

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

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

**Мобильное программное приложение для
поддержки процессов регистрации дорожно-
транспортных происшествий**

Студент _____
(подпись, дата) _____
(ФИО полностью) **Нешкин Даниил Андреевич**

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

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

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

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

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

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

Пенза, 2024

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

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов
« 25 » декабря 2023 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Нешкин Даниил Андреевич гр. 20ВП1 факультета ВТ

<Фамилия, имя, отчество полностью>

2. Тема работы Мобильное программное приложение для поддержки процессов регистрации дорожно-транспортных происшествий

Тема утверждена приказом ПГУ № _____ от “__” _____ 20__

3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Князев В.Н.

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

1) Разработать приложение для мобильных устройств, обеспечивающее поддержку процессов регистрации дорожно-транспортных происшествий.

2) Среда выполнения приложения: операционная система Android 9 и выше
Среда выполнения сервера: операционная система Windows 10 и выше
Языковые средства разработки: C#, UML, SQL

Инструментальные средства разработки: Microsoft SQL Server 2019,
Microsoft Visual Studio 2022, Microsoft Project, draw.io
Технологии разработки: RUP, ООП, Entity Framework, .net MAUI

3) Основные функции мобильного приложения:
а) оформление электронного европротокола;
б) отправка электронного европротокола в страховую компанию;
в) просмотр оформленных электронных европротоколов;
г) регистрация пользователей с разными ролями;
д) просмотр правил ПДД

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области и постановка задачи
2) Проектирование
3) Реализация

4) *Контроль качества программного обеспечения*

5) *Оформление пояснительной записки*

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	<i>Анализ предметной области постановка задачи</i>	10%	08.04.2024– 13.04.2024	
2	<i>Проектирование</i>	20%	15.04.2024– 25.04.2024	
3	<i>Реализация</i>	30%	26.04.2024– 12.05.2024	
4	<i>Контроль качества программного обеспечения</i>	20%	13.05.2024– 22.05.2024	
5	<i>Оформление пояснительной записки</i>	20%	23.05.2024– 03.06.2024	

Дата выдачи задания «21» декабря 2023г.

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 195 листов, 83 рисунка, 13 таблиц, 33 использованных источников, 6 приложений.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ЭЛЕКТРОННЫЙ ЕВРОПРОТОКОЛ, ANDROID, MS SQL SERVER, C#, RUP, UML, ЭЛЕКТРОННАЯ ПОДПИСЬ

Объектом разработки является мобильное приложение для поддержки процессов регистрации дорожно-транспортных происшествий.

Цель работы – разработка мобильного приложения для оформления электронного европротокола. Данное приложение позволит оформлять европротокол в электронном виде, а также отправлять его в страховую компанию.

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# при помощи мультиплатформенного фреймворка .NET MAUI. В качестве среды разработки использовалась среда программирования Microsoft Visual Studio 2022, операционная система Windows 10. Приложение предназначено для использования под управлением ОС Android версии 9.0 и выше, а его серверная часть работает под управлением операционной системы **Windows 10** и выше.

					ПГУ 09.03.04 – 08BP201.23 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Мобильное программное приложение для поддержки процессов регистрации дорожно-транспортных происшествий. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Нешкин Д.А.					4	
Пров.		Князев В.Н.						
Н. контр.		Такташкин Д.В.				Группа 20ВП1		
Утверд.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Анализ предметной области и требований.....	10
1.1 Анализ предметной области и постановка задачи.....	10
1.2 Анализ программных продуктов аналогичного назначения	11
1.2.1 Мобильное приложение «Госуслуги Авто»	11
1.2.2 Мобильное приложение «ДТП. Европротокол»	12
1.2.3 Мобильное приложение «Помощник ОСАГО»	15
1.2.4 Сравнение аналогов	16
1.3 Анализ целевой аудитории.....	17
1.4 Анализ функциональных требований	20
1.5 Анализ нефункциональных требований	26
1.6 Планирование разработки и оценка бюджета	27
2 Проектирование мобильного приложения	37
2.1 Выбор технологий и инструментальных средств разработки	37
2.2 Проектирование архитектуры программного обеспечения.....	38
2.3 Проектирование интерфейса пользователя	40
2.4 Логическое проектирование базы данных.....	46
2.5 Проектирование серверной части приложения.....	50
3 Реализация мобильного приложения	51
3.1 Реализация базы данных.....	51
3.2 Разработка серверной части	52
3.3 Реализация клиентской части приложения	54
4 Тестирование программного продукта	58
4.1 Тестирование серверной части приложения	58
4.2 Тестирование клиентской части приложения	59
4.2.1 Функциональное тестирование.....	59
4.2.2 Модульное тестирование	62
4.3 Метрики кода.....	64
4.3.1 Метрики размера программы.....	65
4.3.2 Метрики Холстеда	66

Закключение	68
Список использованных источников	69
Приложение А (справочное) Глоссарий	73
Приложение Б (рекомендуемое) Исходный код скриптов базы данных	75
Приложение В (рекомендуемое) Исходный код серверной части.....	79
Приложение Г (рекомендуемое) Исходный код мобильного приложения...	103
Приложение Д (рекомендуемое) Результаты тестирования серверной части.....	168
Приложение Е (рекомендуемое) Результаты тестирования мобильного приложения	178

Введение

В современном мире информационные технологии широко проникают в различные области человеческой деятельности. Каждый год компьютеры все активнее используются по всему миру в различных отраслях экономики, что существенно изменяет жизнь людей.

Наша страна не остается в стороне от этой тенденции и активно внедряет информационные технологии во всех сферах деятельности на государственном уровне. 9 мая 2017 года была принята «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы», которая определяет основные цели и задачи политики в сфере информационных технологий и направлена на формирование национальной цифровой экономики [1]. Также в 2017 году была запущена государственная программа «Цифровая экономика», целью которой является развитие цифровой экономики, повышение конкурентоспособности страны, улучшение качества жизни граждан и обеспечение экономического роста. Одной из ключевых особенностей развития информационных технологий является все более широкое распространение мобильных устройств с доступом к интернету, которые постепенно вытесняют стационарные компьютеры благодаря своей удобности. Подтверждением служит исследование в рамках программы «Цифровая экономика», согласно которому в 2016 году на территории нашей страны «на 100 человек приходилось 159,95 мобильного телефона и из 100 человек 71,29 человека использовали мобильный доступ к сети Интернет» [2]. Со временем эта тенденция развивается все сильнее.

В связи с этим для большинства веб-сайтов пользователи сотовых устройств стали немаловажную роль, вытесняя пользователей персональных компьютеров на задний план. Поэтому создатели интернет-ресурсов в первую очередь стараются подстроить свои продукты под мобильные платформы, что и привело к активному росту мобильных приложений в наши дни.

Информационные технологии также затронули сферу дорожно-транспортных происшествий. Причиной тому стало появление альтернативы бумажным европротоколам в лице их электронных версий. Электронные протоколы имеют ряд преимуществ над классическими. Бумажные версии занимают много места, что не позволяет оперативно работать с ними. Цифровой же вариант может быть заполнен или прочитан в любом удобном для человека месте. Обязательным условием для работы с таким документом является только наличие ноутбука, планшета или любого другого портативного устройства.

Сфера дорожно-транспортных происшествий, как и любая другая, должна идти в ногу со временем, адаптируя цифровые решения в свою работу. Использование электронных европротоколов при оформлении дорожно-транспортных происшествий имеет высокую актуальность в современном мире. Электронные формы европротоколов позволяют ускорить процесс оформления ДТП и снизить бюрократическую нагрузку на участников. Они обеспечивают более быстрое и эффективное взаимодействие страховых компаний и участников ДТП, исключая при этом ГАИ, что обеспечивает меньшую нагруженность данного ведомства. Электронные европротоколы способствуют сокращению времени на рассмотрение и урегулирование страховых случаев, что повышает удобство для всех сторон. Они также способствуют уменьшению риска мошенничества и фальсификации данных при оформлении ДТП. Система электронных европротоколов обеспечивает более точное и надежное хранение информации о происшествиях. Электронные европротоколы способствуют повышению прозрачности и доступности информации о ДТП для всех заинтересованных сторон. Они упрощают процесс ведения статистики по дорожно-транспортным происшествиям и позволяют быстрее выявлять тенденции и проблемные участки на дорогах. В целом, использование электронных европротоколов при оформлении ДТП способствует повышению эффективности работы системы урегулирования страховых случаев на дорогах.

Важность внедрения информационных технологий в сферы государственного управления была признана в том числе и на государственном уровне 7 мая 2024 года вышел «Указ о национальных целях развития Российской Федерации на период до 2030 года и на перспективу до 2036 года». Согласно указу, необходимо выполнить «увеличение к 2030 году до 95 процентов доли использования российского программного обеспечения в государственных органах, государственных корпорациях, государственных компаниях и хозяйственных обществах, в уставном капитале которых доля участия Российской Федерации в совокупности превышает 50 процентов, а также в их аффилированных юридических лицах» [3]. Поэтому одной из главных задач органов государственного управления является их модернизация. Для её обеспечения необходима «цифровизация процессов предоставления государственных услуг и исполнения государственных функций государственными органами власти», то есть замена бумажных документов и протоколов на их электронные аналоги [4].

Таким образом, на основании приведенных выше факторов можно сделать вывод о том, что разработка мобильного приложения для оформления электронного европротокола весьма актуальна и перспективна в современном обществе.

1 Анализ предметной области и требований

1.1 Анализ предметной области и постановка задачи

Разрабатываемое мобильное приложение должно предоставлять максимально удобный процесс оформления и отправки европротокала. Благодаря ему пользователям не потребуется обязательно иметь при себе бумажный европротокол. Достаточно будет иметь мобильное устройство с установленным приложением и подключением к сети интернет. Это позволит пользователям получить доступ к электронной версии европротокола, что предоставит возможность оформлять его параллельно двум людям и отправлять сразу же с места происшествия в страховую компанию.

Приложение должно позволять всем зарегистрированным пользователям оформлять и отправлять электронный европротокол. Также любому пользователю должна быть предоставлена возможность редактировать свои персональные данные.

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

- регистрация новых пользователей;
- вход зарегистрированных пользователей в систему;
- просмотр персональной информации;
- редактирование персональных данных;
- оформление электронного европротокола;
- отправка электронного электропротокола.

Помимо рядовых пользователей в системе будут присутствовать администраторы и представители страховых компаний, которые будут иметь иной набор возможностей. Администраторам будет предоставляться возможность добавлять и удалять правила дорожного движения и бланки электронных европротоколов. Представители же страховых компаний смогут просматривать полный список извещений и каждое электронный европротокол отдельно.

1.2 Анализ программных продуктов аналогичного назначения

В настоящее время существует несколько приложений для оформления электронного европротокола. Рассмотрим наиболее известные интернет-ресурсы такого рода и произведем для них сравнительный анализ.

1.2.1 Мобильное приложение «Госуслуги Авто»

В рамках нацпрограммы «Цифровая экономика» в сентябре 2021 года для пользователей по всей стране заработало приложение «Госуслуги Авто». С момента запуска его скачали уже более 1,6 млн раз. Это технологическое решение призвано стать единым окном, аккумулирующим самые востребованные сервисы для автовладельцев. Главной функцией приложения стала возможность загрузить свидетельство о регистрации транспортного средства и предъявлять его инспектору в электронном виде. В августе 2022 году у пользователей появилась возможность оформлять Европротокол онлайн.

Главная страница приложения Госуслуги Авто представлена на рисунке 1.

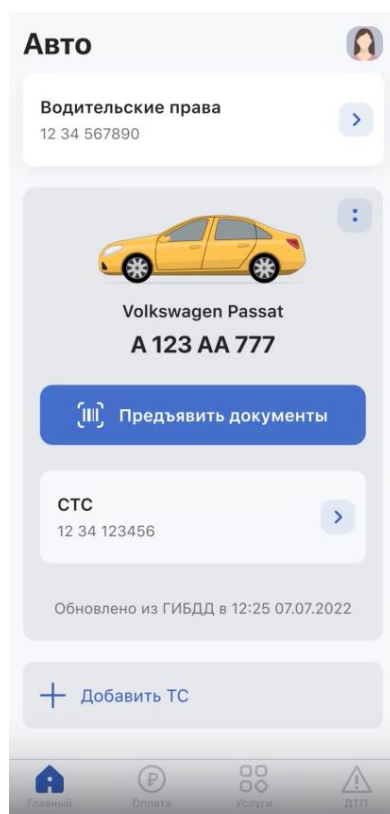


Рисунок 1 – Главная страница приложения Госуслуги Авто

Приложение «Госуслуги Авто» начало работать в 2021 году [5]. Оно позволяет:

- хранить СТС и предъявлять его в электронном виде инспектору ГАИ;
- передавать электронный СТС другому человеку;
- получать уведомления и оплачивать штрафы онлайн;
- получать выписку из реестра транспортных средств с информацией об автомобиле.

Мобильное приложение предоставляет достаточно удобный и понятный интерфейс для заполнения электронного европротокола. Продукт позволяет заполнять электронное извещение параллельно двум участникам сразу, что ускоряет процесс. Кроме того, Госуслуги Авто гарантируют надежную отправку данных.

Недостатком данного мобильного приложения является отсутствие электронной подписи пользователя, подтверждающая его личность. Вместо этого в приложении реализована кнопка «Подписать», которая не имеет ничего общего в настоящей электронной подписью человека.

1.2.2 Мобильное приложение «ДТП. Европротокол»

Российский союз автостраховщиков (далее – РСА) не так давно представил мобильное приложение ДТП.Европротокол, оформление ДТП в котором происходит через автоматическую систему, а участие ГАИ для фиксации аварийного происшествия не требуется. Теперь документы передаются в единую систему, откуда доступ к ним получают страховщики. Также в общую базу попадают координаты аварии, фотографии места происшествия и повреждений, полученных транспортным средством. Чтобы получить максимально возможную выплату по европротоколу (400 000 рублей), необходимо зафиксировать факт ДТП с помощью данного приложения или через систему ГЛОНАСС. Также у участников происшествия не должно быть разногласий относительно обстоятельств ДТП [6].

Главный экран приложения представлен на рисунке 2.

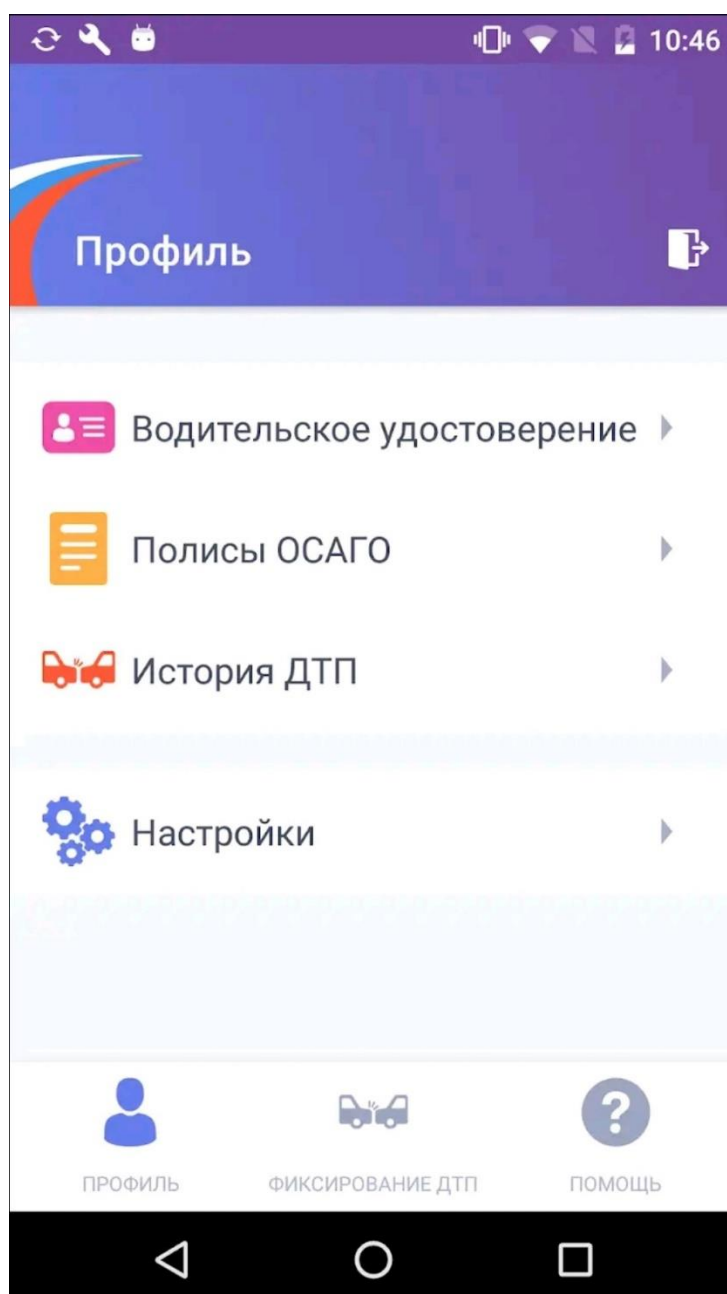


Рисунок 2 – Главный экран мобильного приложения «Европротокол.ДТП»

С помощью приложения Европротокол оформление ДТП станет более быстрым и удобным. В личный кабинет можно добавить несколько полисов, а при оформлении дорожно-транспортного происшествия – выбрать нужный из ранее добавленных. Схема автомобиля для выбора поврежденных деталей интерактивная, доступен большой перечень частей ТС. Кроме того, авторизоваться можно через Госуслуги, а также дополнительно установить защитный пин-код или сканирование отпечатка пальца [7].

В приложении «Европротокол.ДТП» так же можно оформлять и отправлять электронный европротокол. Главным достоинством продукта является интерфейс. Он удобен и понятен для пользователя, что облегчает его использование. Приложение позволяет оформлять электронное извещение параллельно для каждого участника.

На этом положительные качества приложения заканчиваются и начинаются отрицательные. Главным недостатком продукта является отсутствие цифровизации европротокола, то есть для оформления электронного извещения необходимо сначала заполнить бумажный экземпляр, а потом сделать его фотографию. Следующим недостатком данного приложения является отсутствие гарантий передачи данных в страховую компанию. Подтверждением тому служит одно из условий соглашения использования продукта: «По настоящему Соглашению право использования Мобильного приложения предоставляется на условиях «как есть». РСА не предоставляет никаких гарантий и заверений в отношении безошибочной работы(функционирования) Мобильного приложения или отдельных его частей/материалов (самостоятельных информационных элементов), соответствия Мобильного приложения конкретным целям и ожиданиям Пользователя, не гарантирует достоверность, точность и полноту сведений(материалов), содержащихся в Мобильном приложении, а также не предоставляет никаких иных гарантий или заверений в отношении Мобильного приложения. РСА не несет ответственность за какие-либо последствия, связанные с использованием, неиспользованием или невозможностью использования Мобильного приложения или отдельных его частей/материалов (самостоятельных информационных элементов), а также не несет ответственность за прямые или косвенные убытки, причиненные Пользователю или иным третьим лицам в результате использования, неиспользования или невозможности использования Мобильного приложения или отдельных его частей/материалов (самостоятельных информационных элементов), в том числе из-за ошибок или сбоев в

работе(функционировании) Мобильного приложения, за исключением случаев, предусмотренных законодательством Российской Федерации» [8].

1.2.3 Мобильное приложение «Помощник ОСАГО»

«Помощник ОСАГО» – это мобильное приложение, созданное Банком России и Российском Союзом Автостраховщиков (РСА) для электронного урегулирования убытка без визита в офис страховщика [9].

Главная страница приложения приведена на рисунке 3.



Рисунок 3 – Главная страница приложения «Помощник.ОСАГО»

В настоящее время доступны следующие функции:

- оформление извещения о ДТП в виде электронного документа;
- фотосъемка транспортных средств и их повреждений на месте ДТП и передача этих данных РСА, откуда они поступают в страховую компанию.

Основным достоинством приложения является возможность заполнения электронного европротокола сразу двумя участниками одновременно, что ускоряет сам процесс оформления. Для объединения данных пользователи используют QR-код. Продукт имеет удобный и понятный интерфейс, что облегчает заполнение электронного европротокола. Приложение гарантирует сохранность данных при их передаче в РСА.

Главным же недостатком «Помощника ОСАГО» является отсутствие электронной подписи, идентифицирующей личность пользователя. Это означает, что такой электронный документ не будет иметь юридической силы при суде или при любых других государственных разбирательствах.

1.2.4 Сравнение аналогов

Сравнительный анализ рассмотренных выше приложений для оформления электронных протоколов представлен в таблице 1.

Таблица 1 – Сравнение аналогов

Название приложения	Удобный и понятный интерфейс	Европротокол могут заполнять два пользователя	Данные передаются надежно	Есть электронная подпись
Госуслуги Авто	Да	Да	Да	Нет
Европротокол.ДТП	Да	Нет	Нет	Нет
Помощник ОСАГО	Да	Да	Да	Нет
Разрабатываемое приложение	Да	Да	Да	Да

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

1.3 Анализ целевой аудитории

Целевой аудиторией проекта являются водители транспортных средств, использующие мобильные устройства. В настоящее время число пользователей, которые оформляют дорожно-транспортные происшествия при помощи электронных протоколов, стабильно растет с каждым годом. Подтверждением этой информации может послужить статистика изменения числа пользователей мобильного приложения «Госуслуги Авто». График, на котором показана динамика изменения числа пользователей с течением времени, приведен на рисунке 4.

