

```
namespace AutoServiceMobile.Models
{
    public class GroupedSchedule : INotifyPropertyChanged
    {

        private ObservableCollection<WorkInterval> intervals =
new();
        public ObservableCollection<WorkInterval> Intervals
        {
            get => intervals;
            set
            {
                intervals = value;
                OnPropertyChanged();
            }
        }

        public bool IsLoaded { get; set; } = false;

        private bool isLoading;
        public bool IsLoading
        {
            get => isLoading;
            set
            {
                isLoading = value;
                OnPropertyChanged();
            }
        }

        public GroupedSchedule(string day)
        {
            Day = day;
        }
```

```
        public event PropertyChangedEventHandler PropertyChanged;
        private void OnPropertyChanged([CallerMemberName] string
name = "") =>
            PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(name));
    }
}
```

Листинг Г.74 – Файл CarModel.cs

```
public class CarModel : INotifyPropertyChanged
{
    public bool IsStatusVisible
    {
        get => isStatusVisible;
        set
        {
            if (isStatusVisible != value)
            {
                isStatusVisible = value;
                OnPropertyChanged();
            }
        }
    }
}
```

Листинг Г.75 – Файл AppSettings.cs

```
namespace AutoServiceMobile.Auth
{
    /// <summary>
    /// Настройка подключения
    /// </summary>
    public static class AppSettings
    {
        public static string BaseApiUrl =
"http://192.168.0.101:5198/";
    }
}
```

Листинг Г.76 – Файл AppointmentItem.cs

```
using System;
using System.Collections.ObjectModel;

namespace AutoServiceMobile.Models
{
    public class AppointmentItem
    {
        public string ServiceName { get; set; }
        public DateTime DateTime { get; set; }
        public string Status { get; set; }
        public ObservableCollection<string> NotificationHistory {
get; set; } = new();
```

```
    }
}
```

Листинг Г.77 – Файл AppointmentModel.cs

```
using System.ComponentModel;
using System.Runtime.CompilerServices;
using Microsoft.Maui.Graphics;

namespace AutoServiceMobile.Models
{
    public class AppointmentModel : INotifyPropertyChanged
    {
        private string status;

        public string Time { get; set; }
        public string ServiceType { get; set; }
        public string Car { get; set; }
        public string Status
        {
            get => status;
            set
            {
                if (status != value)
                {
                    status = value;
                    OnPropertyChanged();
                    OnPropertyChanged(nameof(StatusColor));
                }
            }
        }

        public Color StatusColor
            => Status switch
            {
                "В работе" => Colors.Orange,
                "Ожидает" => Colors.SteelBlue,
                "Завершено" => Colors.ForestGreen,
                _ => Colors.Gray
            };

        public event PropertyChangedEventHandler PropertyChanged;
        private void OnPropertyChanged([CallerMemberName] string
name = null) =>
            PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(name));
    }
}
```

Листинг Г.78 – Файл AuthResponse.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;

namespace AutoServiceMobile.Models
{
    public class AuthResponse
    {
        public string Token { get; set; }
        public string Role { get; set; }
    }

    public class UserProfile
    {
        public int Id { get; set; }
        public string FullName { get; set; }
        public string Email { get; set; }
        public string PhoneNumber { get; set; }
        public string Role { get; set; }
    }

}
```

Листинг Г.79 – Файл BookingData.cs

```
namespace AutoServiceMobile.Models
{
    public class BookingData
    {
        public string SelectedService { get; set; }
        public string CarModel { get; set; }
        public string MasterName { get; set; }
        public DateTime? AppointmentTime { get; set; }
    }
}
```

Листинг Г.80 – Файл AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android">
     <application android:allowBackup="true"
                android:icon="@mipmap/appicon"
                android:roundIcon="@mipmap/appicon_round"
                android:supportsRtl="true"
                android:usesCleartextTraffic="true"></application>
     <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
     <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

Приложение Д

(обязательное)

Исходный код функций тестирования

Листинг Д.1 – Файл CreateAppointment_Test.cs

```
[Fact]
public async Task
CreateAppointment_ValidClient_ReturnsCreatedResultWithAppointment(
)
{
    // Arrange
    var mockContext = TestDbContextFactory.Create();
    var controller =
TestControllerFactory.CreateAppointmentControllerWithUser(mockCont
ext, userId: 3, role: "Client");

    var newAppointment = new Appointment
    {
        MasterId = 1,
        ServiceId = 2,
        StartDateTime = DateTime.Now.AddHours(1),
        Status = "Ожидает",
        CreatedAt = DateTime.Now
    };

    // Act
    var response = await
controller.CreateAppointment(newAppointment);
    var result =
Assert.IsType<CreatedAtActionResult>(response.Result);
    var created = Assert.IsType<Appointment>(result.Value);

    // Assert
    Assert.Equal(3, created.ClientId); // Проверка назначения ID
клиента
    Assert.Equal("Ожидает", created.Status);
    Assert.Equal(1, mockContext.Appointments.Count());
}
```

Листинг Д.2 – Файл DeleteAppointment_Test.cs

```
[Fact]
public async Task DeleteAppointment_ByOwner_DeletesSuccessfully()
{
    // Arrange
    var appointment = new Appointment
    {
        Id = 7,
        ClientId = 99,
        MasterId = 1,
        ServiceId = 1,
        StartDateTime = DateTime.Now.AddDays(1),
        Status = "Запланировано",
        CreatedAt = DateTime.UtcNow
    };

    var context = TestDbContextFactory.Create();
```

```
    context.Appointments.Add(appointment);
    await context.SaveChangesAsync();

    var controller =
TestControllerFactory.CreateAppointmentControllerWithUser(context,
userId: 99, role: "Client");

    // Act
    var result = await controller.DeleteAppointment(7);