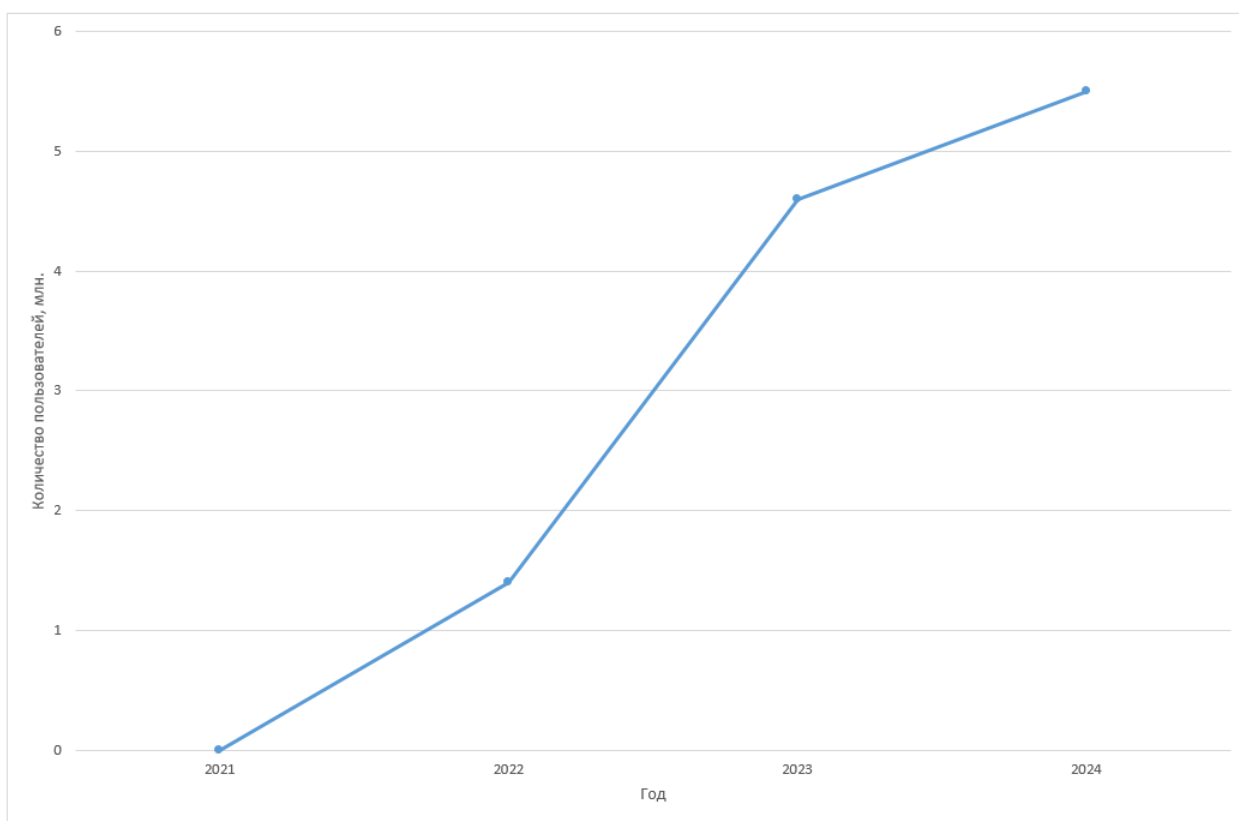


Рисунок 4 – Изменение числа пользователей мобильного приложения «Госуслуги Авто»

Приложение «Госуслуги Авто» было запущено в России в 2021 году, оно установлено у 5,5 млн водителей. С того момента в его базу загрузили 4 млн электронных копий свидетельств о регистрации транспортного средства (СТС) и 2,5 млн водительских удостоверений. Об этом сообщили в пресс-службе Минцифры РФ [10]. Приложение помогло более 26 тыс. раз – оформить ДТП онлайн, более 3 тыс. раз – подать заявление на возмещение

убытков по ОСАГО после ДТП. Оформление ДТП по европротоколу в приложении занимает в среднем 30 минут [11]. Как можно увидеть всего за 3 года «Госуслуги Авто» набрало огромную популярность, что подтверждает значимость подобных приложений среди автолюбителей.

Согласно исследованию Canalys, компания Samsung отгрузила больше всех смартфонов, а ее модели занимают 20% рынка (22% в III квартале 2022 года). Второе место в ТОП-5 производителей смартфонов занимает Apple с долей 17% (18% в 2022 году). Третья строчка – Xiaomi – 14%, как и год назад. Четвертое место у OPPO (включая OnePlus) – 9%, что на 1% меньше, чем годом ранее. Долю в 9% рынка также имеет производитель TRANSSION, известный по брендам iTel Mobile, Tecno Mobile и Infinix Mobile [12].

График, демонстрирующий динамику изменения доли производителей мобильных устройств на рынке в течение 2021 – 2023 годов, показан на рисунке 5.

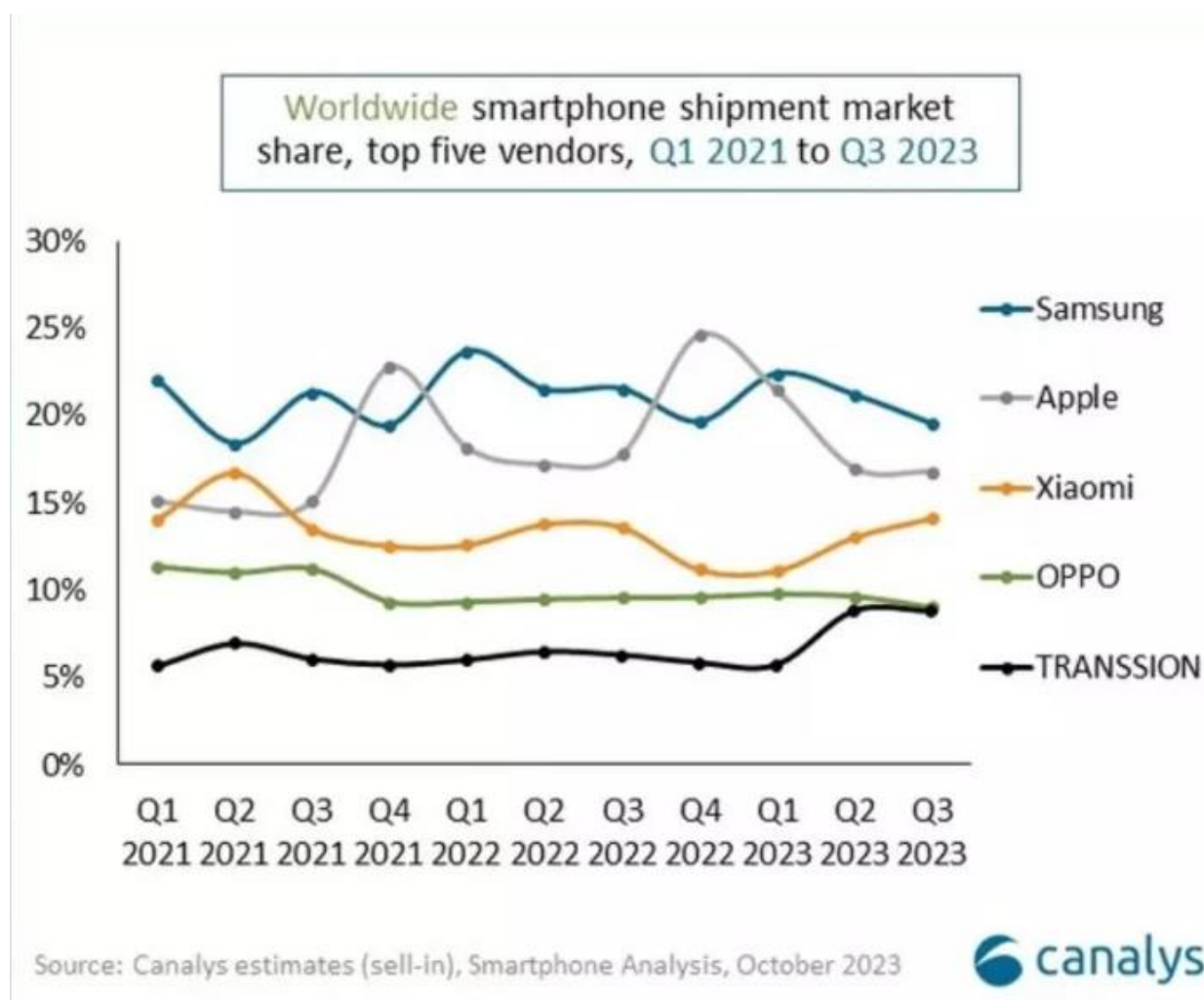


Рисунок 5 – Изменение доли производителей мобильных устройств на рынке

Можно заметить, что большая часть производителей мобильных устройств – Samsung, Xiaomi, OPPO, и др. – используют операционную систему Android. Таким образом, можно сделать вывод, что доля рынка на смартфоны с операционной системой Android составляет около 83%.

Единственным производителем с другой ОС является Apple, который использует собственную систему – iOS. Однако, при всей популярности компании Apple, ее продукция даже в периоды пикового спроса составляет лишь 25% от мирового рынка, что в несколько раз меньше, чем доля устройств с Android. Доля же Apple на российском рынке ещё меньше. По данным Big Data МТС, с которыми ознакомился Forbes, в 2023 году больше всего в сети оператора работало смартфонов китайского производителя Xiaomi, за год прибавившего 0,8 п. п. – 24,3% от всех телефонов. Ранее лидером по проникновению в сети МТС был Samsung – по итогам 2022 года он занимал 25,7%, но в 2023-м доля южнокорейского производителя сократилась до 22,7%.

Apple сохранила третье место и, более того, увеличила долю с 20,7% в 2022 до 22,3% по итогам 2023-го. Китайские бренды Realme, Tecno и Infinix за 2023 год нарастили долю с 5,4% до 11,2% от всех устройств, зарегистрированных в сети МТС. По данным «Вымпелкома» (бренд билайн), больше всего в сети оператора в 2023 году было телефонов Samsung, при этом доля вендора за прошлый год снизилась с 23,2% до 20,6%. На втором месте — Xiaomi с 17,5% (его смартфоны «приросли» на 0,3 п. п. за год). А на третьем месте в сети «Вымпелкома» в 2023 году, так же как и у МТС, находится Apple с долей 15% (годом ранее доля вендора составляла 13,5%). Напомним, у МТС самая большая абонентская база среди российских операторов — 80 млн по итогам 2022 года. У «Мегафона» она составляла 75,3 млн человек, у «Вымпелкома» — 44,8 млн, а у Tele2 — 47,5 млн пользователей [13].

Статистика распространения разных версий системы Android представлена на рисунке 6 [14].

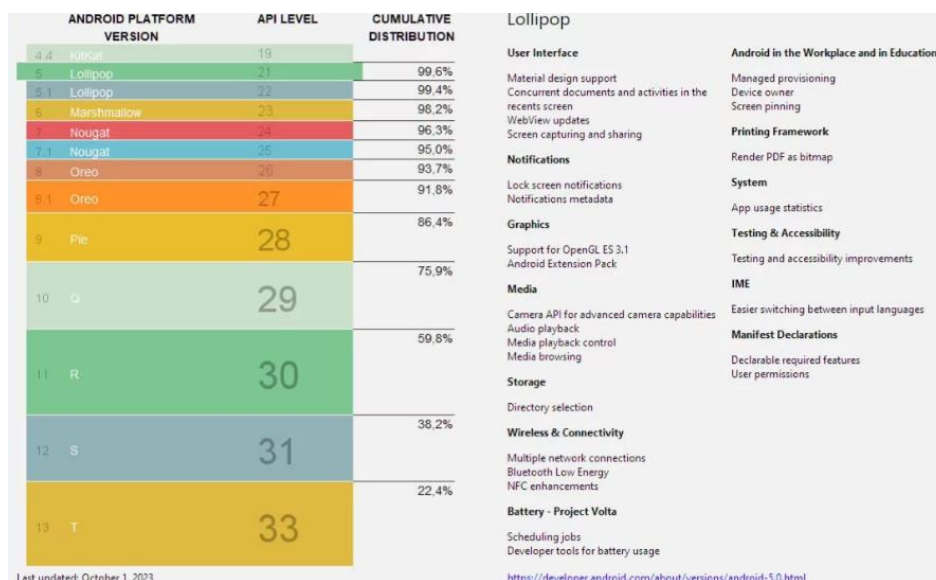


Рисунок 6 – Распространение различных версий системы Android

Можно увидеть, что 86,4% пользователей ОС Android использует версии не старше 9.0. Поэтому приложение будет разрабатываться для операционных систем Android 9.0 и выше.

1.4 Анализ функциональных требований

Основными объектами системы будут являться европротоколы. Для каждого из них будет храниться следующая информация: основные данные ДТП (дата и время, схема, описание и так далее), персональные данные участников и данные их транспортных средств.

В системе должно быть три основных типа пользователей: обычные пользователи, администраторы и представители страховых компаний. Всем пользователям система должна предоставлять возможность регистрации и авторизации.

Обычный пользователь должен иметь возможность оформлять и отправлять электронный европротокол, а также редактировать свои персональные данные.

Представитель страховой компании должен иметь возможность просматривать как списки электронных европротоколов, так и конкретное извещение из списка.

Администраторы должны иметь все возможности обычного пользователя и представителя страховой компании. Кроме того, администратору должно быть позволено добавлять и удалять правила ПДД и бланки электронных европротоколов.

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

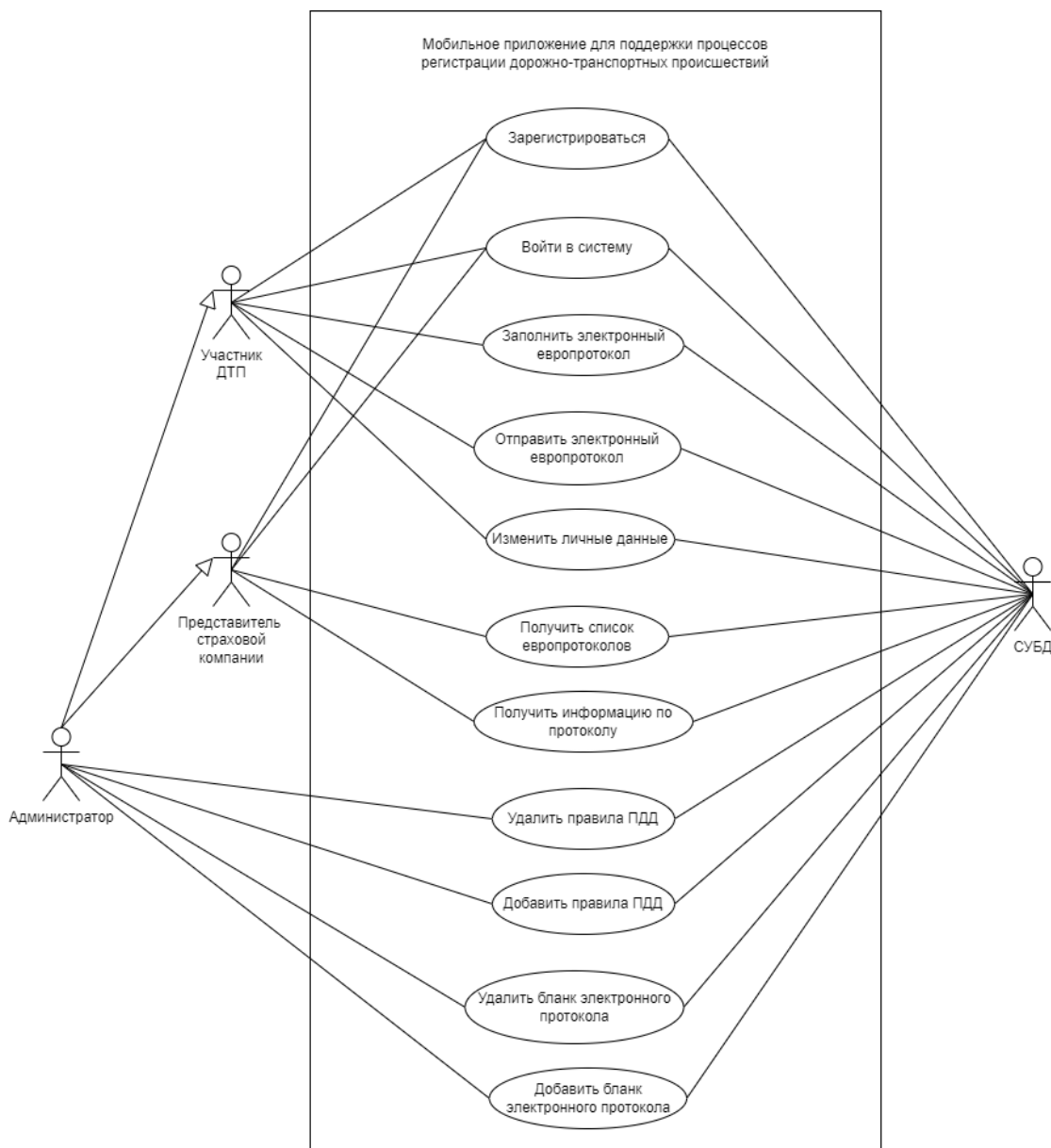


Рисунок 7 – Диаграмма вариантов использования мобильного приложения

Составим спецификации основных вариантов использования: «Зарегистрироваться», «Войти в систему», «Заполнить электронный европротокол», «Отправить электронный европротокол», «Изменить личные данные», «Получить список европротоколов».

Спецификация варианта использования «Зарегистрироваться» приведена на рисунке 8.

Название: зарегистрироваться
Назначение: регистрация нового аккаунта пользователя
Актеры: участник ДТП, представитель страховой компании, СУБД
Предусловие: приложение запущено
Основной сценарий: <ol style="list-style-type: none"> 1. Участник ДТП или представитель страховой компании выбирает регистрацию. 2. Участник ДТП или представитель страховой компании вводит логин и пароль. 3. Система проверяет логин и пароль на корректность. 4. Если логин и пароль корректны. <ol style="list-style-type: none"> 4.1. Система запрашивает у СУБД сведения о наличии аккаунта с введенным логином. 4.2. СУБД передает сведения о наличии аккаунта с введенным логином. 4.3. Если такого аккаунта нет <ol style="list-style-type: none"> 4.3.1. СУБД добавляет в БД информацию о новом пользователе. 4.3.2. Система осуществляет вход нового пользователя. 4.4. Иначе система выводит сообщение о том, что выбранный логин занят. 5. Иначе система выводит сообщение о недопустимом логине или пароле.
Постусловие: регистрация выполнена

Рисунок 8 – Спецификация варианта использования «Зарегистрироваться»

Спецификация варианта использования «Отправить электронный европротокол» приведена на рисунке 9.

Название: отправить электронный европротокол
Назначение: отправка электронного европротокола
Актеры: участник ДТП, СУБД
Предусловие: участники ДТП заполнили электронный европротокол
Основной сценарий: <ol style="list-style-type: none"> 1. Участник ДТП выбирает отправить электронный европротокол. 2. Система передает данные электронного европротокола в СУБД. 3. СУБД добавляет данные
Постусловие: электронный европротокол успешно сохранен

Рисунок 9 – Спецификация варианта использования «Отправить электронный европротокол»

Спецификация варианта использования «Войти в систему» представлена на рисунке 10.

Название: войти в систему
Назначение: вход в систему с существующего аккаунта пользователя
Актеры: участник ДТП, представитель страховой компании, СУБД
Предусловие: приложение запущено
Основной сценарий: <ol style="list-style-type: none"> 1. Участник ДТП или представитель страховой компании выбирает авторизацию 2. Участник ДТП или представитель страховой компании вводит логин и пароль. 3. Система запрашивает у СУБД сведения о наличии аккаунта с введенным логином и паролем. 4. СУБД передает сведения о наличии аккаунта с такими логином и паролем 5. Если такой аккаунт существует <ol style="list-style-type: none"> 5.1. Система проверяет наличие у него активной блокировки. 5.2. Если активной блокировки нет <ol style="list-style-type: none"> 5.2.1. Система осуществляет вход нового пользователя. 5.3. Иначе система сообщает о наличии у пользователя блокировки. 6. Иначе система выводит сообщение о неверном логине или пароле.
Постусловие: вход в систему выполнен

Рисунок 10 – Спецификация варианта использования «Войти в систему»

Спецификация варианта использования «Изменить личные данные» приведена на рисунке 11.

Название: изменить личные данные
Назначение: изменение персональных данных
Актеры: участник ДТП, СУБД
Предусловие: участники ДТП вошел в свой личный кабинет
Основной сценарий: <ol style="list-style-type: none"> 2. Участник ДТП изменяет персональные данные в соответствующей окне. 3. Участник ДТП отправляет запрос на изменение. 4. Если новые данные корректны. <ol style="list-style-type: none"> 4.1. СУБД заменяет старые данные на новые. 5. Иначе система выводит сообщение о некорректности данных.
Постусловие: данные участника ДТП успешно обновлены

Рисунок 11 – Спецификация варианта использования «Изменить личные данные»

Спецификации вариантов использования «Заполнить электронный европротокол» и «Получить список европротоколов» представлены на рисунках 12 и 13 соответственно.

Название: заполнить электронный европротокол
Назначение: заполнение электронного европротокола для отправки
Актеры: участник ДТП, СУБД
Предусловие: участник ДТП авторизован
Основной сценарий: 1. Участник ДТП выбирает оформить ДТП. 2. Система запрашивает бланк электронного европротокола. 3. Участник ДТП вводит необходимую информацию в поля. 4. Участник ДТП ждет пока свою информацию заполнит второй участник. 5. Участники заполнили информацию. 6. Система проверяет корректность введенных данных. 7. Если все данные введены корректно. 7.1. Система объединяет данные в электронный европротокол. 7.2. Система предоставляет электронный европротокол участникам. 7.3. Если участники подписали электронный европротокол. 7.3.1. Электронный европротокол отправляется в страховую компанию. 7.4. Иначе участники возвращаются к редактированию своих данных. 8. Система выводит сообщение о некорректности данных.
Постусловие: электронный европротокол успешно заполнен

Рисунок 12 – Спецификация варианта использования «Заполнить электронный европротокол»

Название: получить список европротоколов
Назначение: получение информации о имеющихся европротоколах
Актеры: представитель страховой компании, СУБД
Предусловие: представитель страховой компании вошел в систему
Основной сценарий: 1. Представитель страховой компании перешел на страницу списка. 2. Представитель страховой компании запросил данные у системы. 3. Система запросила у СУБД список имеющихся европротоколов. 4. Система получила список. 5. Система отобразила список.
Постусловие: система отобразила список имеющихся европротоколов

Рисунок 13 – Спецификация варианта использования «Получить список европротоколов»

Рассмотрим некоторые варианты использования конкретнее. Для варианта использования «Заполнить электронный европротокол» была составлена диаграмма деятельности, представленная на рисунке 14.