    // Assert
    Assert.IsType<NoContentResult>(result);
    Assert.Empty(context.Appointments);
}
```

Приложение Е

(рекомендованное)

Результаты тестирования серверной части приложения

Рисунок Е.1 – Добавление нового пользователя

Рисунок Е.2 – Авторизация пользователя

Рисунок Е.3 – Получение сведений о клиенте с id=2

Рисунок Е.4 – Изменение пароля клиента

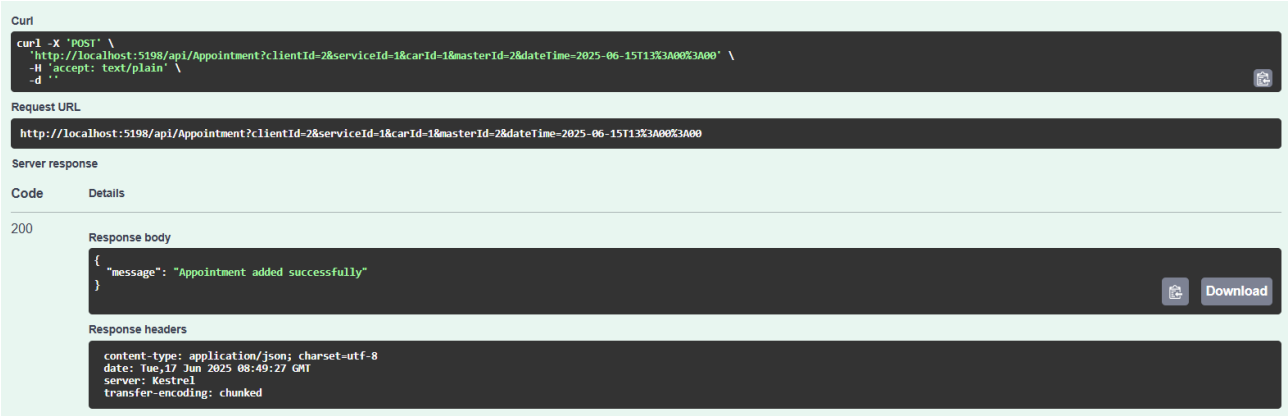

Рисунок Е.5 – Создание записи на услугу