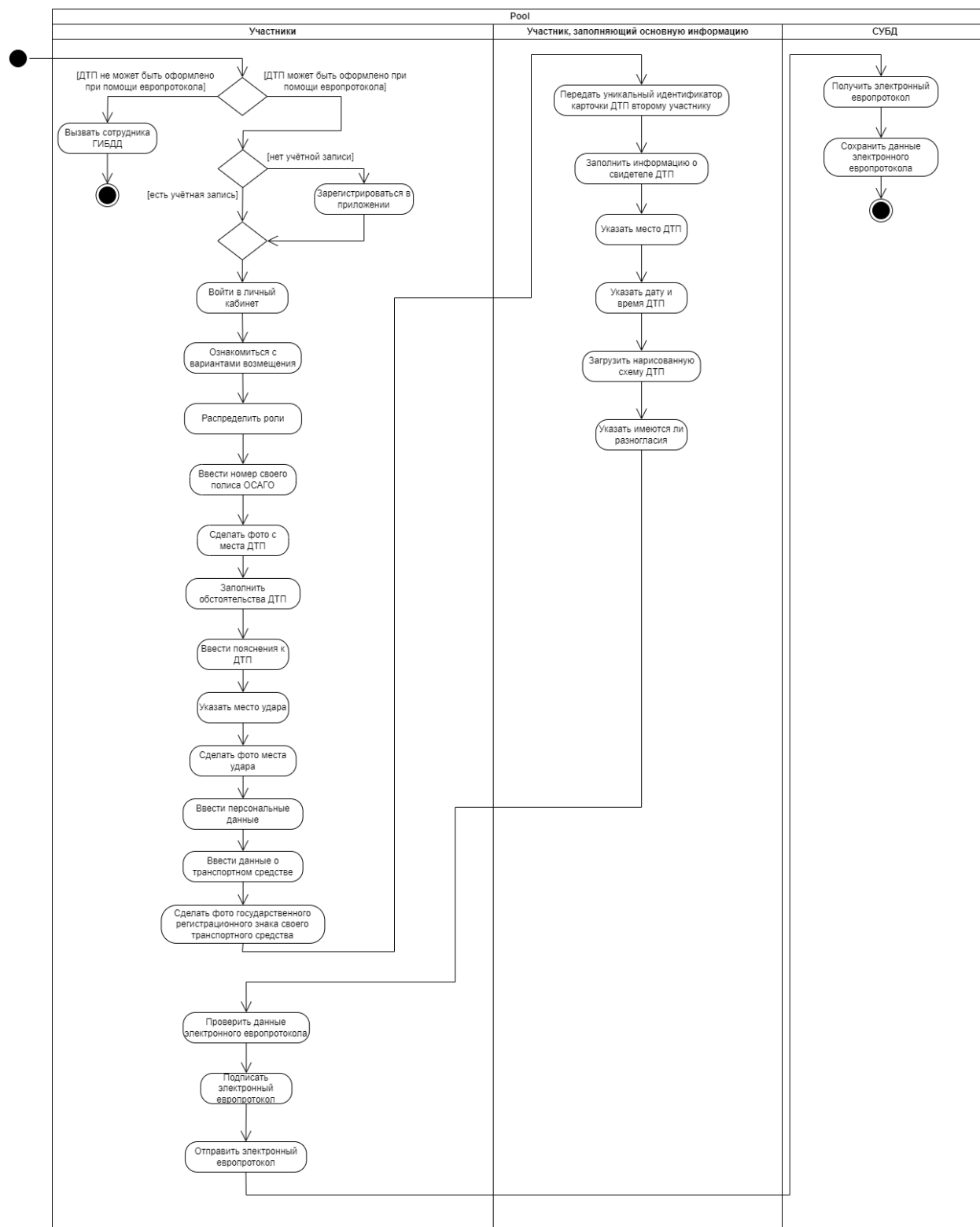


Рисунок 14 – Диаграмма деятельности варианта использования «Заполнить электронный европротокол»

Для варианта использования «Войти в систему» была составлена диаграмма последовательности, представленная на рисунке 15.

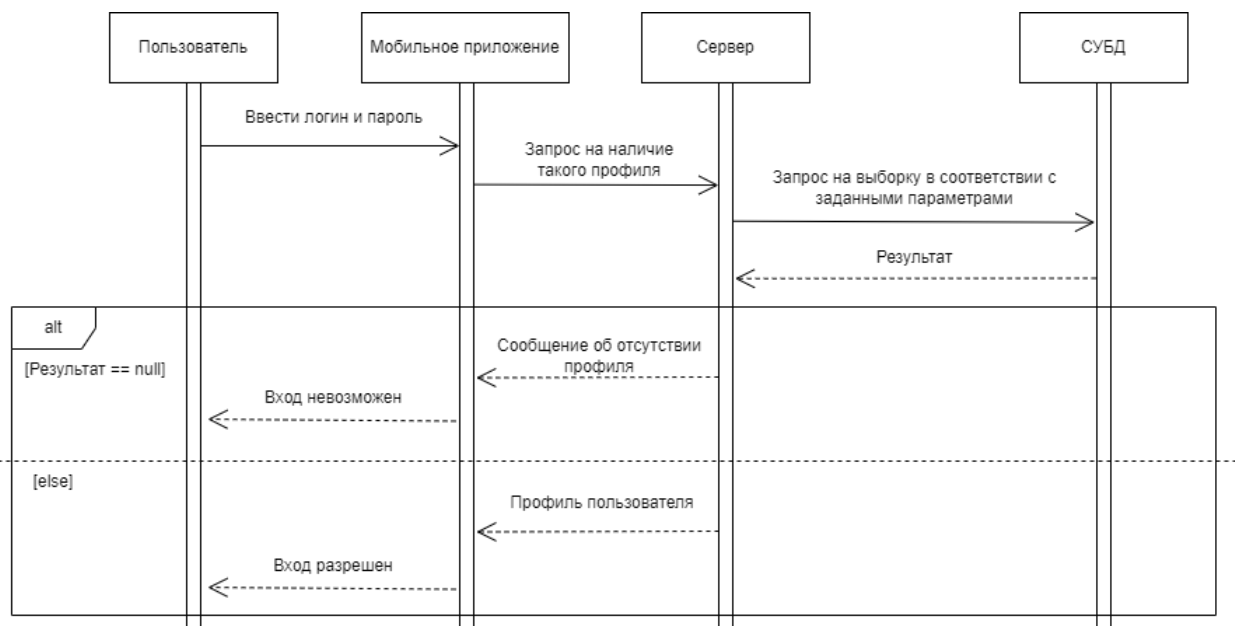


Рисунок 15 – Диаграмма последовательности варианта использования «Войти в систему»

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

1.5 Анализ нефункциональных требований

Серверная часть приложения должна быть выполнена на современном универсальном языке программирования C# с использованием платформы ASP.NET, в среде программирования Microsoft Visual Studio 2022. Данные на сервере должны храниться в базе данных, реализованной при помощи СУБД Microsoft SQL Server, запросы к базе данных осуществляются на языке SQL. Серверная часть должна работать под управлением операционной системы Windows 10 и выше.

Клиентская часть приложения должна быть реализована на языке программирования C# с использованием платформы .NET MAUI, в среде программирования Microsoft Visual Studio 2022. Она должна представлять

собой мобильное приложение с удобным и интуитивно понятным пользовательским интерфейсом. Клиент должен представлять собой мобильное приложение с удобным и понятным пользовательским интерфейсом. Клиентское приложение должно быть отлажено, а также успешно выполняться под управлением ОС Android 9.0 и выше.

1.6 Планирование разработки и оценка бюджета

Качественное выполнение работ по любому проекту, в том числе и в сфере информационных технологий, невозможно при отсутствии грамотного планирования, которое позволяет обеспечить выполнение работ над проектом таким образом, чтобы добиться выполнения всех поставленных целей за минимально возможное время, с минимально возможными затратами и с приемлемым качеством [15 – 19].

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

Таблица 2 – Перечень задач проекта

№	Назначение задачи	Длительность	Предшественники
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Анализ предметной области	3 дня	1
4	Анализ приложений-аналогов	6 дней	3
5	Анализ требований к функционалу	6 дней	4
6	Анализ требований к интерфейсу	6 дней	5
7	Консультация в ГАИ	2 дня	6
8	Постановка задачи	1 день	7
9	Анализ требований завершен	-	8
10	Выбор средств реализации проекта	-	-
11	Анализ возможных программных средств	4 дня	1
12	Выбор подходящего ПО для реализации	1 день	6; 11
13	Выбор средств реализации завершен	-	12
14	Проектирование приложения	-	-
15	Проектирование базы данных	7 дней	9; 13
16	Проектирование архитектуры приложения	7 дней	15
17	Проектирование интерфейса приложения	5 дней	9; 13
18	Проектирование приложения завершенно	-	16; 17
19	Реализация проекта	-	-
20	Реализация базы данных	10 дней	18
21	Реализация серверной части приложения	15 дней	16; 20
22	Реализация клиентской части приложения	9 дней	18
23	Реализация проекта завершена	-	21; 22

Продолжение таблицы 2

№	Назначение задачи	Длительность	Предшественники
24	Тестирование приложения	-	-
25	Тестирование серверной части приложения	10 дней	23
26	Тестирование клиентской части приложения	8 дней	23
27	Исправление ошибок	10 дней	25; 26
28	Тестирование приложения завершено	-	27
29	Внедрение программного продукта	-	-
30	Подготовка документации	2 дня	28
31	Подпись документации государственными органами	3 дня	30
32	Внесение в систему бланка европротокола	1 день	31
33	Внедрение программного продукта завершено	-	32
34	Конец проекта	-	33

Таким образом, для реализации проекта нужно выполнить 34 задачи.

В таблице 3 представлено описание затрачиваемых ресурсов.

Таблица 3 – Список ресурсов проекта

Название ресурса	Тип	Стандартная	Ставка сверхурочных
Программист 1	Трудовой	500 Р/ч	550 Р/ч
Программист 2	Трудовой	500 Р/ч	550 Р/ч
Дизайнер	Трудовой	350 Р/ч	400 Р/ч
Аналитик	Трудовой	300 Р/ч	350 Р/ч
Тестировщик	Трудовой	350 Р/ч	400 Р/ч
Компьютер	Материальный	5000 Р	
Мобильное устройство	Материальный	3000 Р	
Принтер	Материальный	750 Р	
Бумага	Материальный	250 Р	
Интернет	Затраты		
Междугородние переговоры	Затраты		

Распределение ресурсов по задачам представлено в таблице 4.

Таблица 4 – Назначение ресурсов

№	Назначение задачи	Ресурс	Единицы
1	Начало проекта		
2	Анализ требований		
3	Анализ предметной области	Аналитик Компьютер Интернет	100% 1 шт
4	Анализ приложений-аналогов	Аналитик Компьютер Интернет	100% 1 шт

Продолжение таблицы 4

№	Назначение задачи	Ресурс	Единицы
5	Анализ требований к функционалу	Аналитик Программист 1 Компьютер Интернет	100% 50% 1 шт
6	Анализ требований к интерфейсу	Аналитик Дизайнер Компьютер Интернет	100% 100% 1 шт
7	Консультация в ГАИ	Аналитик Междугородние переговоры	50%
8	Постановка задачи	Аналитик Программист 1 Программист 2 Дизайнер	100% 25% 25% 25%
9	Анализ требований завершен		
10	Выбор средств реализации проекта		
11	Анализ возможных программных средств	Программист 1 Программист 2 Компьютер Интернет	50% 50% 2 шт
12	Выбор подходящего ПО для реализации	Программист 1 Программист 2	50% 50%
13	Выбор средств реализации завершен		
14	Проектирование приложения		
15	Проектирование базы данных	Программист 1 Программист 2 Компьютер Интернет	100% 50% 2 шт
16	Проектирование архитектуры приложения	Программист 1 Программист 2 Компьютер Интернет	50% 100% 2 шт
17	Проектирование интерфейса приложения	Дизайнер Компьютер Интернет	100% 1
18	Проектирование приложения завершено		
19	Реализация проекта		
20	Реализация базы данных	Программист 1 Программист 2 Компьютер Интернет	100% 100% 2 шт
21	Реализация серверной части приложения	Программист 1 Программист 2 Компьютер Интернет	100% 100% 2 шт

Окончание таблицы 4

№	Название задачи	Ресурс	Единицы
22	Реализация клиентской части приложения	Программист 1 Дизайнер Компьютер Интернет Мобильное устройство Программист 2	50% 50% 2 шт 1 шт 50%
23	Реализация проекта завершена		
24	Тестирование приложения		
25	Тестирование серверной части приложения	Программист 1 Компьютер	100% 1 шт
26	Тестирование клиентской части приложения	Программист 2 Компьютер Мобильное устройство	100% 1 шт 2 шт
27	Исправление ошибок	Программист 1 Программист 2 Дизайнер Компьютер Мобильное устройство Интернет	100% 100% 100% 2 шт 1 шт
28	Тестирование приложения завершено		
29	Внедрение программного продукта		
30	Подготовка документации	Аналитик Принтер Бумага	100% 1 шт 2 пачки
31	Подпись документации государственными органами	Аналитик Междугородние переговоры	50%
32	Внесение в систему бланка европротокола	Программист 2 Компьютер Интернет	50% 1 шт
33	Внедрение программного продукта завершено		
34	Конец проекта		

Таким образом, реализацией проекта будут заниматься: два программиста, дизайнер, аналитик, тестировщик. Для работы им будут необходимы: компьютеры, мобильные устройства, принтер, бумага, доступ к интернету и деньги на разъезды для междугородних переговоров. Для вычисления длительности реализации проекта и его критического пути построен сетевой график (рисунок 16)

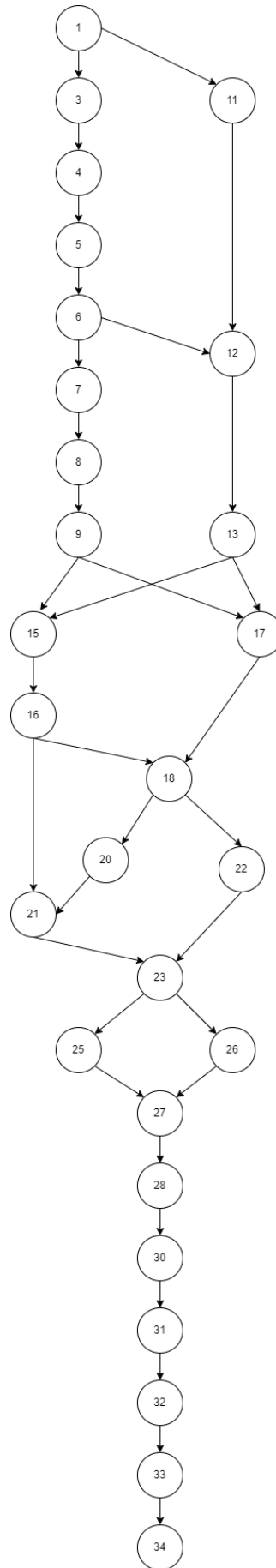


Рисунок 16 – Сетевой график проекта

Для того, чтобы рассчитать продолжительность проекта, а также вычислить величину его бюджета, воспользуемся системой управления проектами Microsoft Project, которая позволяет автоматически произвести все необходимые вычисления [20 – 23]. Список задач из таблицы 2, внесенный в систему Microsoft Project, представлен на рисунке 17. В качестве времени начала проекта была принята дата 10.01.2024.

		Название задачи	Длительность	Начало	Окончание	Предшественн
1		Начало проекта	0 дней	Ср 10.01.24	Ср 10.01.24	
2		Анализ требований	24 дней	Ср 10.01.24	Пн 12.02.24	
3		Анализ предметной области	3 дней	Ср 10.01.24	Пт 12.01.24	1
4		Анализ приложений-аналогов	6 дней	Пн 15.01.24	Пн 22.01.24	3
5		Анализ требований к функционалу	6 дней	Вт 23.01.24	Вт 30.01.24	4
6		Анализ требований к интерфейсу	6 дней	Ср 31.01.24	Ср 07.02.24	5
7		Консультация в ГИБДД	2 дней	Чт 08.02.24	Пт 09.02.24	6
8		Постановка задачи	1 день	Пн 12.02.24	Пн 12.02.24	7
9		Анализ требований завершен	0 дней	Пн 12.02.24	Пн 12.02.24	8
10		Выбор средств реализации проекта	22 дней	Ср 10.01.24	Чт 08.02.24	
11		Анализ возможных программных средств	4 дней	Ср 10.01.24	Пн 15.01.24	1
12		Выбор подходящего ПО для реализации	1 день	Чт 08.02.24	Чт 08.02.24	6;11
13		Выбор средств реализации завершен	0 дней	Чт 08.02.24	Чт 08.02.24	12
14		Проектирование приложения	14 дней	Вт 13.02.24	Пт 01.03.24	
15		Проектирование базы данных	7 дней	Вт 13.02.24	Ср 21.02.24	9;13
16		Проектирование архитектуры приложения	7 дней	Чт 22.02.24	Пт 01.03.24	15
17		Проектирование интерфейса приложения	5 дней	Вт 13.02.24	Пн 19.02.24	9;13
18		Проектирование приложения завершенно	0 дней	Пт 01.03.24	Пт 01.03.24	16;17
19		Реализация проекта	25 дней	Пн 04.03.24	Пт 05.04.24	
20		Реализация базы данных	10 дней	Пн 04.03.24	Пт 15.03.24	18
21		Реализация серверной части приложения	15 дней	Пн 18.03.24	Пт 05.04.24	16;20
22		Реализация клиентской части приложения	14 дней	Пн 04.03.24	Чт 21.03.24	18
23		Реализация проекта завершен	0 дней	Пт 05.04.24	Пт 05.04.24	21;22
24		Тестирование приложения	20 дней	Пн 08.04.24	Пт 03.05.24	
25		Тестирование серверной части приложения	10 дней	Пн 08.04.24	Пт 19.04.24	23
26		Тестирование клиентской части приложения	8 дней	Пн 08.04.24	Ср 17.04.24	23
27		Исправление ошибок	10 дней	Пн 22.04.24	Пт 03.05.24	25;26
28		Тестирование приложения завершенно	0 дней	Пт 03.05.24	Пт 03.05.24	27
29		Внедрение программного продукта	6 дней	Пн 06.05.24	Пн 13.05.24	
30		Подготовка документации	2 дней	Пн 06.05.24	Вт 07.05.24	28
31		Подпись документации государственными органами	3 дней	Ср 08.05.24	Пт 10.05.24	30
32		Внесение в систему бланка европротокола	1 день	Пн 13.05.24	Пн 13.05.24	31
33		Внедрение программного продукта завершенно	0 дней	Пн 13.05.24	Пн 13.05.24	32
34		Конец проекта	0 дней	Пн 13.05.24	Пн 13.05.24	33

Рисунок 17 – Задачи проекта в среде Microsoft Project

В среде Microsoft Project на основе введенных была построена диаграмма Ганта. На её основании можно понять, что критический путь проходит через задачи 1, 3, 4, 5, 6, 7, 8, 9, 15, 16, 18, 20, 21, 23, 25, 27, 28, 30, 31, 32, 33, 34. На рисунке 18 представлен граф проекта, на котором штриховыми линиями указан критический путь.

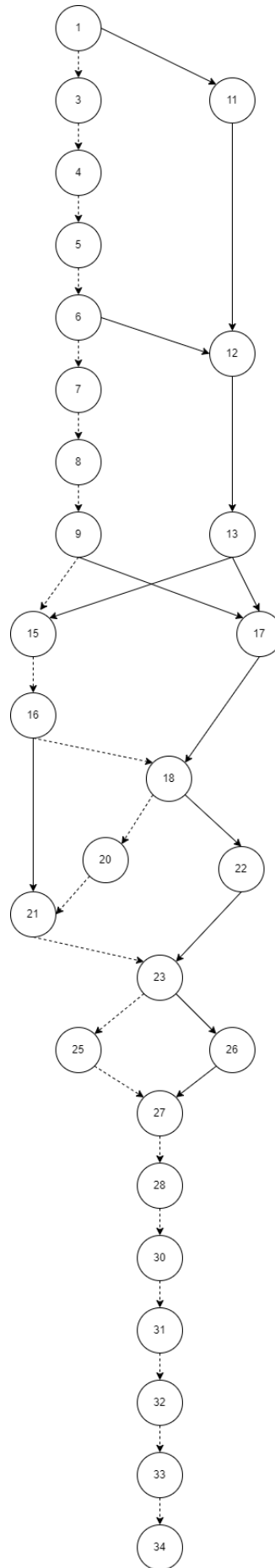


Рисунок 18 – Критический путь проекта

Диаграмма Ганта проекта приведена на рисунке 19.



Рисунок 19 – Диаграмма Ганта

На рисунках 20 – 22 представлены графики трудовых ресурсов «Программист 1», «Программист 2», «Аналитик».

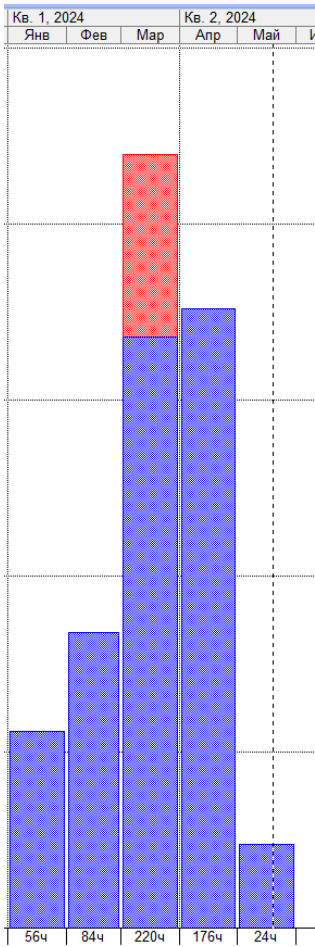


Рисунок 20 – График ресурсов для ресурса «Программист 1»

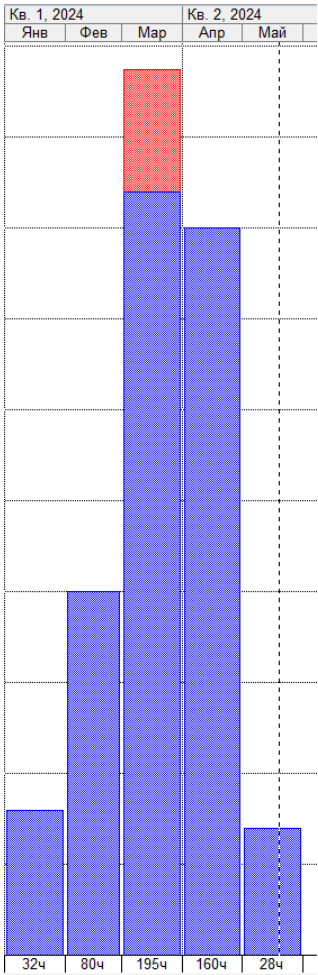


Рисунок 21 – График ресурсов для ресурса «Программист 2»

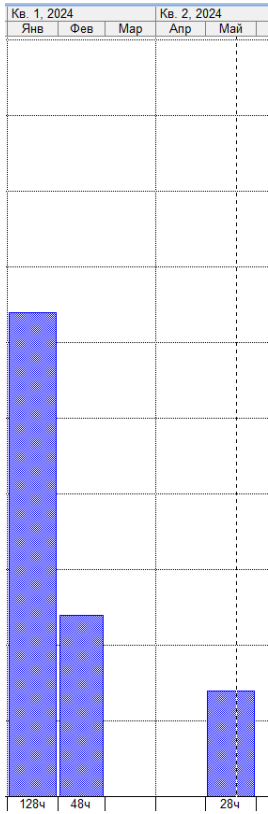


Рисунок 22 – График ресурсов для ресурса «Аналитик»

Как можно заметить, у ресурсов «Программист 1», «Программист 2» в марте наблюдается перегрузка.

На рисунке 23 представлен обзор проекта. Из рисунка видно, что общая длительность проекта 89 дней, а предполагаемые затраты – 782 100 рублей.

на дату Пн 20.05.24

Даты			
Начало:	Ср 10.01.24	Окончание:	Пн 13.05.24
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
Длительность			
Текущий план:	89 дней	Оставшаяся:	89 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	89 дней	Процент завершения:	0%
Трудозатраты			
Текущий план:	1 454 ч	Оставшиеся:	1 454 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	1 454 ч	Процент завершения:	0%
Затраты			
Текущий план:	782 100,00 Р	Оставшиеся:	782 100,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	782 100,00 Р		
Состояние задачи		Состояние ресурса	
Еще не начатые задачи	34	Трудовые ресурсы:	5
Выполняющиеся задачи	0	Трудовые ресурсы с прев. дос:	2
Завершенные задачи:	0	Материальные ресурсы:	4
Итого задач:	34	Итого ресурсов:	11

Рисунок 23 – Обзорные отчеты проекта

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

2 Проектирование мобильного приложения

2.1 Выбор технологий и инструментальных средств разработки

В процессе разработки приложения было решено использовать методологию RUP. Разработка приложения в соответствии с методологией RUP носит итеративный характер. На каждой итерации перед разработчиками ставятся определенные цели, которые должны быть достигнуты к ее концу. В результате в конце каждой итерации должна получиться промежуточная функциональная версия разрабатываемого ПО. В методологии RUP существуют 4 основных фазы, каждая из которых может включать в себя одну или несколько итераций [24]:

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

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

Выбор в пользу данной методологии продиктован рядом ее достоинств. С одной стороны, методология RUP обеспечивает гибкость процесса разработки и позволяет быстро вносить коррективы в проект в случае выявления в нем ошибок или изменения требований к разрабатываемому приложению. Кроме того, благодаря данной методологии удастся выявить основные риски уже на ранних стадиях разработки. Для анализа требований, а также моделирования структуры проекта в соответствии с принципами RUP было решено использовать диаграммы на языке моделирования

UML [25]. Для построения диаграмм использовалось средство UML-моделирования draw.io.

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

В качестве базы данных выбрана MS SQL Server 2019, является одной из самых востребованных СУБД на рынке. Серверная часть взаимодействует с БД с помощью ORM-технологии Entity Framework Core 6. Данная технология позволяет управлять сущностями БД через их проекцию – классы C# описанные с использованием специальных атрибутов, а также предоставляет возможность выполнения SQL запросов через язык запросов Linq.

Серверная часть приложения будет реализована при помощи платформы ASP.NET, данная платформа предоставляет обширный базовый функционал для обработки HTTP запросов и методы управления поведением сервера. Сервер будет работать под управлением Windows 10.

Для создания клиентского мобильного приложения было решено использовать мультиплатформенный фреймворк и среду программирования Microsoft Visual Studio 2022, предназначенную для разработки программ под разные операционные системы (в нашем случае Android), и язык программирования C#. Проектирование базы данных выполнялось с использованием приложения draw.io. Реализация базы данных производилась с использованием языка программирования SQL.

2.2 Проектирование архитектуры программного обеспечения

Было принято решение использовать в приложении двухуровневую архитектуру «клиент-сервер». Ее схема представлена на рисунке 24.

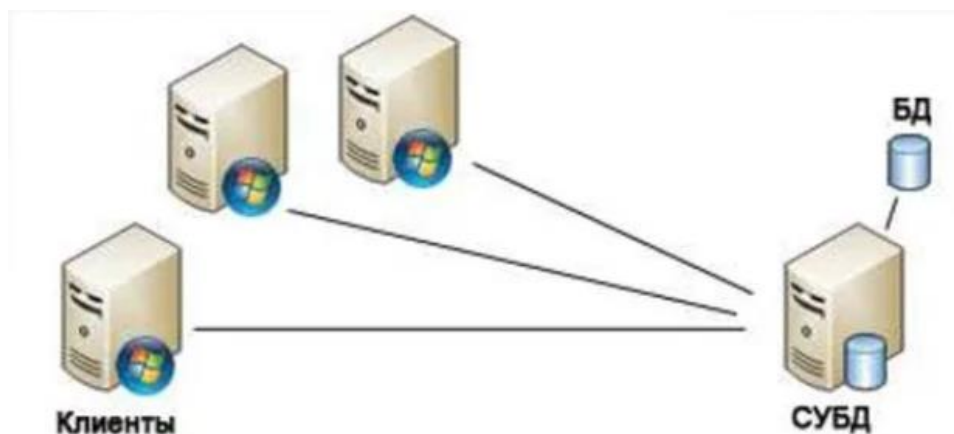


Рисунок 24 – Двухуровневая архитектура «клиент-сервер»

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

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

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

Преимуществом архитектуры является то, что все данные хранятся на сервере, что позволяет уменьшить размер клиентского приложения. Благодаря независимости серверной части от клиентской, можно создавать различные варианты клиентов для потенциального расширения приложения на различные платформы, помимо Android.

2.3 Проектирование интерфейса пользователя

На основе выявленных функциональных требований был разработан интерфейс мобильного приложения. Он включает в себя страницы входа, регистрации, заполнения электронного европротокола, просмотра итогового электронного европротокола, редактирования персональных данных, добавления/удаления правил ПДД, добавления/удаления бланка европротокола, получения списка электронных европротоколов, получения информации по конкретному европротоколу. Ниже будет приведено проектирование основных страниц приложения.

Страница входа представлена на рисунке 25. На ней присутствуют следующие элементы: 1 – поле для ввода логина пользователя, 2 – поле для ввода пароля пользователя, 3 – кнопка «Вход», 4 – кнопка для перехода на страницу регистрации. При нажатии на кнопку «Вход» система проверяет, существует ли на самом деле пользователь с таким логином и паролем. Если такого пользователя нет, будет выведено сообщение о неверном логине или пароле.

The diagram shows a vertical rectangle representing the login screen. Inside, there are four elements arranged vertically: 1. A rectangular input field for the username. 2. A rectangular input field for the password. 3. A rectangular button labeled 'Вход' (Login) with the number 3 to its right. 4. A rectangular button labeled 'Регистрация' (Registration) with the number 4 to its right. Each element has a small horizontal line pointing to it from the number label.

Рисунок 25 – Страница входа

Страница для регистрации автомобилиста представлена на рисунке 26. На ней имеются следующие элементы: 1 – поле для ввода логина, 2 – поле для ввода пароля пользователя, 3 – область, содержащая поля для ввода персональной информации о пользователе, 4 – кнопка «Зарегистрироваться».

При нажатии кнопки «Зарегистрироваться» система проверяет, являются ли введенные данные корректными. Если это условие не выполняется, будет выведено сообщение о недопустимых данных. В противном случае будет проверено, не занят ли данный логин. Если он занят, выводится соответствующее сообщение. Иначе авторизация будет считаться выполненной успешно.

Скриншот веб-формы регистрации. Форма имеет белый фон и черную рамку. В верхней части расположены два горизонтальных поля ввода: первое (логин) и второе (пароль). Ниже них находится блок с заголовком «Персональные данные», содержащий несколько пустых полей для ввода. В самом низу формы расположена кнопка «Зарегистрироваться». Числа 1, 2, 3 и 4 с линиями указывают на следующие элементы: 1 – поле для ввода логина, 2 – поле для ввода пароля, 3 – поле для ввода фамилии, 4 – кнопка «Зарегистрироваться».

Рисунок 26 – Страница регистрации автомобилиста

Страница регистрации администратора приведена на рисунке 27. На ней имеются следующие элементы: 1 – поле для ввода логина, 2 – поле для ввода пароля, 3 – поле для ввода фамилии пользователя, 4 – поле для ввода имени пользователя, 5 – поле для ввода отчества пользователя, 6 – поле для ввода телефона пользователя, 7 – кнопка «Зарегистрироваться». При нажатии кнопки «Зарегистрироваться» система проверяет, являются ли введенные логин и пароль допустимыми. Если это условие не выполняется, будет выведено сообщение о недопустимом логине или пароле. В противном случае будет проверено, не занят ли данный логин. Если он занят, выводится

соответствующее сообщение. Иначе авторизация будет считаться выполненной успешно.

The diagram illustrates the registration form for an administrator. It consists of a vertical stack of six rectangular input fields, each followed by a number from 1 to 6. Below these fields is a button labeled 'Зарегистрироваться' (Register), which is indicated by the number 7. The entire form is enclosed in a rectangular border.

1
2
3
4
5
6
7 Зарегистрироваться

Рисунок 27 – Страница регистрации администратора

Страница регистрации представителя страховой компании приведена на рисунке 28. На ней имеются следующие элементы: 1 – поле для ввода логина, 2 – поле для ввода пароля, 3 – поле для ввода фамилии пользователя, 4 – поле для ввода имени пользователя, 5 – поле для ввода отчества пользователя, 6 – поле для ввода телефона пользователя, 7 – поле для ввода названия страховой компании, 8 – кнопка «Зарегистрироваться». При нажатии кнопки «Регистрация» система проверяет, являются ли введенные логин и пароль допустимыми. Если это условие не выполняется, будет выведено сообщение о недопустимом логине или пароле. В противном случае будет проверено, не занят ли данный логин. Если он занят, выводится соответствующее сообщение. Иначе авторизация будет считаться выполненной успешно.

1

2

3

4

5

6

7

8

Зарегистрироваться

Рисунок 28 – Страница регистрации представителя страховой компании

Страница главного меню автомобилиста приведена на рисунке 29. На ней приведены следующие элементы: 1 – поле, которое отражает личные данные пользователя, 2 – кнопка для перехода к оформлению электронного европротокола, 3 – кнопка для перехода к странице изменения личной информации пользователя, 4 – кнопка для перехода к странице просмотра правил ПДД.

1

4 Правила ПДД

3 Мои данные

2 ДТП

Рисунок 29 – Страница главного меню автомобилиста

Страница оформления электронного европротокола приведена на рисунке 30. На ней имеются следующие элементы: 1 – область вывода информации по разделам, 2 – область, содержащая поля для ввода персональных данных участника ДТП, 3 – область, содержащая поля для ввода сведений о транспортном средстве участника ДТП, 4 – кнопка «Отправить», 5 – кнопка для возврата в главное меню. При нажатии кнопки «Отправить» система отправляет часть протокола на сервер для того, чтобы объединить её со второй частью другого участника.

Информация о ДТП	
1	
Персональные данные	
2	
Сведения о ТС	
3	
Отправить	4
Назад	5

Рисунок 30 – Страница заполнения электронного европротокола

Страница предварительной проверки электронного европротокола приведена на рисунке 31. На ней имеются следующие элементы: 1 – область, содержащая информацию о ДТП, 2 – область, содержащая информацию о персональных данных первого участника, 3 – область, содержащая сведения о транспортном средстве первого участника, 4 – область, содержащая информацию о персональных данных второго участника, 5 – область, содержащая сведения о транспортном средстве первого участника, 6 – кнопка «Подписать и отправить», 7 – кнопка для возвращения к заполнению электронного европротокола. При нажатии кнопки «Подписать и отправить» система переводит пользователя к окну, в котором он должен выполнить

электронную подпись документа, после чего европротокол будет сохранен для страховой компании.

Информация о ДТП	1
Первый участник	
Персональные данные	2
Сведения о ТС	
3	
Второй участник	
Персональные данные	4
Сведения о ТС	
5	
Подписать и отправить	6
Назад	7

Рисунок 31 – Страница предварительной проверки электронного европротокола

Электронная подпись – это атрибут электронного документа, который позволяет установить авторство и неизменность после подписания. В зависимости от своего вида электронная подпись может быть полностью равнозначна рукописной и обеспечивает подписанным файлам юридическую силу. Для данного приложения будет использоваться простая электронная подпись – знакомая многим пара логин-пароль в личных кабинетах. Причиной тому служит её легкодоступность для целевой аудитории приложения. Кроме простой также существуют неквалифицированная электронная подпись и квалифицированная электронная подпись.

2.4 Логическое проектирование базы данных

На основе анализа предметной области и функциональных требований была построена концептуальная модель базы данных. Она представлена в виде диаграммы «сущность-связь» (рисунок 32).

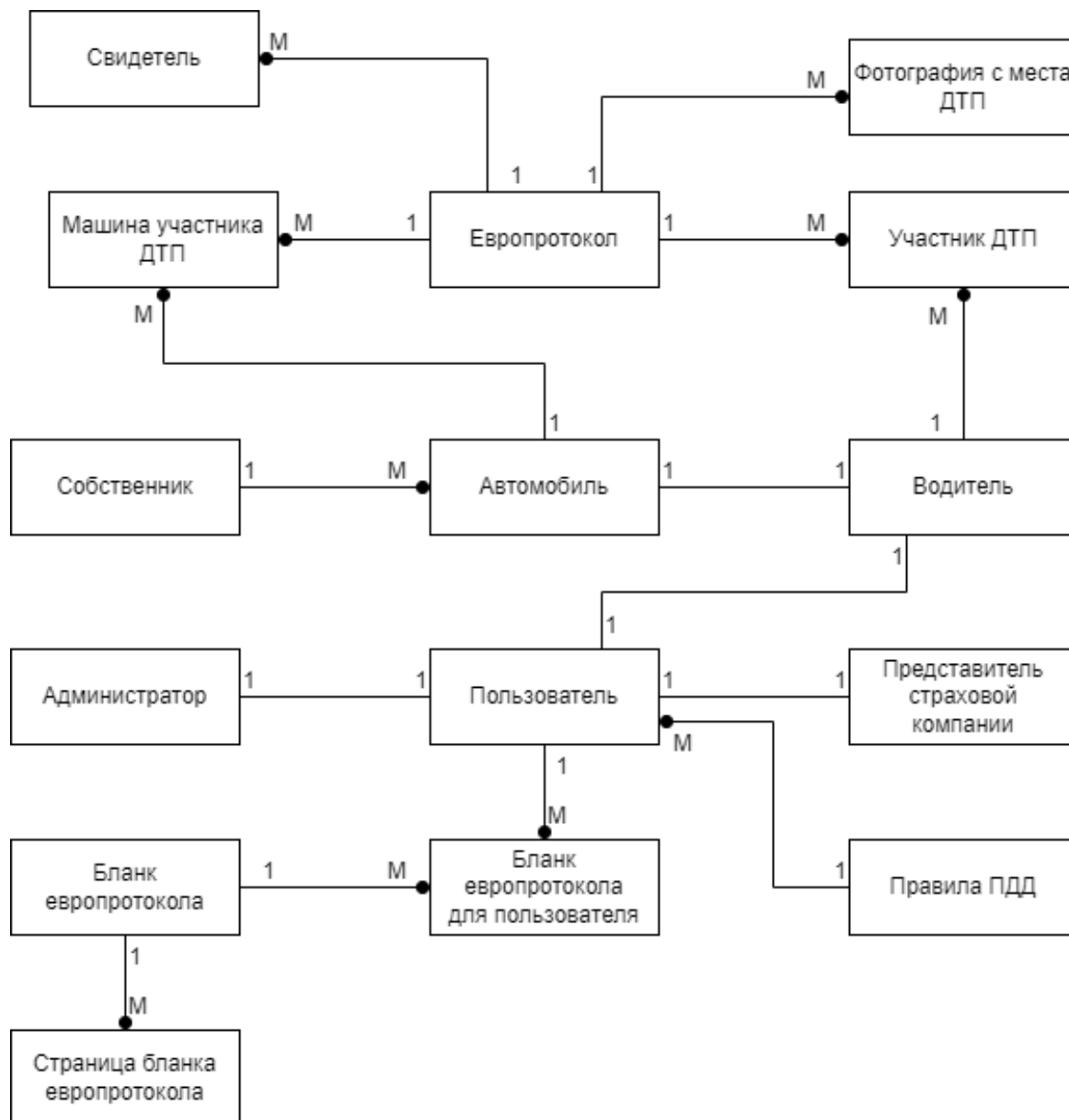


Рисунок 32 – Диаграмма «сущность-связь»

Сущности концептуальной модели данных описаны в таблице 5.

Таблица 5 – Сущности предметной области

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Свидетель	Список свидетелей по разным ДТП	Стержневая	100000	В систему добавляются новые свидетели при добавлении нового европротокола

Продолжение таблицы 5

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
2	Фотография с места ДТП	Список фотографий с места ДТП	Стержневая	100000	В систему добавляются новые фотографии при добавлении нового европротокола
3	Европротокол	Список оформленных европротоколов	Стержневая	1000000	Обновляется после оформлении нового ДТП
4	Участник ДТП	Список участников разных ДТП	Ассоциативная	1000000	Обновляется при добавлении нового протокола
5	Водитель	Список водителей	Стержневая	1000000	Новые водители добавляются при добавлении новых европротоколов, если таких водителей ещё нет в системе
6	Автомобиль	Список автомобилей	Стержневая	1000000	Обновляется при добавлении водителя
7	Собственник	Список собственников автомобилей	Стержневая	1000000	При добавлении автомобиля добавляется так же информация о его собственнике
8	Администратор	Список администраторов	Стержневая	1000	Администраторы добавляются в систему при регистрации пользователя с ролью «Администратор»
9	Пользователь	Список пользователей системы	Стержневая	10000	В систему пользователи добавляются при регистрации
10	Представитель страховой компании	Список лиц, представляющих страховые компании	Стержневая	1000	Представители добавляются в систему при регистрации пользователя с ролью «Страховая компания»
11	Бланк европротокола	Список бланков, с помощью которых можно оформить европротокол	Стержневая	1000	Бланки европротоколов в систему добавляет администратор
12	Бланк европротокола для пользователя	Список бланков, которые используют пользователи	Ассоциативная	1000	Такие бланки добавляются в систему при добавлении нового пользователя
13	Правила ПДД	Список файлов с правилами ПДД	Стержневая	1000	Новые правила ПДД добавляет администратор
14	Страница бланка европротокола	Список страниц для формирования бланка европротокола	Стержневая	10000	Новые страницы в систему добавляет администратор
15	Машина участника ДТП	Список машин, побывавших в конкретном ДТП	Ассоциативная	10000	Обновляется при добавлении нового европротокола

В таблице 6 представлено описание связей между сущностями предметной области.

Таблица 6 – Описание связей между сущностями предметной области

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
1	Европротокол	Свидетель	В одном европротоколе может быть один свидетель / Один свидетель может быть у нескольких европротоколов	1:M	Необязательная (у ДТП может отсутствовать свидетель)	Не может быть несколько свидетелей в одном европротоколе
2	Европротокол	Фотография с места ДТП	В одном европротоколе может быть несколько фотографий ДТП / Одна фотография может быть только в одном европротоколе	1:M	Обязательная	
3	Европротокол	Водитель	В одном европротоколе может быть несколько водителей / Водитель может быть в нескольких европротоколах	M:M	Обязательная	В одном европротоколе может быть только два водителя
4	Водитель	Автомобиль	У одного водителя может быть одна машина / У одного автомобиля может быть только один водитель	1:1	Обязательная	
5	Собственник	Автомобиль	Собственник может иметь несколько автомобилей / Автомобиль может иметь одного собственника	1:M	Обязательная	
6	Пользователь	Водитель	Пользователь может быть водителем / Водитель может иметь только одну учетную запись	1:1	Обязательная	
7	Пользователь	Администратор	Пользователь может быть администратором / Администратор может иметь только одну учетную запись	1:1	Обязательная	
8	Пользователь	Представитель страховой компании	Пользователь может быть представителем страховой компании / Представитель компании может иметь только одну учетную запись	1:1	Обязательная	
9	Правила ПДД	Пользователь	Правила ПДД могут иметь несколько пользователей / Пользователь может иметь только один файл с правилами ПДД	1:M	Обязательная	

Продолжение таблицы 6

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
10	Пользователь	Бланк европротокола	Пользователь может иметь несколько бланков европротокола / Бланк европротокола может иметь несколько пользователей	M:M	Обязательная	
11	Бланк европротокола	Страница бланка европротокола	Бланк европротокола может иметь несколько страниц бланка европротокола / Страница бланка европротокола может иметь только один бланк	1:M	Обязательная	
12	Европротокол	Автомобиль	В одном европротоколе может быть несколько автомобилей / Автомобиль может быть в нескольких европротоколах	M:M	Обязательная	В одном европротоколе может быть только два автомобиля

На основании концептуальной модели данных была построена логическая модель базы данных. Её структура описана диаграммой в нотации IDEF1X, представленной на рисунке 33.

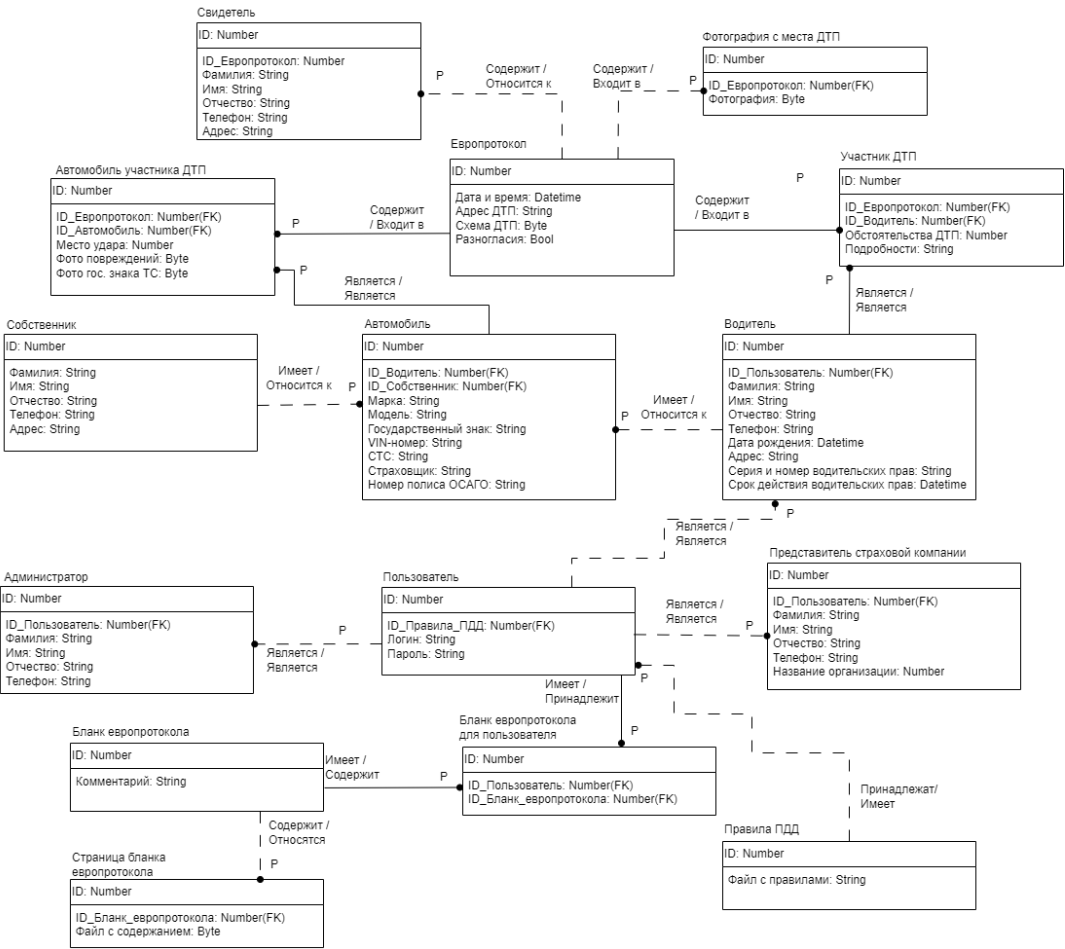


Рисунок 33 – Логическая модель базы данных

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

2.5 Проектирование серверной части приложения

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

В приложении используется архитектура MVC (Model-View-Controller). Ее схема представлена на рисунке 34.