Рисунок Е.6 – Отмена записи с id=12

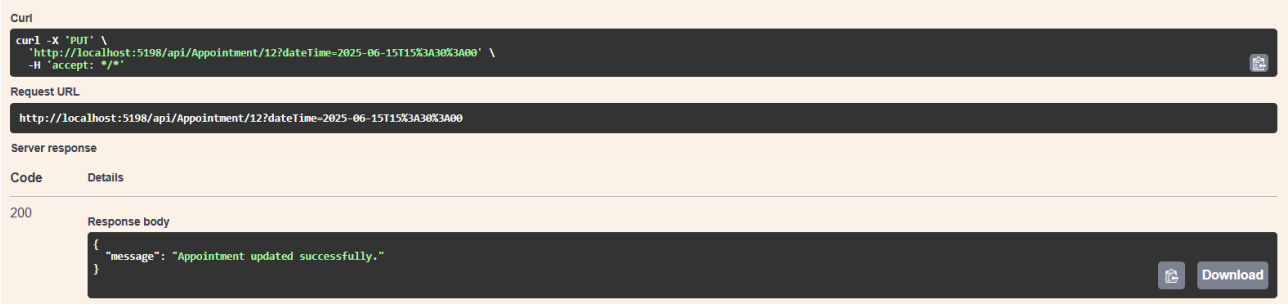

Рисунок Е.7 – Изменение времени записи с id=12

Приложение Ж

(рекомендованное)

Результаты тестирования мобильного приложения

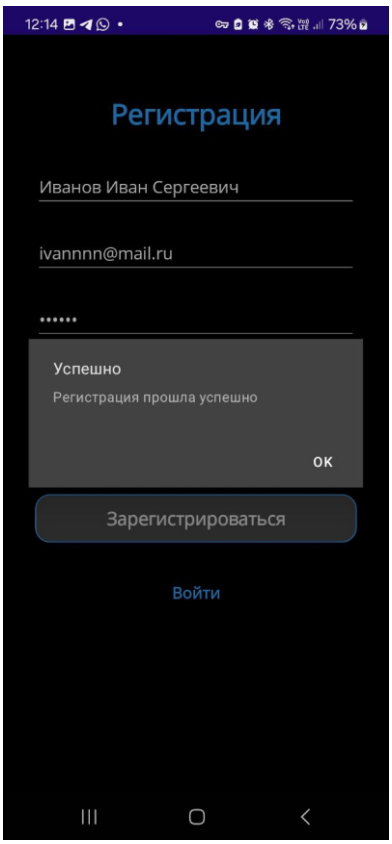

Рисунок Ж.1 – Успешная регистрация

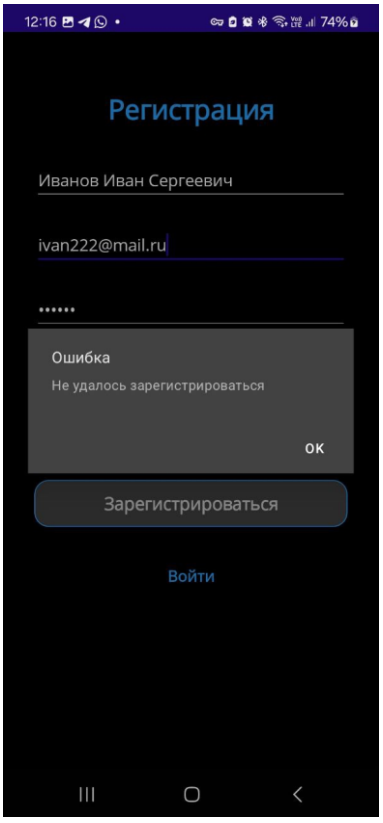

Рисунок Ж.2 – Попытка регистрации с уже использованным email

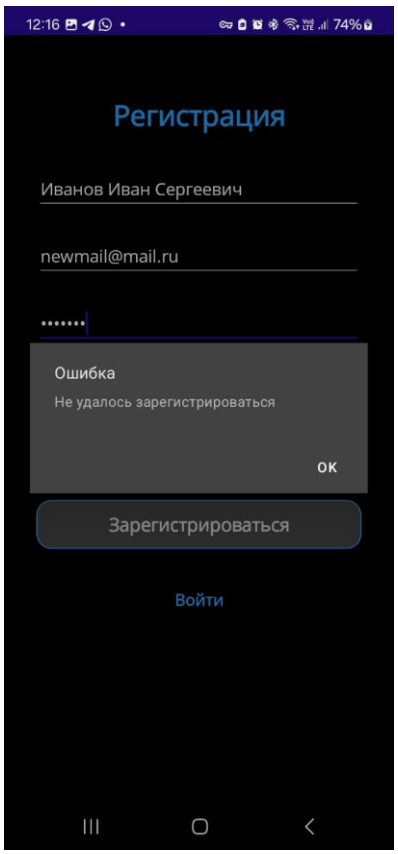