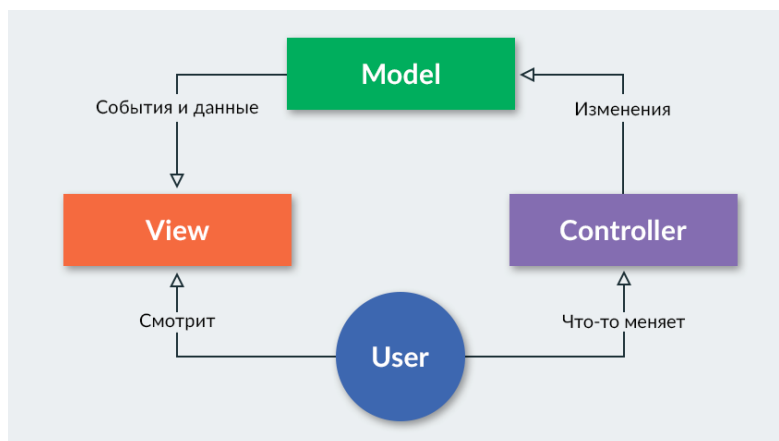


Рисунок 34 – Архитектура MVC

Серверная часть приложения включает в себя классы Модели, отвечающие за организацию доступа к данным (т. е. за непосредственное взаимодействие с БД) и классы Контроллера, отвечающие за организацию взаимодействия пользователя с клиентом. Именно они обрабатывают HTTP-запросы клиента и формируют ответы на них. К клиенту же отойдет только уровень Представления, отвечающий за взаимодействие с пользователем.

3 Реализация мобильного приложения

3.1 Реализация базы данных

На основе составленной ранее логической модели БД с учетом специфики СУБД Microsoft SQL Server при помощи веб-сервиса draw.io была построена физическая модель. Ее структура представлена на рисунке 35.

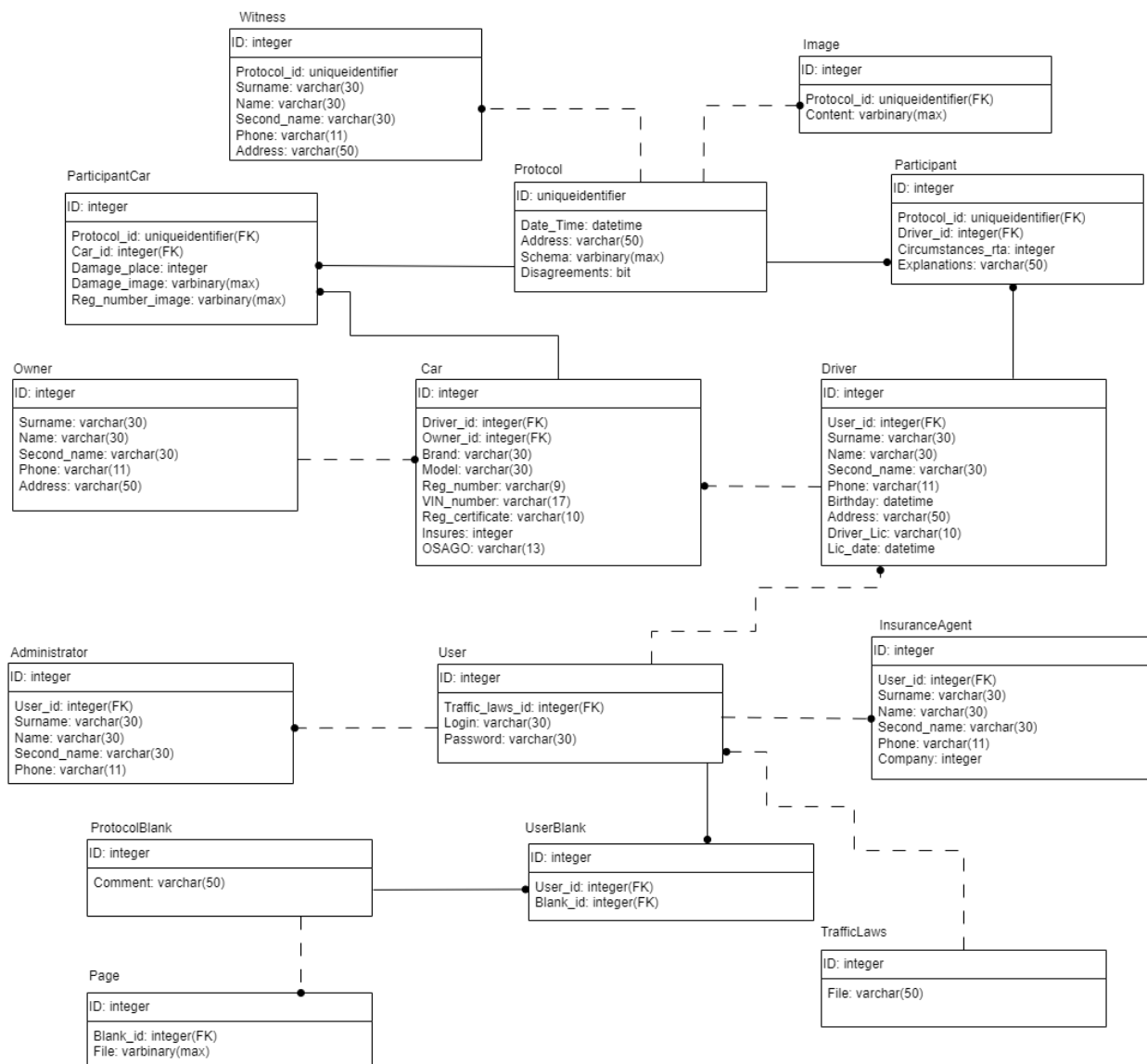


Рисунок 35 – Физическая модель базы данных

На основе составленной физической модели БД был составлен скрипт на языке SQL для создания базы данных. Его код представлен в приложении Б. Развертывание спроектированной базы данных было выполнено путем запуска данного скрипта с использованием утилиты

SQL Server Management Studio, используемой для администрирования баз данных под управлением СУБД Microsoft SQL Server.

3.2 Разработка серверной части

Разработка серверной части приложения производилась в среде программирования Microsoft Visual Studio 2022 на языке C# с применением технологии Web API [26 – 27]. Для взаимодействия с базой данных использовалась технология Entity Framework [28].

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

Диаграмма классов слоя модели представлена на рисунке 36. Все эти классы были автоматически сгенерированы средой программирования Visual Studio 2022. Генерация производилась на основе предварительно разработанной до этого базы данных (подход Database First) [29 – 30].

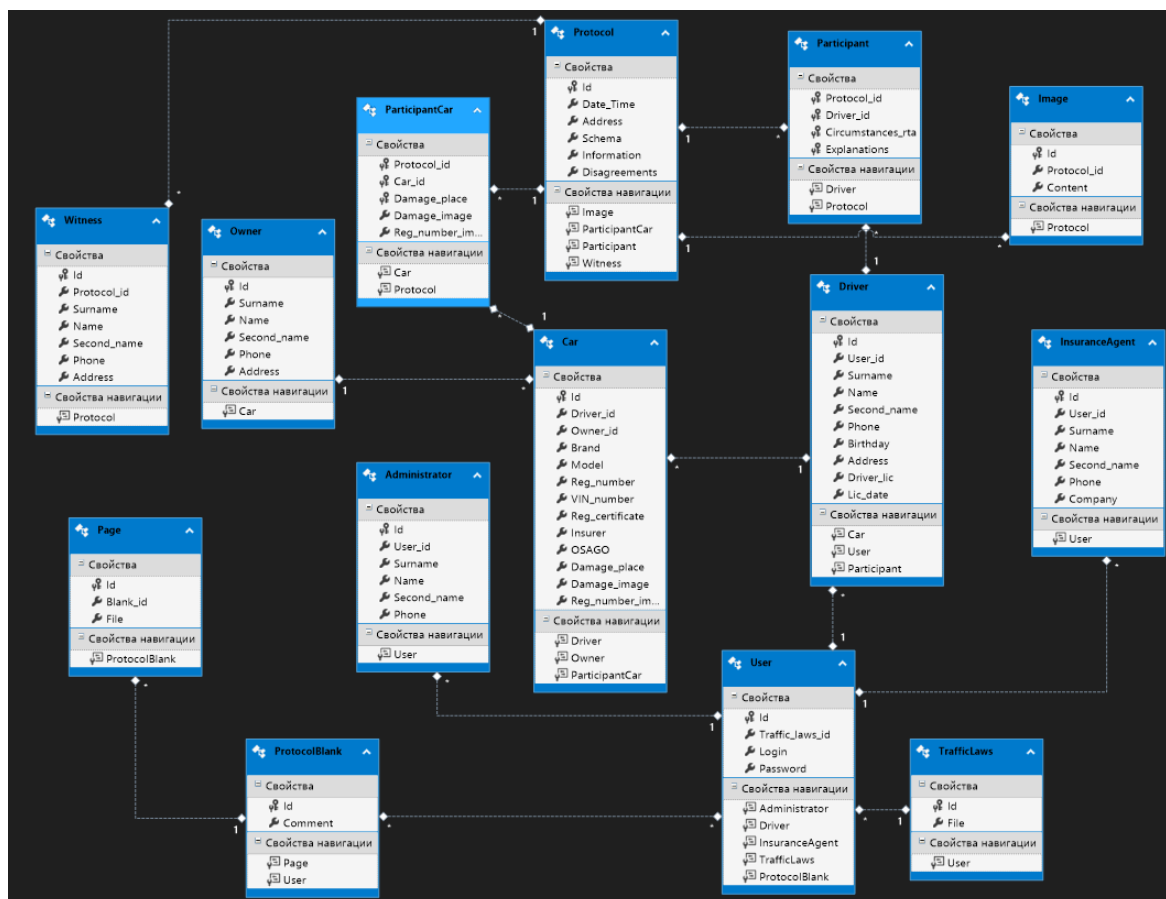


Рисунок 36 – Диаграмма классов серверной части (слой модели)

Как можно заметить, все классы были созданы на основании одноименных таблиц база данных, что свидетельствует о корректной генерации схемы.

Диаграмма классов слоя контроллеров представлена на рисунке 37.



Рисунок 37 – Диаграмма классов серверной части (слой контроллеров)

Более подробное описание классов слоя контроллеров и их назначений представлено в таблице 7.

Таблица 7 – Описание классов слоя контроллеров

Класс	Назначение
AdministratorsController	Отвечает за обработку запросов на добавление в систему новых администраторов и на получение информации по определённому из имеющихся
CarsController	Отвечает за обработку запросов на добавление новых машин, а так же на обновление и получения списка уже имеющихся машин
DriversController	Отвечает за обработку запросов на добавление в систему новых водителей, а так же на обновление и выборку уже имеющихся в системе

Продолжение таблицы 7

ImagesController	Отвечает за обработку запросов на добавление в систему новых фотографий с места ДТП, а так же на обновление и выборку уже имеющихся в системе
InsuranceAgentsController	Отвечает за обработку запросов на добавление в систему новых представителей страховых компаний и получение списка уже имеющихся
OwnersController	Отвечает за обработку зпросов на добавление в систему новых собственников, а так же выборку и обновление уже имеющихся в системе
PageController	Отвечает за обработку запросов на добавление в систему новых страниц и на удаление уже имеющихся, а также на выборку страниц в соответствии с некоторыми критериями, на получение полной информации о любой из имеющихся страниц
ParticipantCarsController	Отвечает за обработку запросов на получение машины участника в соответствии с некоторыми критериями, а так же добавление нового транспортного средства
ParticipantsController	Отвечает за обработку запросов на добавление новых участников ДТП и на получение конкретного из списка
ProtocolBlanksController	Отвечает за обработку запросов на добавление новых списков бланков, а так же выборку из уже имеющихся
WitnessController	Отвечает за обработку запросов на добавление новых свидетелей, а так же выборку из уже имеющихся
UsersController	Отвечает за обработку запросов на добавление новых аккаунтов и на получение списка имеющихся
UserBlanksController	Отвечает за обработку запросов на добавление новых пользовательских списков бланков и на получение информации об имеющихся
TrafficLawsController	Отвечает за обработку запросов на добавление новых правил ПДД, а так же удаление и получение уже имеющихся правил в системе
ProtocolsController	Отвечает за обработку запросов на добавление в систему новых европротоколов и на удаление уже имеющихся, а также на выборку европротоколов в соответствии с некоторыми критериями, на получение полной информации о любом из имеющихся европротоколов

Листинг серверной части представлен в приложении В.

3.3 Реализация клиентской части приложения

Разработка мобильного приложения производилась на современном языке C# с использованием высокотехнологичной среды программирования Visual Studio 2022. Для создания мобильного мультиплатформенного пользовательского интерфейса был использован фреймворк .NET MAUI, который позволяет создавать приложения для iOS, Android и Windows с общим кодом. Диаграмма компонентов, отражающая структуру клиентского приложения, представлена на рисунке 38.

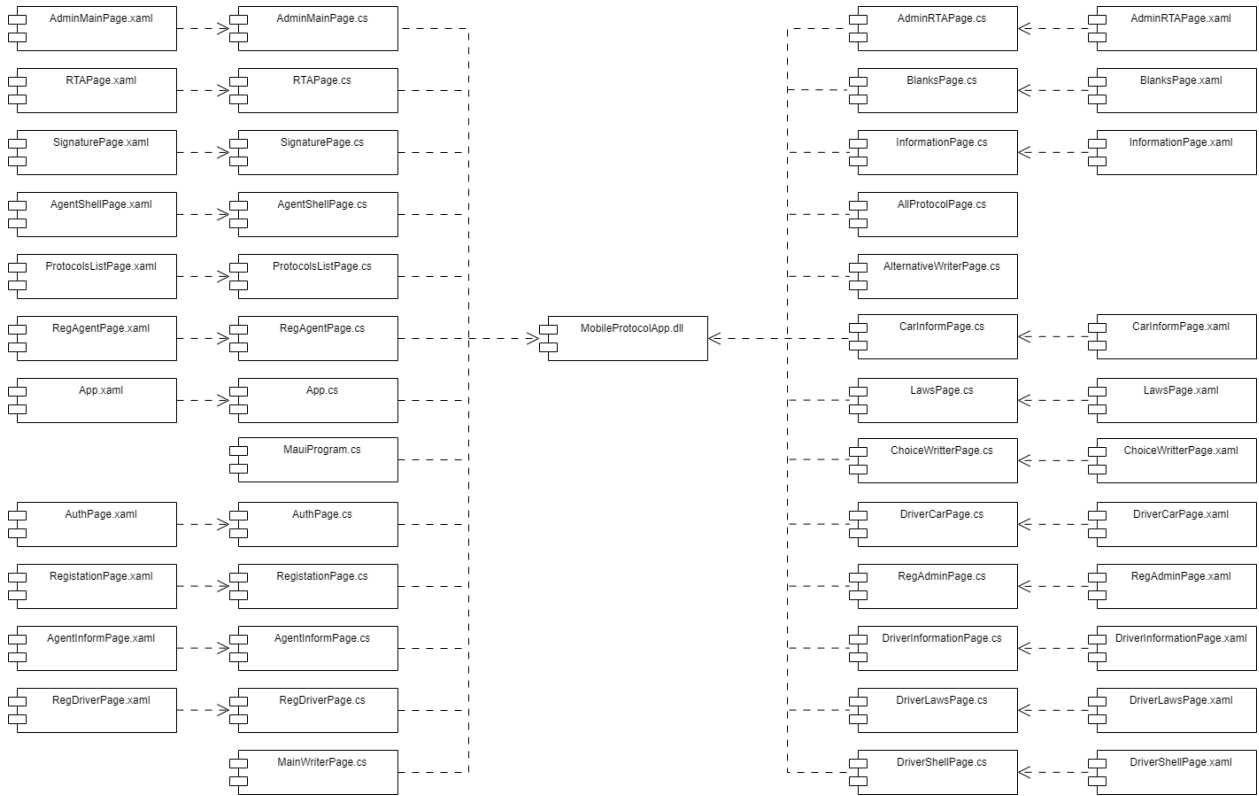


Рисунок 38 – Диаграмма компонентов мобильного приложения

Описание компонентов представлено в таблице 8.

Таблица 8 – Компоненты клиентского приложения

Компонент	Назначение
AdminMainPage.xaml	ХАМЛ-файл описания интерфейса главной страницы администратора
AdminMainPage.xaml.cs	Управляет логикой страницы администратора
AdminRTAPage.xaml	ХАМЛ-файл описания интерфейса страницы с правилами оформления европротокола для администратора
AdminRTAPage.xaml.cs	Управляет логикой страницы с правилами оформления европротокола для администратора
BlanksPage.xaml	ХАМЛ-файл описания интерфейса страницы добавления новых бланков
BlanksPage.xaml.cs	Управляет логикой страницы добавления новых бланков
InformationPage.xaml	ХАМЛ-файл описания интерфейса страницы персональной информации администратора
InformationPage.xaml.cs	Управляет логикой и данными страницы персональной информации администратора
LawsPage.xaml	ХАМЛ-файл описания интерфейса страницы добавления правил ПДД
LawsPage.xaml.cs	Управляет логикой и данными страницы добавления правил ПДД
RegAdminPage.xaml	ХАМЛ-файл описания интерфейса страницы регистрации администратора
RegAdminPage.xaml.cs	Управляет логикой и данными страницы регистрации администратора
AllProtocolPage.cs	Управляет данными и логикой подгружаемой страницы просмотра заполненного европротокола

Продолжение таблицы 8

Компонент	Назначение
AlternativeWriterPage.cs	Управляет логикой и данными подгружаемой страницы заполнения альтернативных данных ДТП
CarInformPage.xaml	XAML-файл описания интерфейса страницы просмотра информации о машине
CarInformPage.xaml.cs	Управляет логикой и данными страницы просмотра информации о машине
ChoiceWriterPage.xaml	XAML-файл описания интерфейса страницы выбора ролей в оформлении электронного европротокола
ChoiceWriterPage.xaml.cs	Управляет логикой и данными страницы выбора ролей в оформлении электронного европротокола
DriverCarPage.xaml	XAML-файл описания интерфейса страницы просмотра списка имеющихся автомобилей
DriverCarPage.xaml.cs	Управляет логикой и данными страницы просмотра списка имеющихся автомобилей
DriverInformationPage.xaml	XAML-файл описания интерфейса страницы просмотра личной информации водителя
DriverInformationPage.xaml.cs	Управляет логикой и данными страницы просмотра личной информации водителя
DriverLawsPage.xaml	XAML-файл описания интерфейса страницы просмотра правил ПДД
DriverLawsPage.xaml.cs	Управляет логикой и данными страницы просмотра правил ПДД
DriverShellPage.xaml	XAML-файл описания интерфейса главной страницы пользователя с ролью водитель
DriverShellPage.xaml.cs	Управляет логикой и данными главной страницы пользователя с ролью водитель
MainWriterPage.cs	Управляет логикой и данными подгружаемой страницы заполнения основных данных ДТП
RegDriverPage.xaml	XAML-файл описания интерфейса страницы регистрации водителя
RegDriverPage.xaml.cs	Управляет логикой и данными страницы регистрации водителя
RTAPage.xaml	XAML-файл описания интерфейса страницы с правилами оформления европротокола для водителя
RTAPage.xaml.cs	Управляет логикой и данными страницы с правилами оформления европротокола для водителя
SignaturePage.xaml	XAML-файл описания интерфейса страницы электронной подписи европротокола
SignaturePage.xaml.cs	Управляет логикой и данными страницы электронной подписи европротокола
AgentInformPage.xaml	XAML-файл описания интерфейса страницы просмотра личных данных представителя страховой компании
AgentInformPage.xaml.cs	Управляет логикой и данными страницы просмотра личных данных представителя страховой компании
AgentShellPage.xaml	XAML-файл описания интерфейса главной страницы представителя страховой компании
AgentShellPage.xaml.cs	Управляет логикой и данными главной страницы представителя страховой компании
ProtocolsListPage.xaml	XAML-файл описания интерфейса страницы просмотра списка оформленных электронных европротоколов
ProtocolsListPage.xaml.cs	Управляет логикой и данными страницы просмотра списка оформленных электронных европротоколов
RegAgentPage.xaml	XAML-файл описания интерфейса страницы регистрации представителя страховой компании
RegAgentPage.xaml.cs	Управляет логикой и данными страницы регистрации представителя страховой компании
App.xaml	XAML-файл описания подключаемых ресурсов к приложению

Окончание таблицы 8

Компонент	Назначение
App.xaml.cs	Является начальной точкой загрузки, закрытия приложения
AuthPage.xaml	ХАМЛ-файл описания интерфейса страницы входа в приложение
AuthPage.xaml.cs	Управляет логикой и данными страницы входа в приложение
MauiProgram.cs	Содержит класс MauiProgram, который определяет стартовый класс приложения (по умолчанию класс App) и ряд общих для приложения настроек
RegistrationPage.xaml	ХАМЛ-файл описания интерфейса страницы выбора роли регистрируемого пользователя
RegistrationPage.xaml.cs	Управляет логикой и данными страницы выбора роли регистрируемого пользователя

Диаграмма развертывания, на которой описывается физическое расположение компонентов системы, представлена на рисунке 39.

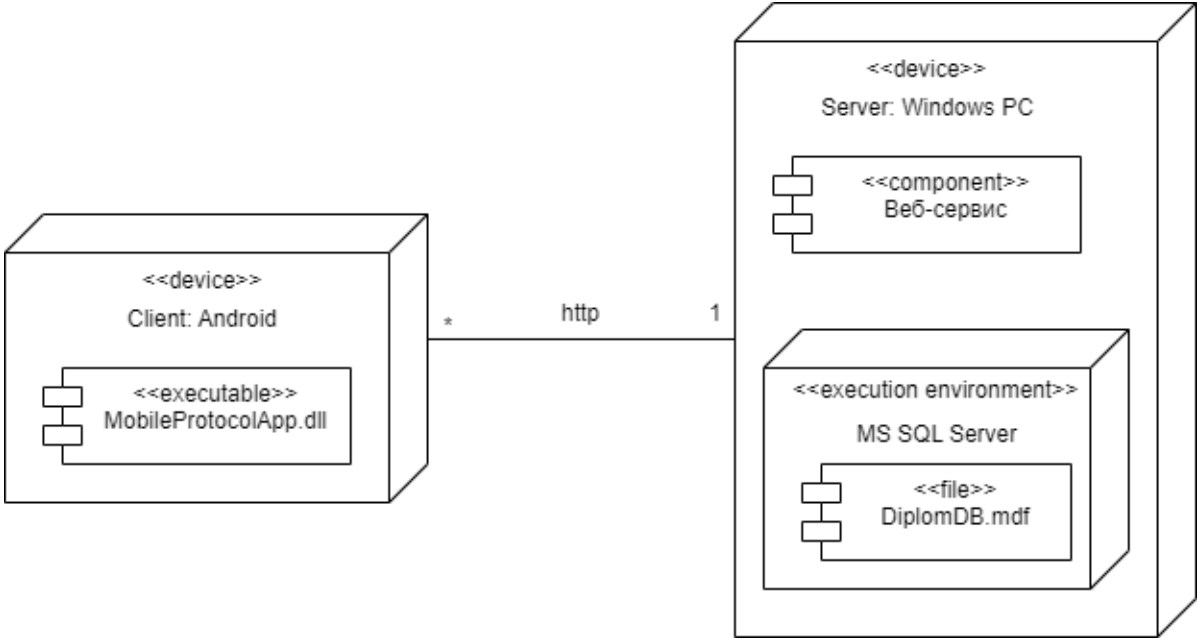


Рисунок 39 – Диаграмма развертывания мобильного приложения

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 проверить корректность работы серверной части приложения. С этой целью было проведено тестирование серверной части приложения с использованием инструментального средства Swagger. Результаты тестирования серверной части представлены в таблице 9. Скриншоты тестов находятся в приложении Д.

Таблица 9 – Результаты тестирования серверной части

№	Описание теста	Структура запроса	Результат
1	Получение списка всех администраторов	Метод: GET URL: api/Administrators	Тест пройден. Найдены все администраторы (рисунок Д.1).
2	Получение списка всех пользователей	Метод: GET URL: api/Users	Тест пройден. Найдены все пользователи (рисунок Д.2).
3	Получение конкретного администратора	Метод: GET URL: api/Administrators/1	Тест пройден. Администратор найден (рисунок Д.3).
4	Получение конкретного собственника авто	Метод: GET URL: api/Owners/6	Тест пройден. Собственник авто найден (рисунок Д.4).
5	Обновление данных собственника авто	Метод: PUT URL: api/Owners Body: {id = 14, name = Никита, ...}	Тест пройден. Собственник авто обновлен (рисунок Д.5).
6	Добавление нового собственника авто	Метод: POST URL: api/Owners	Тест пройден. Собственник авто добавлен (рисунок Д.6).
7	Получение списка водителей	Метод: GET URL: api/Drivers	Тест пройден. Получен список водителей (рисунок Д.7).
8	Получение одного водителя	Метод: GET URL: api/Drivers/3	Тест пройден. Получен конкретный водитель (рисунок Д.8).
9	Обновление данных о водителе	Метод: PUT URL: api/Drivers	Тест пройден. Обновлено информация о водителе (рисунок Д.9).
10	Получение списка машин	Метод: GET URL: api/Cars	Тест пройден. Получен список автомобилей (рисунок Д.10).
11	Получение списка машин, принадлежащих конкретному водителю	Метод: GET URL: api/Cars/3	Тест пройден. Получен список автомобилей конкретного водителя (рисунок Д.11).
12	Получение списка правил ПДД	Метод: GET URL: api/TrafficLaws	Тест пройден. Получен список правил ПДД (рисунок Д.12).
13	Добавление новых правил ПДД	Метод: POST URL: api/TrafficLaws	Тест пройден. Новые правила добавлены (рисунок Д.13).
14	Удаление правил ПДД	Метод: DELETE URL: api/TrafficLaws?id=3	Тест пройден. Правила успешно удалены (рисунок Д.14).
15	Добавление нового набора бланков	Метод: POST URL: api/ProtocolBlanks	Тест пройден. Новый набор успешно добавлен (рисунок Д.15).

Продолжение таблицы 9

№	Описание теста	Структура запроса	Результат
16	Авторизация пользователя	Метод: GET URL: api/Users/Victor/onetwo	Тест успешно пройден. Пользователь найден. Авторизация успешна (рисунок Д.16).
17	Получение списка европротоколов	Метод: GET URL: api/Protocols	Тест успешно пройден. Получен список европротоколов (рисунок Д.17).
18	Удаление европротокола	Метод: DELETE URL: api/Protocols/ec7a5f40-2de5-4308-97cc-73d2d8287ebe	Тест успешно пройден. Европротокол успешно удален (рисунок Д.18).
19	Получение конкретного представителя страховой компании	Метод: GET URL: api/InsuranceAgents/9	Тест успешно пройден. Получен конкретный представитель страховой компании (рисунок Д.19).
20	Получение списка всех фотографий с места ДТП	Метод: GET URL: api/Images	Тест успешно пройден. Получен список всех фотографий с места ДТП (рисунок Д.20).
21	Удаление бланка европротокола	Метод: DELETE URL: api/Pape/10	Тест успешно пройден. Бланк европротокола успешно удален (рисунок Д.21).
22	Получение конкретного участника ДТП	Метод: GET URL: api/Participants/c23a51d6-5a84-4886-af70-57a2094a9822	Тест успешно пройден. Получен конкретный участник ДТП (рисунок Д.22).

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

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 10. Скриншоты тестов находятся в приложении Е.

Таблица 10 – Функциональное тестирование

Вариант использования	Тест и параметры	Результат
Зарегистрироваться	Попробовать зарегистрироваться без введенной информации.	Тест пройден. Система вывела сообщение о незаполненных полях (рисунок Е.1).
	Попробовать зарегистрироваться с недопустимым логином.	Тест пройден. Система вывела сообщение о некорректном логине (рисунок Е.2).

Продолжение таблицы 10

Вариант использования	Тест и параметры	Результат
Зарегистрироваться	Выбрать регистрацию администратора. Ввести валидные данные: Логин: Igor@mail.com. Пароль: Igor45!Igor. Фамилия: Маркин. Имя: Игорь. Отчество: Витальевич. Телефон: 4346798765.	Тест пройден. Пользователь администратор зарегистрирован в системе (рисунок Е.3).
	Выбрать регистрацию водителя транспортного средства. Ввести валидные данные для водителя: Логин: elena@gmail.com. Пароль: Elena123**RED. Фамилия: Краснова. Имя: Елена. Отчество: Павловна. Телефон: 4578765356. Дата рождения: 1995-04-14. Адрес проживания: ул Тамбовская 33. Серия и водительских прав: 1234345675. Срок окончания окончания водительских прав: 2024-09-13.	Тест пройден. Водитель зарегистрирован в системе (рисунок Е.4).
	Выбрать регистрацию представителя страховой компании. Ввести валидные данные: Логин: anna1@gmail.com. Пароль: Anna65!ol. Фамилия: Фомина. Имя: Анна. Отчество: Михайловна. Телефон: 34567765578. Название компании: Ренессанс.	Тест пройден. Представитель страховой компании зарегистрирован в системе (рисунок Е.5).
	Попробовать зарегистрироваться с недопустимым паролем.	Тест пройден. Система вывела сообщение о некорректном пароле (рисунок Е.6).
Авторизоваться	Попытаться войти как администратор. Ввести валидные данные: Логин: Igor@mail.com Пароль: Igor45!Igor	Тест пройден. Авторизация прошла успешно (рисунок Е.7).

Продолжение таблицы 10

Вариант использования	Тест и параметры	Результат
Авторизоваться	Попытаться войти как водитель. Ввести валидные данные: Логин: Oleg@gmail.ru. Пароль: Oleg12x*	Тест пройден. Авторизация прошла успешно (рисунок Е.8)
	Попытаться войти как представитель страховой компании. Ввести валидные данные: Логин: anna1@gmail.com. Пароль: Anna65!ol	Тест успешно пройден. Авторизация прошла успешно (рисунок Е.9).
	Попытаться войти без введенных данных.	Тест успешно пройден. Система вывела сообщение о недостатке данных (рисунок Е.10).
	Попытаться войти в несуществующий личный кабинет.	Тест успешно пройден. Система вывела сообщение об отсутствии пользователя (рисунок Е.11).
Добавить новые правила ПДД	Ввести данные нового ресурса: Ссылка на портал: http://pddportal/	Тест успешно пройден. Правила ПДД добавлены в систему (рисунок Е.12).
Добавить набор для бланков европротокола	Ввести данные для набора: Комментарий: новый набор2	Тест успешно пройден. Новый набор добавлен в систему (рисунок Е.13).
Добавить бланк европротокола	Ввести корректные данные: Номер набора: новый набор2. Тип страницы: Альтернативный оформитель. Файл страницы: MainPage.txt.	Тест успешно пройден. Новая страница бланка добавлена в систему (рисунок Е.14).
Посмотреть правила ПДД	Перейти на вкладку «Правила ПДД»	Тест успешно пройден. Текущие правила ПДД отображаются на странице (рисунок Е.15).
Изменить личные данные	Изменить данные водителя. Подтвердить изменения.	Тест успешно пройден. Изменения применены (рисунок Е.16).
Посмотреть список своих транспортных средств	Перейти на вкладку «Мои ТС». Обновить данные.	Тест успешно пройден. Список транспортных средств отображается в системе (рисунок Е.17).
Заполнить электронный европротокол	Ввести валидные данные для всех полей. Попытаться перейти к просмотру всего европротокола.	Тест успешно пройден. Часть европротокола успешно сохранена в системе (рисунок Е.18).
Получить список европротоколов	Перейти на вкладку «Список европротоколов». Обновить данные.	Тест успешно пройден. Система отобразила список всех оформленных европротоколов (рисунок Е.19).

Окончание таблицы 10

Вариант использования	Тест и параметры	Результат
Посмотреть содержимое европротокола	Выбрать необходимый европротокол из списка.	Тест успешно пройден. Система отобразила информацию о выбранном европротоколе (рисунок E.20).
Выбрать роль в заполнении европротокола	Попытаться перейти к заполнению европротокола без выбора своей роли.	Тест успешно пройден. Система вывела сообщение об отсутствии выбранных данных (рисунок E.21).

Функциональное тестирование показало корректность работы клиентской части мобильного приложения.

4.2.2 Модульное тестирование

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

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

Для проведения модульных тестов был разработан тестовый класс UnitTest1, код которого представлен в листинге 1.

Листинг 1 – Исходный код модульных тестов

```
namespace TestProject1
{
    public class UnitTest1
    {
        [Fact]
        public void ValidLogin()
        {
            Assert.True(TestClass.IsLogin("Oleg@gmail.com"));
        }

        [Fact]
        public void InvalidLogin()
        {

```

```

        Assert.False(TestClass.IsLogin("Oleggmail.ru"));
    }

    [Fact]
    public void ShortLogin()
    {
        Assert.False(TestClass.IsLogin("T"));
    }

    [Fact]
    public void LongLogin()
    {
        Assert.False(TestClass.IsLogin("khsgdlghsgsdgiyyiihksdg12414fhhl@gmail.com"));
    }

    [Fact]
    public void NullLogin()
    {
        Assert.False(TestClass.IsLogin(null));
    }

    [Fact]
    public void ValidPassword()
    {
        Assert.True(TestClass.IsPassword("O1lleg12c*3"));
    }

    [Fact]
    public void ShortPassword()
    {
        Assert.False(TestClass.IsPassword("Oo1*"));
    }

    [Fact]
    public void LongPassword()
    {
        Assert.False(TestClass.IsPassword("OOFFFasfafaf12--=.sfsdlsdlfpsdn9"));
    }

    [Fact]
    public void NullPassword()
    {
        Assert.False(TestClass.IsPassword(null));
    }

    [Fact]
    public void NoSymbolsPassword()
    {
        Assert.False(TestClass.IsPassword("Ronan1212S"));
    }

    [Fact]
    public void NoLetterPassword()
    {
        Assert.False(TestClass.IsPassword("12314535***2"));
    }

    [Fact]
    public void OnlyNumberPassword()
    {
        Assert.False(TestClass.IsPassword("1234567865"));
    }

```

```
[Fact]
public void OnlySymbolsPassword()
{
    Assert.False(TestClass.IsPassword("*&*!@!!_+"));
}
}
```

Результаты модульных тестов приведены на рисунке 40.

Тестирование	Длительн...	Призн
TestProject1 (13)	4 мс	
TestProject1 (13)	4 мс	
UnitTest1 (13)	4 мс	
InvalidLogin	< 1 мс	
LongLogin	< 1 мс	
LongPassword	3 мс	
NoLetterPassword	< 1 мс	
NoSymbolsPassword	< 1 мс	
NullLogin	< 1 мс	
NullPassword	< 1 мс	
OnlyNumberPassword	< 1 мс	
OnlySymbolsPassword	< 1 мс	
ShortLogin	1 мс	
ShortPassword	< 1 мс	
ValidLogin	< 1 мс	
ValidPassword	< 1 мс	

Рисунок 40 – Результаты модульных тестов

Все тесты были пройдены успешно, что свидетельствует о корректности работы методов IsLogin и IsPassword.

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 GetPage класса BlanksPage, отвечающий за обработку запроса на выборку страниц для европротокола. Его код представлен на листинге 2.

Листинг 2 – Добавление новой страницы

```
private async void GetPage(object sender, EventArgs e)
```

```

{
    PageService? service = new();
    // Получаем страницы от сервера
    IEnumerable<MobileProtocolApp.Models.Page>? pages = await
    service.Get();
    // Если данных нет, то выходим
    if (pages == null)
        return;
    // Добавляем данные в элемент Picker
    foreach (var page in pages)
    {
        string line = page.Id.ToString();
        picker_page_number_.Items.Add(line);
    }
    await DisplayAlert("Message", "Список обновлен", "OK");
}

```

Данный метод находится в клиентской части приложения и запускается при попытке администратора обновить список страниц для просмотра имеющихся. При вызове он запрашивает данные с сервера. Если страниц в базе данных нет, то метод просто не обновляет список, и он остается пустым. В противном случае метод заполняет элемент Picker страницы администратора теми данными, которые были получены.

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 11.

Таблица 11 – Метрики размера программы

Метрика	Количество
Количество строк кода	14
Строки-комментарии	3

Также вычислим процент комментариев. Он равен $(3/14)*100=21$.

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 4 характеристики программы [31 – 33]:

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

С целью вычисления этих характеристик были составлен словарь операторов (таблица 12) и словарь операндов (таблица 13).

Таблица 12 – Словарь операторов

Оператор	Число операторов
async	1
await	2
{}	2
?	2
;	6
new	1
<>	1
=	3
==	1
.	6
if	1
return	1
var	1
in	1
foreach	1
n1 = 15	N1 = 30
service	2
MobileProtocolApp	1
Models	1
Page	1

Таблица 13 – Словарь операндов

Операнд	Число операндов
Get	1
pages	3
page	2
line	2
Id	1
ToString	1
null	1
picker_page_number_	1

Продолжение таблицы 13

Операнд	Число операндов
Items	1
Add	1
DisplayAlert	1
«Message»	1
«Список обновлен»	1
«OK»	1
$n2 = 18$	$N2 = 23$

Таким образом, число уникальных операторов программы $n1 = 15$, число уникальных операндов программы $n2 = 18$, общее число операторов в программе $N1 = 30$, общее число операндов в программе $N2 = 23$.

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

- словарь программы (n):

$$n = n1 + n2;$$

- длина программы(N):

$$N = N1 + N2;$$

- объем программы (V):

$$V = N * \log_2 n;$$

- сложность программы (D):

$$D = (n1/2) * (N2/n2);$$

- оценка интеллектуальных усилий (E):

$$E = D * V.$$

Рассчитаем словарь программы: $n = 15 + 18 = 33$.

Рассчитаем длину программы $N = 30 + 23 = 53$.

Рассчитаем объем программы: $V = 53 * \log_2 33 = 267,3$.

Рассчитаем сложность программы: $D = (15/2) * (23/18) = 9,6$.

Рассчитаем оценку интеллектуальных усилий: $E = 9,6 * 360 = 3450$.

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

Заключение

В ходе выполнения выпускной квалификационной работы было разработано мобильное приложение для поддержки процессов регистрации дорожно-транспортных происшествий.

На этапе анализа предметной области и формулирования требований был выполнен анализ программного обеспечения аналогичного назначения, в результате которого были выявлены их основные достоинства и недостатки, произведен анализ целевой аудитории и определены функциональные и нефункциональные требования, а также разработана концептуальная модель БД.

Было осуществлено планирование разработки и оценки бюджета. По результатам планирования было выявлено, что реализация проекта займет 89 дня, а предполагаемые затраты будут 773100 рублей, из чего можно сделать вывод о его экономической целесообразности.

На этапе проектирования была разработана логическая модель базы данных. Выявлено что база данных будет развернута под управлением MS SQL Server, серверная часть будет реализована на платформе ASP.NET Core, а мобильное приложение будет написано на языке C# на платформе .NET MAUI.

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

В дальнейшем планируется опубликовать разработанное приложение в магазине мобильных приложений для системы Android Google Play и развернуть его серверную часть на облачном сервисе, а также создать версию мобильного приложения для операционной системы iOS.

Таким образом, в ходе прохождения преддипломной практики были успешно выполнены все поставленные задачи.

Список использованных источников

1. Распоряжение Президента Российской Федерации от 9 мая 2017 г. № 203 «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы». [Электронный ресурс]. URL: http://zmedu.hostedu.ru/files/ykaz_7668.pdf (дата обращения: 12.05.2024).
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: 12.05.2024).
3. Указ о национальных целях развития Российской Федерации на период до 2030 года и на перспективу до 2036 года. [Электронный ресурс]. URL: <http://www.kremlin.ru/events/president/news/73986> (дата обращения 12.05.2024).
4. Постановление Правительства Российской Федерации от 15.04.2014 г. № 313. [Электронный ресурс]. URL: <http://government.ru/docs/all/91296/> (дата обращения 12.05.2024).
5. Продукт «Госуслуги.Авто» // Tadviser [Электронный ресурс]. URL: <https://www.tadviser.ru/index.php/Продукт:Госуслуги.Авто> (дата обращения 12.05.2024).
6. Приложение "Европротокол ДТП" – как официально оформить ДТП по телефону? // АвтоТонкости [Электронный ресурс]. URL: <https://autotonkosti.ru/q/prilozhenie-evroprotokol-dtp-kak-oficialno-oformit-dtp-ro-telefonu> (дата обращения 12.05.2024).
7. Мобильное приложение ДТП.Европротоколо. // Выберу [Электронный ресурс]. URL: https://www.vbr.ru/strahovanie/help/osago/mp-dtp-europrotokol-osago/?utm_referrer=https%3A%2F%2Fyandex.ru%2F (дата обращения 12.05.2024).
8. Появилось приложение для европротокола при ДТП. Что с ним не так? // Тинькофф журнал [Электронный ресурс]. URL:

<https://journal.tinkoff.ru/news/europrotokol-dtp-mp/> (дата обращения 12.05.2024).

9. О мобильном приложении «Помощник ОСАГО» // Электронный полис ОСАГО [Электронный ресурс]. URL: https://rsapolis.ru/pomoshhnik-osago/?utm_referrer=https%3A%2F%2Fyandex.ru%2F (дата обращения 12.05.2024).

10. Россияне загрузили в «Госуслуги Авто» почти 2,5 млн водительских удостоверений // www1.ru [Электронный ресурс]. URL: <https://www1.ru/news/2023/10/03/rossiiane-zagruzili-v-gosuslugi-avto-pochti-25-mln-voditelskix-udostoverenii.html> (дата обращения 12.05.2024)

11. «Госуслуги Авто» за два года набрали миллионы пользователей // [Telecom Daily](http://TelecomDaily.ru) [Электронный ресурс]. URL: <https://telecomdaily.ru/news/2023/11/16/gosuslugi-avto-za-dva-goda-nabrali-milliony-polzovateley> (дата обращения 12.05.2024).

12. Эксперты назвали самые популярные смартфоны и самых успешных производителей // Новые известия [Электронный ресурс]. URL: <https://newizv.ru/news/2023-12-07/eksperty-nazvali-samye-populyarnye-smartfony-i-samyh-uspeshnyh-proizvoditeley-424786> (дата обращения 13.05.2024).

13. Почему iPhone на руках у россиян стало больше, хотя продажи падают // [Forbes](http://Forbes.ru) [Электронный ресурс]. URL: <https://www.forbes.ru/tekhnologii/506311-pocemu-iphone-na-rukah-u-rossian-stalo-bol-se-hota-prodazi-padaut> (дата обращения 13.05.2024)

14. Опубликована статистика популярности версий Android: ситуация исправляется // Trashbox.ru [Электронный ресурс]. URL: <https://trashbox.ru/link/2023-10-31-statistika-populyarnosti-versij-android> (дата обращения 13.05.2024)

15. Ехлаков, Ю.П. Управление программными проектами. Стандарты, модели: Учебное пособие / Ю.П. Ехлаков. – СПб.: Лань, 2019. – 244 с.