Рисунок Ж.3 – Недопустимые символы в пароле

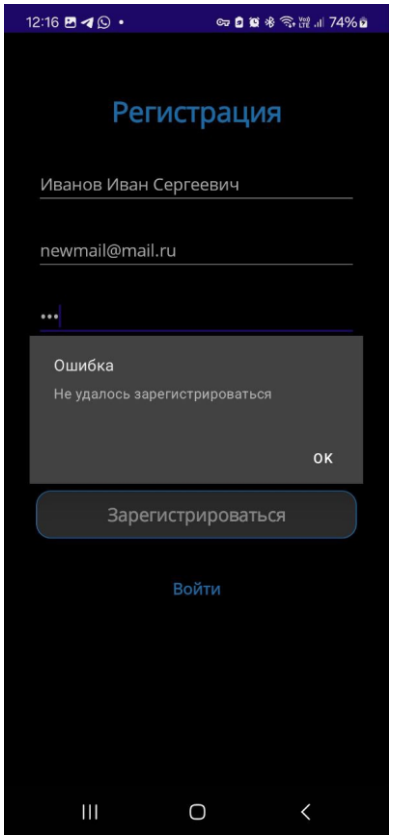

Рисунок Ж.4 – Короткий пароль

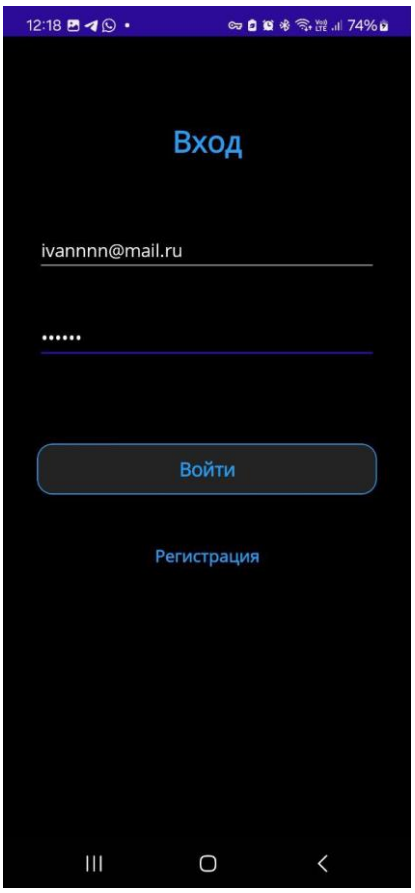

Рисунок Ж.5 – Успешный вход в систему

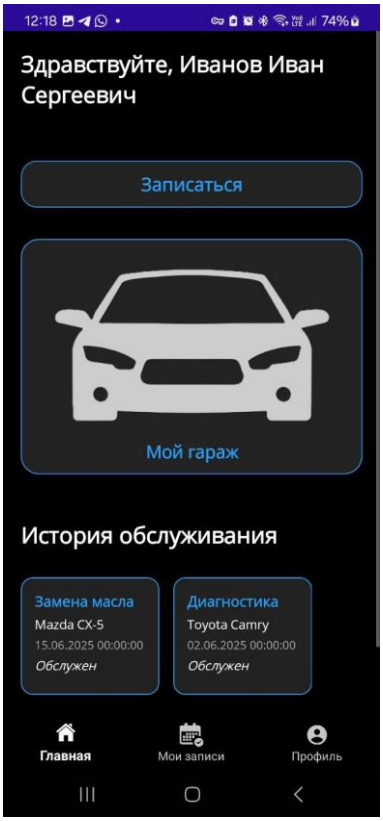

Рисунок Ж.6 – Главный экран при успешном входе в систему

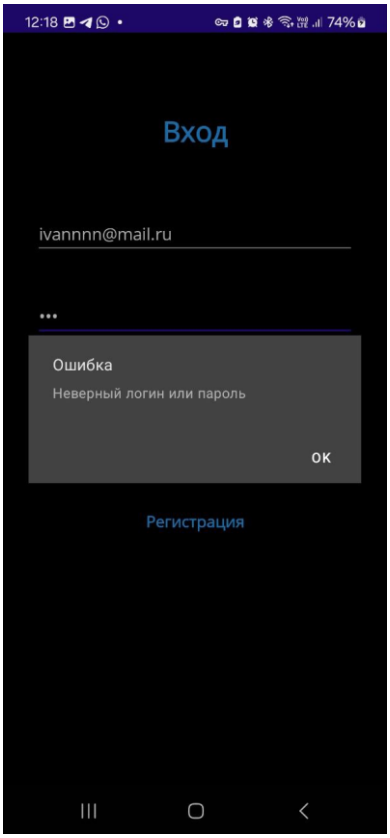

Рисунок Ж.7 – Неверный пароль

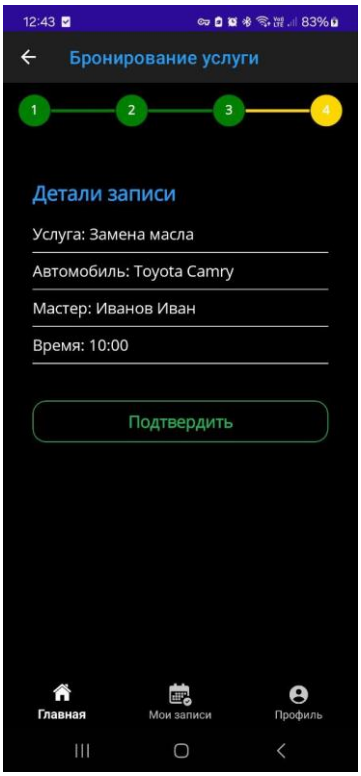

Рисунок Ж.8 – Успешная запись на услугу

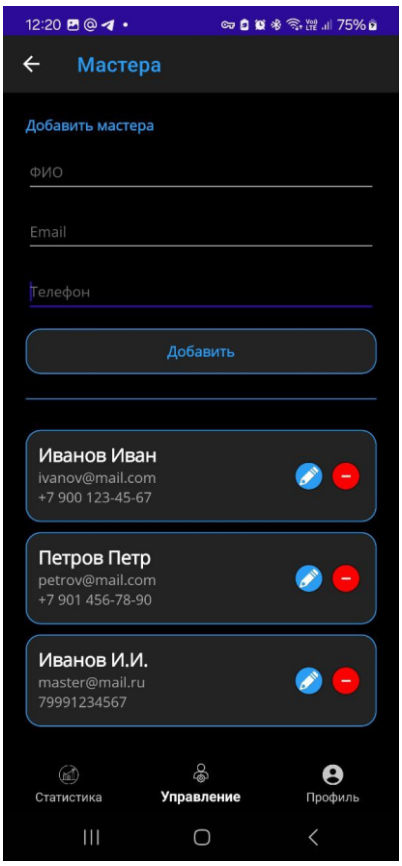

Рисунок Ж.9 – Успешное добавление нового мастера

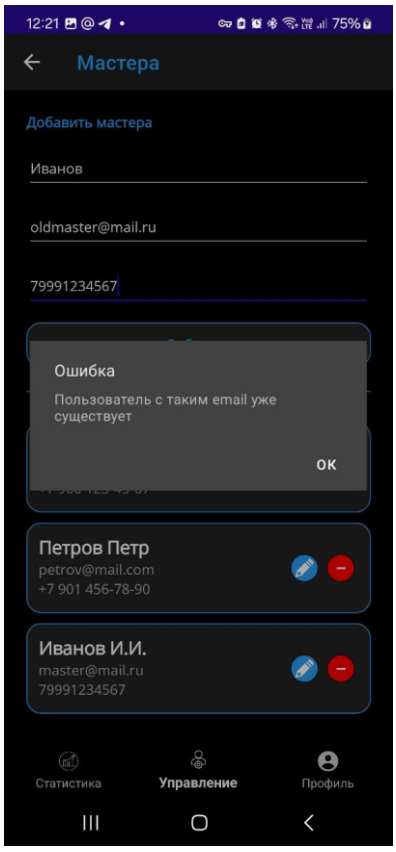

Рисунок Ж.10 – Попытка добавления мастера с уже использованным email

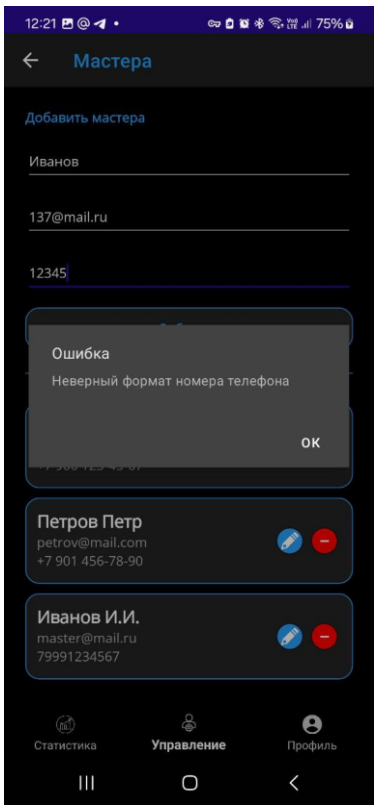

Рисунок Ж.11 – Некорректный формат номера телефона

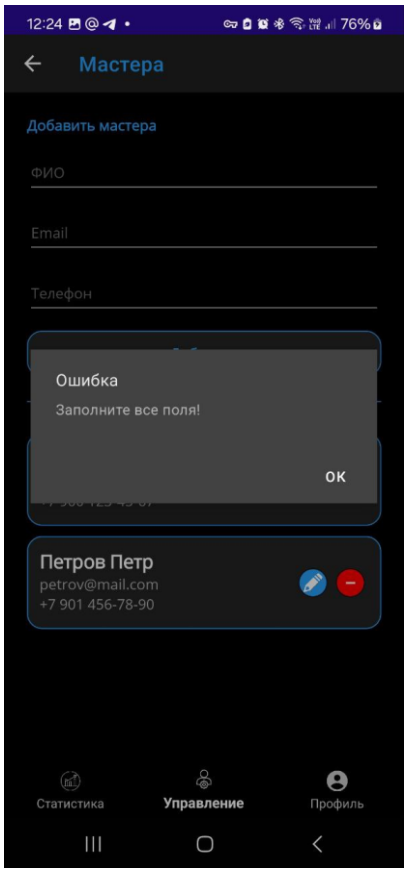

Рисунок Ж.12 – Незаполненные обязательные поля