16. Иванов, П.В. Управление проектами: Учебное пособие / П.В. Иванов. – Рн/Д: Феникс, 2018. – 102 с.
17. Гонтарева, И.В. Управление проектами / И.В. Гонтарева, Р.М. Нижегородцев, Д.А. Новиков. – М.: КД Либроком, 2018. – 384 с.
18. Ньютон, Р. Управление проектами от А до Я / Р. Ньютон. – М.: Альпина Паблишер, 2018. – 180 с.
19. Вольфсон, Б.Л. Гибкое управление проектами и продуктами / Б.Л. Вольфсон. – СПб.: Питер, 2019. – 208 с.
20. Скороход, С.В. Управление проектами средствами Microsoft Project. Курс лекций. / С.В. Скороход. – М.: Интуит НОУ, 2016. – 319 стр.
21. Куперштейн, В.И. Microsoft Project 2013 в управлении проектами / В.И. Куперштейн. – СПб.:БХВ-Петербург, 2013. – 432 с.
22. Джонсон, Т. Microsoft Project 2016. Шаг за шагом / Т.Джонсон, К.Четфилд. – М., Изд-во «КТК Галактика», 2018. – 566 с.
23. Акатова, Н.А. Планирование и контроль выполнения проектов с использованием MS Project / Н.А. Акатова, М.Г. Коротких. – М.: МИСиС, 2019. – 74 с.
24. Унифицированный процесс Rational (RUP) // Интернет-технологии.ру [Электронный ресурс]. – URL: <https://www.internet-technologies.ru/articles/newbie/unificirovannyy-process-rational-rup.html> (дата обращения: 16.05.2024).
25. Арлоу, Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2-е издание / Д. Арлоу, А. Нейштадт – СПб: Символ Плюс, 2007. – 624 с.
26. Введение в Web API // Metanit.com [Электронный ресурс]. URL: <https://metanit.com/sharp/mvc/12.1.php> (дата обращения: 23.05.2024).
27. Андерсон Р. Создание веб-API с помощью ASP.NET Core и Visula Studio для Windows // Microsoft [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.1&tabs=visual-studio> (дата обращения: 23.05.2024).

28. Руководство по Entity Framework // Metanit.com [Электронный ресурс]. URL: <https://metanit.com/sharp/entityframework/2.4.php> (дата обращения: 23.05.2024).
29. Database First // Microsoft [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/ef/ef6/modeling/designer/workflows/database-first> (дата обращения: 23.05.2024).
30. Entity Framework 6. Использование Database First // Professor Web [Электронный ресурс]. URL: https://professorweb.ru/my/entity-framework/6/level1/1_6.php (дата обращения: 23.05.2024).
31. Программный код и его метрики // Habr.com [Электронный ресурс]. URL: <https://habr.com/ru/company/intel/blog/106082/> (дата обращения: 23.05.2021).
32. Аверьянов, А.В. Применение метрик Холстеда для количественного оценивания характеристик программ ЭВМ / А.В. Аверьянов, И.Н. Кошель, В.В. Кузнецов // Известия высших учебных заведений. Приборостроение. – 2019. – №11. – С. 970 – 975.
33. Полякова, Э.Р. Исследование программного обеспечения встраиваемой вычислительной системы метриками Холстеда / Э.Р. Полякова, А.А. Поляков, А.Л.Шустов, Д.Г. Придачкин // Cloud of Science. – 2020. – №4. – С. 950 – 959.

Приложение А
(справочное)
Глоссарий

Финансовые факторы – факторы на основе которых можно сделать выводы о эффективности, стабильности экономической деятельности.

Бюджет – финансовый план определенного субъекта установленный на определенный период времени.

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

Android – мобильная операционная система используемая в подавляющем большинстве мобильных устройств.

СУБД (Система управления базой данных) – специализированное программное обеспечение, предоставляющее программный интерфейс, который дает возможность создавать, модифицировать и использовать базу данных.

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

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

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

REST API – архитектурный стиль взаимодействия компонентов распределенного приложения сети.

ООП – объектно ориентированное программирование, методология программирования, основанная на представлении программы в виде совокупности объектов.

Приложение Б
(рекомендуемое)
Исходный код скриптов базы данных

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

```

create table Administrator
(
    Id int primary key identity,
    [User_id] int not null,
    Surname varchar(30) check(Surname like replicate('a-яА-Я',
len(Surname))) not null,
    [Name] varchar(30) check([Name] like replicate('a-яА-Я', len([Name])))
not null,
    Second_name varchar(30) check(Second_Name like replicate('a-яА-Я',
len(Second_Name))) not null,
    Phone varchar(11) check(Phone like replicate('[0-9]', len(Phone))) not
null,
    foreign key ([User_id]) references [User] (Id)
);

create table InsuranceAgent
(
    Id int primary key identity,
    [User_id] int not null,
    Surname varchar(30) check(Surname like replicate('a-яА-Я',
len(Surname))) not null,
    [Name] varchar(30) check([Name] like replicate('a-яА-Я', len([Name])))
not null,
    Second_name varchar(30) check(Second_Name like replicate('a-яА-Я',
len(Second_Name))) not null,
    Phone varchar(11) check(Phone like replicate('[0-9]', len(Phone))) not
null,
    Company int not null,
    foreign key ([User_id]) references [User] (Id)
);

create table ProtocolBlank
(
    Id int primary key identity,
    Comment varchar(50) not null
);

create table UserBlank
(
    Id int primary key identity,
    [User_id] int not null,
    Blank_id int not null,
    foreign key ([User_id]) references [User] (Id),
    foreign key (Blank_id) references ProtocolBlank (Id)
);

create table ParticipantCar
(
    Id int primary key identity,
    Protocol_id uniqueidentifier not null,
    Car_id int not null,
    Damage_place int check(Damage_place >= 1 and Damage_place <= 10) not
null,
    Damage_image varbinary(MAX),
    Reg_number_image varbinary(MAX),
    foreign key (Protocol_id) references Protocol (Id) ON DELETE CASCADE,
    foreign key (Car_id) references Car (Id)
);

create table Car
(
    Id int primary key identity,

```

```

        Driver_id int not null,
        Owner_id int not null,
        Brand varchar(30) check(Brand LIKE REPLICATE('[ a-zA-Z]', len(Brand)))
not null,
        Model varchar(30) check(Model LIKE REPLICATE('[ a-zA-Z0-9]',
len(Model))) not null,
        Reg_number varchar(9) check(Reg_number LIKE REPLICATE('[a-я{1}0-9{3}a-
я{2}0-9{3}]', len(Reg_number))) not null,
        VIN_number varchar(17) check(VIN_number like replicate('[A-Z0-9]',
len(VIN_number))) not null,
        Reg_certificate varchar(10) check(len(Reg_certificate) = 10) not null,
        Insurer int not null,
        OSAGO varchar(13) check(OSAGO like replicate('[0-9]', len(OSAGO))) not
null,
        foreign key (Driver_id) references Driver (Id),
        foreign key (Owner_id) references [Owner] (Id)
);

create table Driver
(
        Id int primary key identity,
        [User_id] int not null,
        Surname varchar(30) check(Surname like replicate('[a-яA-Я]',
len(Surname))) not null,
        [Name] varchar(30) check([Name] like replicate('[a-яA-Я]', len([Name])))
not null,
        Second_name varchar(30) check(Second_Name like replicate('[a-яA-Я]',
len(Second_Name))) not null,
        Phone varchar(11) check(Phone like replicate('[0-9]', len(Phone))) not
null,
        Birthday datetime not null,
        [Address] varchar(50) not null,
        Driver_lic varchar(10) check(Driver_Lic LIKE REPLICATE('[0-9]',
len(Driver_Lic))),
        Lic_date datetime not null,
        foreign key ([User_id]) references [User] (Id)
);

create table TrafficLaws
(
        Id int primary key identity,
        [File] varchar(50) not null
);

create table [Owner]
(
        Id int primary key identity,
        Surname varchar(30) check(Surname like replicate('[a-яA-Я]',
len(Surname))) not null,
        [Name] varchar(30) check([Name] like replicate('[a-яA-Я]', len([Name])))
not null,
        Second_name varchar(30) check(Second_Name like replicate('[a-яA-Я]',
len(Second_Name))) not null,
        Phone varchar(11) check(Phone like replicate('[0-9]', len(Phone))) not
null,
        [Address] varchar(50) not null
);

create table [Page]
(
        Id int primary key identity,
        Blank_id int not null,
        [File] varbinary(max) not null,
        Page_name int not null,
        foreign key (Blank_id) references ProtocolBlank (Id)
);

```

```

create table Participant
(
    Id int primary key identity,
    Protocol_id uniqueidentifier not null,
    Driver_id int not null,
    Circumstances_rta int check(Circumstances_rta >= 1 and Circumstances_rta
<= 23) not null,
    Explanations varchar(50) not null,
    foreign key (Protocol_id) references Protocol (Id) ON DELETE CASCADE,
    foreign key (Driver_id) references Driver (Id)
);

create table [Image]
(
    Id int primary key identity,
    Protocol_id uniqueidentifier not null,
    Content varbinary(MAX) not null,
    foreign key (Protocol_id) references Protocol (Id) ON DELETE CASCADE
);

create table Protocol
(
    Id uniqueidentifier not null primary key,
    Date_Time datetime not null,
    [Address] varchar(50) not null,
    [Schema] varbinary(MAX) not null,
    Disagreements bit not null
);

create table [User]
(
    Id int primary key identity,
    Traffic_laws_id int not null,
    [Login] varchar(30) not null,
    [Password] varchar(30) not null,
    foreign key (Traffic_laws_id) references Traffic_laws (Id)
);

create table [Witness]
(
    Id int primary key identity,
    Protocol_id uniqueidentifier not null,
    Surname varchar(30) check(Surname like replicate('[a-zA-Я]',
len(Surname))) not null,
    [Name] varchar(30) check([Name] like replicate('[a-zA-Я]', len([Name])))
not null,
    Second_name varchar(30) check(Second_name like replicate('[a-zA-Я]',
len(Second_name))) not null,
    Phone varchar(11) check(Phone like replicate('[0-9]', len(Phone))) not
null,
    [Address] varchar(50) not null,
    foreign key (Protocol_id) references Protocol (Id) ON DELETE CASCADE
);

```

Приложение В
(рекомендуемое)
Исходный код серверной части

Листинг В.1 – Файл Administrators.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace EuroProtocolApp.Models
{
    public class Administrator
    {
        [Key]
        public int Id { get; set; }
        public int User_id { get; set; }
        [ForeignKey(nameof(User_id))]
        public User? User { get; set; }
        public string? Surname { get; set; }
        public string? Name { get; set; }
        public string? Second_name { get; set; }
        public string? Phone { get; set; }

        public override bool Equals(object? obj)
        {
            var other = obj as Administrator;
            return User_id == other.User_id && Surname == other.Surname
                && Name == other.Name && Second_name == other.Second_name;
        }
    }
}
```

Листинг В.2 – Файл Car.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace EuroProtocolApp.Models
{
    public class Car
    {
        [Key]
        public int Id { get; set; }
        public int Driver_id { get; set; }
        [ForeignKey(nameof(Driver_id))]
        public Driver? Driver { get; set; }
        public int Owner_id { get; set; }
        [ForeignKey(nameof(Owner_id))]
        public Owner? Owner { get; set; }
        public string? Brand { get; set; }
        public string? Model { get; set; }
        public string? Reg_number { get; set; }
        public string? VIN_number { get; set; }
        public string? Reg_certificate { get; set; }
        public int Insurer { get; set; }
        public string? OSAGO { get; set; }

        public override bool Equals(object? obj)
        {
            var other = obj as Car;
            return Driver_id == other.Driver_id && Owner_id == other.Owner_id
                && Brand == other.Brand && Model == other.Model
                && Reg_number == other.Reg_number;
        }
    }
}
```

Листинг В.3 – Файл Driver.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace EuroProtocolApp.Models
{
    public class Driver
    {
        [Key]
        public int Id { get; set; }
        public int User_id { get; set; }
        [ForeignKey("User_id")]
        public User? User { get; set; }
        public string? Surname { get; set; }
        public string? Name { get; set; }
        public string? Second_name { get; set; }
        public string? Phone { get; set; }
        public DateTime Birthday { get; set; }
        public string? Address { get; set; }
        public string? Driver_lic { get; set; }
        public DateTime Lic_date { get; set; }

        public override bool Equals(object? obj)
        {
            var other = obj as Driver;
            return Surname == other.Surname && Name == other.Name
                && Second_name == other.Second_name && Driver_lic ==
other.Driver_lic
                && Birthday == other.Birthday;
        }
    }
}
```

Листинг В.4 – Файл Helper.cs

```
using System;

namespace EuroProtocolApp.Models
{
    public class Helper
    {
        public static bool IsExists<T>(ref T value, List<T> list_values)
        {
            foreach (var item in list_values)
            {
                if (item.Equals(value))
                {
                    value = item;
                    return true;
                }
            }
            return false;
        }
    }
}
```

Листинг В.5 – Файл Image.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
```

```

namespace EuroProtocolApp.Models
{
    public class Image
    {
        [Key]
        public int Id { get; set; }
        public Guid Protocol_id { get; set; }
        [ForeignKey("Protocol_id")]
        public Protocol? Protocol { get; set; }
        public byte[]? Content { get; set; }

        public override bool Equals(object? obj)
        {
            var other = obj as Image;
            return Content == other.Content && Protocol_id ==
other.Protocol_id;
        }
    }
}

```

Листинг В.6 – Файл InsuranceAgent.cs

```

using Microsoft.EntityFrameworkCore.Metadata.Internal;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace EuroProtocolApp.Models
{
    public enum InsuranceCompany
    {
        kAlfa,
        kIngosstah,
        kRosgosstrah,
        kSogaz,
        kRenessans
    }
    public class InsuranceAgent
    {
        [Key]
        public int Id { get; set; }
        public int User_id { get; set; }
        [ForeignKey(nameof(User_id))]
        public User? User { get; set; }
        public string? Surname { get; set; }
        public string? Name { get; set; }
        public string? Second_name { get; set; }
        public string? Phone { get; set; }
        public InsuranceCompany Company { get; set; }

        public override bool Equals(object? obj)
        {
            var other = obj as InsuranceAgent;
            return Surname == other.Surname && Name == other.Name
                && Second_name == other.Second_name && Company ==
other.Company;
        }
    }
}

```

Листинг В.7 – Файл Owner.cs

```

using System.ComponentModel.DataAnnotations;

```

```

namespace EuroProtocolApp.Models
{
    public class Owner
    {
        [Key]
        public int Id { get; set; }
        public string? Surname { get; set; }
        public string? Name { get; set; }
        public string? Second_name { get; set; }
        public string? Phone { get; set; }
        public string? Address { get; set; }

        public override bool Equals(object? obj)
        {
            var other = obj as Owner;
            return Surname == other.Surname && Name == other.Name
                && Second_name == other.Second_name;
        }
    }
}

```

Листинг В.8 – Файл Page.cs

```

using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace EuroProtocolApp.Models
{
    public enum PageName
    {
        kMainWritter,
        kAltWritter,
        kFullProtocol
    }
    public class Page
    {
        [Key]
        public int Id { get; set; }
        public int Blank_id { get; set; }
        [ForeignKey(nameof(Blank_id))]
        public ProtocolBlank? Blank { get; set; }
        public byte[]? File { get; set; }

        public PageName Page_name { get; set; }

        public override bool Equals(object? obj)
        {
            var other = obj as Page;
            return Blank_id == other.Blank_id && File == other.File;
        }
    }
}

```

Листинг В.9 – Файл Participant.cs

```

using Microsoft.EntityFrameworkCore;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace EuroProtocolApp.Models
{
    public class Participant

```

```

{
    [Key]
    public int Id { get; set; }
    public Guid Protocol_id { get; set; }
    [ForeignKey("Protocol_id")]
    public Protocol? Protocol { get; set; }
    public int Driver_id { get; set; }
    [ForeignKey("Driver_id")]
    public Driver? Driver { get; set; }
    public int Circumstances_rta { get; set; }
    public string? Explanations { get; set; }

    public override bool Equals(object? obj)
    {
        var other = obj as Participant;
        return Protocol_id == other.Protocol_id
            && Driver_id == other.Driver_id;
    }
}

```

Листинг В.10 – Файл ParticipantCar.cs

```

using Microsoft.EntityFrameworkCore;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace EuroProtocolApp.Models
{
    public class ParticipantCar
    {
        [Key]
        public int Id { get; set; }
        public Guid Protocol_id { get; set; }
        [ForeignKey("Protocol_id")]
        public Protocol? Protocol { get; set; }

        public int Car_id { get; set; }
        [ForeignKey("Car_id")]
        public Car? Car { get; set; }

        public int Damage_place { get; set; }

        public byte[]? Damage_image { get; set; }

        public byte[]? Reg_number_image { get; set; }

        public override bool Equals(object? obj)
        {
            var other = obj as ParticipantCar;
            return Protocol_id == other.Protocol_id && Car_id ==
other.Car_id;
        }
    }
}

```

Листинг В.11 – Файл Protocol.cs

```

using System.ComponentModel.DataAnnotations;

namespace EuroProtocolApp.Models
{

```

```

public class Protocol
{
    [Key]
    public Guid Id { get; set; }
    public DateTime Date_Time { get; set; }
    public string? Address { get; set; }
    public byte[]? Schema { get; set; }
    public bool Disagreements { get; set; }

    public override bool Equals(object? obj)
    {
        var other = obj as Protocol;
        return Date_Time == other.Date_Time;
    }
}

```

Листинг В.12 – Файл ProtocolBlank.cs

```

using System.ComponentModel.DataAnnotations;

namespace EuroProtocolApp.Models
{
    public class ProtocolBlank
    {
        [Key]
        public int Id { get; set; }
        public string? Comment { get; set; }

        public override bool Equals(object? obj)
        {
            var other = obj as ProtocolBlank;
            return Comment == other.Comment;
        }
    }
}

```

Листинг В.13 – Файл ProtocolContext.cs

```

using Microsoft.EntityFrameworkCore;

namespace EuroProtocolApp.Models
{
    public class ProtocolContext : DbContext
    {
        public DbSet<Administrator> Administrator { get; set; }
        public DbSet<Car> Car { get; set; }
        public DbSet<Driver> Driver { get; set; }
        public DbSet<Image> Image { get; set; }
        public DbSet<InsuranceAgent> InsuranceAgent { get; set; }
        public DbSet<Owner> Owner { get; set; }
        public DbSet<Page> Page { get; set; }
        public DbSet<Participant> Participant { get; set; }
        public DbSet<ParticipantCar> ParticipantCar { get; set; }
        public DbSet<Protocol> Protocol { get; set; }
        public DbSet<ProtocolBlank> ProtocolBlank { get; set; }
        public DbSet<TrafficLaws> TrafficLaws { get; set; }
        public DbSet<User> User { get; set; }
        public DbSet<UserBlank> UserBlank { get; set; }
        public DbSet<Witness> Witness { get; set; }
        public ProtocolContext(DbContextOptions<ProtocolContext> options)
            : base(options)
        {
        }
    }
}

```

```

        {
            Database.EnsureCreated();
        }
    }
}

```

Листинг В.14 – Файл TrafficLaws.cs

```

using System.ComponentModel.DataAnnotations;

namespace EuroProtocolApp.Models
{
    public class TrafficLaws
    {
        [Key]
        public int Id { get; set; }
        [Required]
        public string? File { get; set; }

        public override bool Equals(object? obj)
        {
            var other = obj as TrafficLaws;
            return File == other.File;
        }
    }
}

```

Листинг В.15 – Файл User.cs

```

using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace EuroProtocolApp.Models
{
    public class User
    {
        [Key]
        public int Id { get; set; }
        public int Traffic_laws_id { get; set; }
        [ForeignKey("Traffic_laws_id")]
        public TrafficLaws? Laws { get; set; }
        public string? Login { get; set; }
        public string? Password { get; set; }

        public override bool Equals(object? obj)
        {
            var other = obj as User;
            return Login == other.Login && Password == other.Password;
        }
    }
}

```

Листинг В.16 – Файл UserBlank.cs

```

using Microsoft.EntityFrameworkCore;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace EuroProtocolApp.Models

```

```

{
    public class UserBlank
    {
        [Key]
        public int Id { get; set; }
        public int User_id { get; set; }
        [ForeignKey(nameof(User_id))]
        public User? User { get; set; }
        public int Blank_id { get; set; }
        [ForeignKey(nameof(Blank_id))]
        public ProtocolBlank? Blank { get; set; }
    }
}

```

Листинг В.17 – Файл UserType.cs

```

using Microsoft.EntityFrameworkCore;

namespace EuroProtocolApp.Models
{
    public enum Types
    {
        kAdmin,
        kDriver,
        kAgent,
        kNone
    };

    [Keyless]
    public class UserType
    {
        public Types Type { get; set; }
        public int User_id { get; set; }
    }
}

```

Листинг В.18 – Файл Witness.cs

```

using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace EuroProtocolApp.Models
{
    public class Witness
    {
        [Key]
        public int Id { get; set; }
        public Guid Protocol_id { get; set; }

        [ForeignKey("Protocol_id")]
        public Protocol? Protocol { get; set; }
        public string? Surname { get; set; }
        public string? Name { get; set; }
        public string? Second_name { get; set; }
        public string? Phone { get; set; }
        public string? Address { get; set; }

        public override bool Equals(object? obj)
        {
            var other = obj as Witness;
            return Protocol_id == other.Protocol_id;
        }
    }
}

```

```

    }
}

```

Листинг В.19 – Файл AdministratorsController

```

using EuroProtocolApp.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EuroProtocolApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AdministratorsController : ControllerBase
    {
        ProtocolContext db_;
        public AdministratorsController(ProtocolContext db)
        {
            db_ = db;
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<Administrator>>> Get()
        {
            return await db_.Administrator.ToListAsync();
        }

        [HttpGet("{id}")]
        public async Task<ActionResult<Administrator>> Get(int id)
        {
            var admin = await db_.Administrator.FirstOrDefault(x =>
x.User_id == id);
            if (admin == null)
                return NotFound();
            return new ObjectResult(admin);
        }

        [HttpPost]
        public async Task<ActionResult<Administrator>> Create(Administrator
admin)
        {
            if (admin == null)
                return BadRequest();

            if (Helper.IsExists(ref admin, db_.Administrator.ToList()))
                return Ok(admin);

            db_.Administrator.Add(admin);
            await db_.SaveChangesAsync();
            return Ok(admin);
        }
    }
}

```

Листинг В.20 – Файл CarsController.cs

```

using EuroProtocolApp.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EuroProtocolApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]

```

```

public class CarsController : ControllerBase
{
    ProtocolContext db_;
    public CarsController(ProtocolContext db)
    {
        db_ = db;
    }

    [HttpGet]
    public async Task<ActionResult<IEnumerable<Car>>> Get()
    {
        return await db_.Car.ToListAsync();
    }

    [HttpGet("{id}")]
    public async Task<ActionResult<IEnumerable<Car>>> Get(int id)
    {
        var car_list = await db_.Car.ToListAsync();
        var cars = car_list.FindAll(x => x.Driver_id == id);
        if (cars.Count == 0)
            return NotFound();

        return new ObjectResult(cars);
    }

    [HttpPost]
    public async Task<ActionResult<Car>> Create(Car car)
    {
        if (car == null)
            return BadRequest();

        if (Helper.IsExists(ref car, db_.Car.ToList()))
            return Ok(car);

        db_.Car.Add(car);
        await db_.SaveChangesAsync();
        return Ok(car);
    }

    [HttpPut]
    public async Task<ActionResult<Car>> Update(Car car)
    {
        if (car == null)
            return BadRequest();

        if (!db_.Car.Any(x => x.Id == car.Id))
            return NotFound();

        db_.Update(car);
        await db_.SaveChangesAsync();
        return Ok(car);
    }
}

```

Листинг В.21 – Файл DriversController.cs

```

using EuroProtocolApp.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EuroProtocolApp.Controllers
{
    [Route("api/[controller]")]

```

```

[ApiController]
public class DriversController : ControllerBase
{
    ProtocolContext db_;

    public DriversController(ProtocolContext db)
    {
        db_ = db;
    }

    [HttpGet]
    public async Task<ActionResult<IEnumerable<Driver>>> Get()
    {
        return await db_.Driver.ToListAsync();
    }

    [HttpGet("{id}")]
    public async Task<ActionResult<Driver>> Get(int id)
    {
        var driver = await db_.Driver.FirstOrDefaultAsync(x => x.User_id
== id);

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

        return new ObjectResult(driver);
    }

    [HttpPost]
    public async Task<ActionResult<Driver>> Create(Driver driver)
    {
        if (driver == null)
            return BadRequest();

        if (Helper.IsExists(ref driver, db_.Driver.ToList()))
            return Ok(driver);

        db_.Driver.Add(driver);
        await db_.SaveChangesAsync();
        return Ok(driver);
    }

    [HttpPut]
    public async Task<ActionResult<Driver>> Update(Driver driver)
    {
        if (driver == null)
            return BadRequest();

        if (!db_.Driver.Any(x => x.Id == driver.Id))
            return NotFound();

        db_.Driver.Update(driver);
        await db_.SaveChangesAsync();
        return Ok(driver);
    }
}

```

Листинг В.22 – Файл ImagesController.cs

```

using EuroProtocolApp.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

```

```

namespace EuroProtocolApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ImagesController : ControllerBase
    {
        ProtocolContext db_;
        public ImagesController(ProtocolContext db)
        {
            db_ = db;
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<Image>>> Get()
        {
            return await db_.Image.ToListAsync();
        }

        [HttpGet("{id}")]
        public async Task<ActionResult<Image>> Get(Guid id)
        {
            var list = await db_.Image.ToListAsync();
            var images = list.FindAll(x => x.Protocol_id == id);

            if (images.Count == 0)
                return NotFound();

            return new ObjectResult(images);
        }

        [HttpPost]
        public async Task<ActionResult<Image>> Create(Image img)
        {
            if (img == null)
                return BadRequest();

            if (Helper.IsExists(ref img, db_.Image.ToList()))
                return Ok(img);

            db_.Image.Add(img);
            await db_.SaveChangesAsync();
            return Ok(img);
        }

        [HttpPut]
        public async Task<ActionResult<Image>> Update(Image img)
        {
            if (img == null)
                return BadRequest();

            if (!db_.Image.Any(x => x.Id == img.Id))
                return NotFound();

            db_.Image.Update(img);
            await db_.SaveChangesAsync();
            return Ok(img);
        }
    }
}

```

Листинг В.23 – Файл InsuranceAgentsController.cs

```

using EuroProtocolApp.Models;
using Microsoft.AspNetCore.Mvc;

```

```

using Microsoft.EntityFrameworkCore;

namespace EuroProtocolApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class InsuranceAgentsController : ControllerBase
    {
        ProtocolContext db_;
        public InsuranceAgentsController(ProtocolContext db)
        {
            db_ = db;
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<InsuranceAgent>>> Get()
        {
            return await db_.InsuranceAgent.ToListAsync();
        }

        [HttpGet("{id}")]
        public async Task<ActionResult<InsuranceAgent>> Get(int id)
        {
            var agent = await db_.InsuranceAgent.FirstOrDefault(x =>
x.User_id == id);

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

            return Ok(agent);
        }

        [HttpPost]
        public async Task<ActionResult<InsuranceAgent>> Create(InsuranceAgent
agent)
        {
            if (agent == null)
                return BadRequest();

            if (Helper.IsExists(ref agent, db_.InsuranceAgent.ToList()))
                return Ok(agent);

            db_.InsuranceAgent.Add(agent);
            await db_.SaveChangesAsync();
            return Ok(agent);
        }
    }
}

```

Листинг В.24 – Файл OwnersController.cs

```

using EuroProtocolApp.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EuroProtocolApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class OwnersController : ControllerBase
    {
        ProtocolContext db_;

        public OwnersController(ProtocolContext db)

```

```

    {
        db_ = db;
    }

    [HttpGet]
    public async Task<ActionResult<IEnumerable<Owner>>> Get()
    {
        return await db_.Owner.ToListAsync();
    }

    [HttpGet("{id}")]
    public async Task<ActionResult<Owner>> Get(int id)
    {
        var owner = await db_.Owner.FirstOrDefaultAsync(x => x.Id == id);

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

        return Ok(owner);
    }

    [HttpPost]
    public async Task<ActionResult<Owner>> Create(Owner owner)
    {
        if (owner == null)
            return BadRequest();

        if (Helper.IsExists(ref owner, db_.Owner.ToList()))
            return Ok(owner);

        db_.Owner.Add(owner);
        await db_.SaveChangesAsync();
        return Ok(owner);
    }

    [HttpPut]
    public async Task<ActionResult<Owner>> Update(Owner owner)
    {
        if (owner == null)
            return BadRequest();

        if (!db_.Owner.Any(x => x.Id == owner.Id))
            return NotFound();

        db_.Owner.Update(owner);
        await db_.SaveChangesAsync();
        return Ok(owner);
    }
}

```

Листинг В.25 – Файл PageController.cs

```

using EuroProtocolApp.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EuroProtocolApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class PageController : ControllerBase
    {
        ProtocolContext db_;
    }
}

```

```

public PageController(ProtocolContext db)
{
    db_ = db;
}

[HttpGet]
public async Task<ActionResult<IEnumerable<Page>>> Get()
{
    return await db_.Page.ToListAsync();
}

[HttpGet("{id}")]
public async Task<ActionResult<IEnumerable<Page>>> Get(int id)
{
    var page_list = await db_.Page.ToListAsync();
    var pages = page_list.FindAll(x => x.Blank_id == id);

    if (pages.Count == 0)
        return NotFound();

    return new ObjectResult(pages);
}

[HttpPost]
public async Task<ActionResult<Page>> Create(Page page)
{
    if (page == null)
        return BadRequest();

    if (Helper.IsExists(ref page, db_.Page.ToList()))
        return Ok(page);

    db_.Page.Add(page);
    await db_.SaveChangesAsync();
    return Ok(page);
}

[HttpPut]
public async Task<ActionResult<Page>> Update(Page page)
{
    if (page == null)
        return BadRequest();

    if (!db_.Page.Any(x => x.Id == page.Id))
        return NotFound();

    db_.Page.Update(page);
    await db_.SaveChangesAsync();
    return Ok(page);
}

[HttpDelete("{id}")]
public async Task<ActionResult<Page>> Delete(int id)
{
    var page = db_.Page.FirstOrDefault(x => x.Id == id);

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

    db_.Page.Remove(page);
    await db_.SaveChangesAsync();
    return Ok(page);
}
}
}

```

Листинг В.26 – Файл ParticipantCarsController.cs

```

using EuroProtocolApp.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EuroProtocolApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ParticipantCarsController : ControllerBase
    {
        ProtocolContext db_;

        public ParticipantCarsController(ProtocolContext db)
        {
            db_ = db;
        }

        [HttpGet("{id}")]
        public async Task<ActionResult<IEnumerable<ParticipantCar>>> Get(Guid
id)
        {
            var list = await db_.ParticipantCar.ToListAsync();
            var cars = list.FindAll(x => x.Protocol_id == id);

            if (cars.Count == 0)
                return NotFound();

            return new ObjectResult(cars);
        }

        [HttpPost]
        public async Task<ActionResult<ParticipantCar>> Create(ParticipantCar
car)
        {
            if (car == null)
                return BadRequest();

            if (Helper.IsExists(ref car, db_.ParticipantCar.ToList()))
                return Ok(car);

            db_.ParticipantCar.Add(car);
            await db_.SaveChangesAsync();
            return Ok(car);
        }
    }
}

```

Листинг В.27 – Файл ParticipantsController.cs

```

using EuroProtocolApp.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EuroProtocolApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ParticipantsController : ControllerBase
    {
        ProtocolContext db_;

        public ParticipantsController(ProtocolContext db)

```

```

    {
        db_ = db;
    }

    [HttpGet("{id}")]
    public async Task<ActionResult<IEnumerable<Participant>>> Get(Guid
id)
    {
        var list = await db_.Participant.ToListAsync();
        var participants = list.FindAll(x => x.Protocol_id == id);

        if (participants.Count == 0)
            return NotFound();

        return new ObjectResult(participants);
    }

    [HttpPost]
    public async Task<ActionResult<Participant>> Create(Participant
participant)
    {
        if (participant == null)
            return BadRequest();

        if (Helper.IsExists(ref participant, db_.Participant.ToList()))
            return Ok(participant);

        db_.Participant.Add(participant);
        await db_.SaveChangesAsync();
        return Ok(participant);
    }
}

```

Листинг В.28 – Файл ProtocolBlanksController.cs

```

using EuroProtocolApp.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EuroProtocolApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ProtocolBlanksController : ControllerBase
    {
        ProtocolContext db_;

        public ProtocolBlanksController(ProtocolContext db)
        {
            db_ = db;
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<ProtocolBlank>>> Get()
        {
            return await db_.ProtocolBlank.ToListAsync();
        }

        [HttpGet("{id}")]
        public async Task<ActionResult<ProtocolBlank>> Get(int id)
        {
            var blank = await db_.ProtocolBlank.FirstOrDefault(x => x.Id
== id);

```

```

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

        return new ObjectResult(blank);
    }

    [HttpPost]
    public async Task<ActionResult<ProtocolBlank>> Create(ProtocolBlank
blank)
    {
        if (blank == null)
            return BadRequest();

        if (Helper.IsExists(ref blank, db_.ProtocolBlank.ToList()))
            return Ok(blank);

        db_.ProtocolBlank.Add(blank);
        await db_.SaveChangesAsync();
        return Ok(blank);
    }
}

```

Листинг В.29 – Файл ProtocolsController.cs

```

using EuroProtocolApp.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EuroProtocolApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ProtocolsController : ControllerBase
    {
        ProtocolContext db_;
        public ProtocolsController(ProtocolContext db)
        {
            db_ = db;
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<Protocol>>> Get()
        {
            return await db_.Protocol.ToListAsync();
        }

        [HttpGet("{id}")]
        public async Task<ActionResult<Protocol>> Get(Guid id)
        {
            var protocol = await db_.Protocol.FirstOrDefaultAsync(x => x.Id
== id);

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

            return new ObjectResult(protocol);
        }

        [HttpPost]
        public async Task<ActionResult<Protocol>> Create(Protocol protocol)
        {
            if (protocol == null)

```

```

        return BadRequest();

        if (Helper.IsExists(ref protocol, db_.Protocol.ToList()))
            return Ok(protocol);

        db_.Protocol.Add(protocol);
        await db_.SaveChangesAsync();
        return Ok(protocol);
    }

    [HttpPut]
    public async Task<ActionResult<Protocol>> Update(Protocol protocol)
    {
        if (protocol == null)
            return BadRequest();

        if (!db_.Protocol.Any(x => x.Id == protocol.Id))
            return NotFound();

        db_.Protocol.Update(protocol);
        await db_.SaveChangesAsync();
        return Ok(protocol);
    }

    [HttpDelete("{id}")]
    public async Task<ActionResult<Protocol>> Delete(Guid id)
    {
        var protocol = db_.Protocol.FirstOrDefault(x => x.Id == id);

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

        db_.Protocol.Remove(protocol);
        await db_.SaveChangesAsync();
        return Ok(protocol);
    }
}

```

Листинг В.30 – Файл TrafficLawsController.cs

```

using EuroProtocolApp.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EuroProtocolApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class TrafficLawsController : ControllerBase
    {
        ProtocolContext db_;

        public TrafficLawsController(ProtocolContext db)
        {
            db_ = db;
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<TrafficLaws>>> Get()
        {
            return await db_.TrafficLaws.ToListAsync();
        }
    }
}

```

```

[HttpGet("{id}")]
public async Task<ActionResult<TrafficLaws>> Get(int id)
{
    var laws = await db_.TrafficLaws.FirstOrDefaultAsync(x => x.Id ==
id);

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

    return new ObjectResult(laws);
}

[HttpPost]
public async Task<ActionResult<TrafficLaws>> Create(TrafficLaws laws)
{
    if (laws == null)
        return NotFound();

    if (Helper.IsExists(ref laws, db_.TrafficLaws.ToList()))
        return Ok(laws);

    db_.TrafficLaws.Add(laws);
    await db_.SaveChangesAsync();
    return Ok(laws);
}

[HttpDelete]
public async Task<ActionResult<TrafficLaws>> Delete(int id)
{
    var laws = db_.TrafficLaws.FirstOrDefault(x => x.Id == id);

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

    db_.TrafficLaws.Remove(laws);
    await db_.SaveChangesAsync();
    return Ok(laws);
}
}
}

```

Листинг В.31 – Файл UserBlanksController.cs

```

using EuroProtocolApp.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EuroProtocolApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class UserBlanksController : ControllerBase
    {
        ProtocolContext db_;

        public UserBlanksController(ProtocolContext db)
        {
            db_ = db;
        }

        [HttpGet("{id}")]
        public async Task<ActionResult<UserBlank>> Get(int id)
        {

```

```

        var blank = await db_.UserBlank.FirstOrDefaultAsync(x =>
x.User_id == id);
        if (blank == null)
            return NotFound();
        return new ObjectResult(blank);
    }

    [HttpPost]
    public async Task<ActionResult<UserBlank>> Create(UserBlank blank)
    {
        if (blank == null)
            return BadRequest();

        if (Helper.IsExists(ref blank, db_.UserBlank.ToList()))
            return Ok(blank);

        db_.UserBlank.Add(blank);
        await db_.SaveChangesAsync();
        return Ok(blank);
    }
}

```

Листинг В.32 – Файл UsersController.cs

```

using EuroProtocolApp.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EuroProtocolApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class UsersController : ControllerBase
    {
        ProtocolContext db_;

        public UsersController(ProtocolContext db)
        {
            db_ = db;
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<User>>> Get()
        {
            return await db_.User.ToListAsync();
        }

        [HttpGet("{login}/{password}")]
        public async Task<ActionResult<UserType>> Get(string login, string
password)
        {
            var users = await db_.User.ToListAsync();

            if (users == null)
            {
                return NotFound();
            }

            var user = users.Find(x => x.Login == login && x.Password ==
password);

            if (user == null)
            {

```

```

        return NotFound();
    }

    UserType? type = new UserType() { Type = Types.kNone, User_id = -
1 };

    if (db_.Administrator.Any(x => x.User_id == user.Id))
    {
        type.Type = Types.kAdmin;
        type.User_id = user.Id;
    } else if (db_.Driver.Any(x => x.User_id == user.Id))
    {
        type.Type = Types.kDriver;
        type.User_id = user.Id;
    } else if (db_.InsuranceAgent.Any(x => x.User_id == user.Id))
    {
        type.Type = Types.kAgent;
        type.User_id = user.Id;
    }

    return Ok(type);
}

[HttpPost]
public async Task<ActionResult<User>> Create(User user)
{
    if (user == null)
        return BadRequest();

    if (Helper.IsExists(ref user, db_.User.ToList()))
        return BadRequest();

    db_.User.Add(user);
    await db_.SaveChangesAsync();
    return Ok(user);
}
}
}

```

Листинг В.33 – Файл WitnessController.cs

```

using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using EuroProtocolApp.Models;

namespace EuroProtocolApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class WitnessController : ControllerBase
    {
        ProtocolContext db_;

        public WitnessController(ProtocolContext db)
        {
            db_ = db;
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<Witness>>> Get()
        {
            return await db_.Witness.ToListAsync();
        }
    }
}

```

```

[HttpGet("{id}")]
public async Task<ActionResult<Witness>> Get(Guid id)
{
    var witness = await db_.Witness.FirstOrDefaultAsync(x =>
x.Protocol_id == id);

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

    return new ObjectResult(witness);
}

[HttpPost]
public async Task<ActionResult<Witness>> Create(Witness witness)
{
    if (witness == null)
        return BadRequest();

    if (Helper.IsExists(ref witness, db_.Witness.ToList()))
        return Ok(witness);

    db_.Witness.Add(witness);
    await db_.SaveChangesAsync();
    return Ok(witness);
}
}
}

```

Приложение Г
(рекомендуемое)
Исходный код мобильного приложения

Листинг Г.1 – Файл AdminMainPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<Shell
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.AdministratorPages.AdminMainPage"
    xmlns:local="clr-namespace:MobileProtocolApp.AdministratorPages"
    xmlns:agent="clr-namespace:MobileProtocolApp.InsuranceAgentPages"
    Navigated="Shell_Navigated">

    <TabBar>

        <ShellContent
            Title="Правила ПДД"
            ContentTemplate="{DataTemplate local:LawsPage}"
            Route="Laws"/>

        <ShellContent
            Title="Бланки"
            ContentTemplate="{DataTemplate local:BlanksPage}"
            Route="Blanks"/>

        <ShellContent
            Title="Персональная информация"
            ContentTemplate="{DataTemplate local:InformationPage}"
            Route="InformationPage"/>

        <ShellContent
            Title="Список европротоколов"
            ContentTemplate="{DataTemplate agent:ProtocolsListPage}"
            Route="ProtocolsListPage"/>

        <ShellContent
            Title="ДТП"
            ContentTemplate="{DataTemplate local:AdminRTAPage}"
            Route="AdminRTAPage"/>
    </TabBar>

</Shell>
```

Листинг Г.2 – Файл AdminMainPage.xaml.cs

```
using MobileProtocolApp.Models;
namespace MobileProtocolApp.AdministratorPages;

public partial class AdminMainPage : Shell
{
    const string kInformationPagePath = "//InformationPage";
    Administrator admin_;

    public AdminMainPage(Administrator admin)
    {
        InitializeComponent();
        admin_ = admin;
    }

    private async void Shell_Navigated(object sender, ShellNavigatedEventArgs e)
    {
        if (e.Current.Location.ToString() == kInformationPagePath)
        {

```

```

        var navigationParameter = new Dictionary<string, object> { {
"Information", admin_ } };
        await Shell.Current.GoToAsync(kInfomationPagePath,
navigationParameter);
    }
}

```

Листинг Г.3 – Файл AdminRTAPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.AdministatorPages.AdminRTAPage"
    Title="Оформление ДТП. Администратор">
    <VerticalStackLayout VerticalOptions="Center">
        <Label Text="Выберите кто будет заполнять, а кто будет дополнять
европротокол"
            FontSize="20" Margin="0,0,0,0"/>
        <Picker x:Name="picker_role">
            <Picker.Items>
                <x:String>Я буду заполнять, а второй участник будет дополнять
европротокол</x:String>
                <x:String>Я буду дополнять, а второй участник будет
заполнять европротокол</x:String>
            </Picker.Items>
        </Picker>
        <Button Text="Продолжить" FontSize="20" TextColor="White"
            Background="DodgerBlue" Margin="0,20,0,0" Clicked="ToNextPage"/>
    </VerticalStackLayout>
</ContentPage>

```

Листинг Г.4 – Файл AdminRTAPage.xaml.cs

```

using MobileProtocolApp.Models;
namespace MobileProtocolApp.AdministatorPages;

public partial class AdminMainPage : Shell
{
    const string kInfomationPagePath = "//InformationPage";
    Administrator admin_;

    public AdminMainPage(Administrator admin)
    {
        InitializeComponent();
        admin_ = admin;
    }

    private async void Shell_Navigated(object sender, ShellNavigatedEventArgs
e)
    {
        if (e.Current.Location.ToString() == kInfomationPagePath)
        {
            var navigationParameter = new Dictionary<string, object> { {
"Information", admin_ } };
            await Shell.Current.GoToAsync(kInfomationPagePath,
navigationParameter);
        }
    }
}

```

Листинг Г.5 – Файл BlanksPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.AdministatorPages.BlanksPage"
    Title="Бланки">
    <ScrollView>
        <StackLayout>
            <StackLayout>
                <Label Text="Добавление нового набора бланков" FontSize="20"
FontAttributes="Bold"/>
                <Label Text="Комментарий к набору" FontSize="20"
Margin="0,20,0,0"/>
                <Entry x:Name="entry_comment_" FontSize="20"/>
                <Button Text="Добавить" FontSize="20" Background="DodgerBlue"
Margin="0,20,0,0" TextColor="White" Clicked="AddBlank"/>
            </StackLayout>

            <StackLayout Padding="0,50,0,0">
                <Line X2="410" Stroke="Green" ScaleY="5"/>
            </StackLayout>

            <StackLayout Margin="0,20,0,0">
                <Label Text="Добавление страницы бланка" FontSize="20"
FontAttributes="Bold"/>
                <Label Text="Номер набора для страницы" FontSize="20"
Margin="0,20,0,0"/>
                <Picker x:Name="picker_numbers_blank_" FontSize="20"
Margin="0,10,0,0"></Picker>
                <Button Text="Обновить список" FontSize="20"
TextColor="White"
Background="DodgerBlue" Margin="0,20,0,0" Clicked="GetBlanks"/>
                <Label Text="Тип страницы" FontSize="20" Margin="0,20,0,0"/>
                <Picker x:Name="page_name_" FontSize="20">
                    <Picker.Items>
                        <x:String>Главный оформитель</x:String>
                        <x:String>Альтернативный оформитель</x:String>
                        <x:String>Полный бланк</x:String>
                    </Picker.Items>
                </Picker>
                <Label Text="Файл страницы:" FontSize="20"
Margin="0,20,0,0"/>
                <Label x:Name="lbl_file_name_" Text="Не выбрано"
FontSize="20"/>
                <Button Text="Выбрать" FontSize="20" TextColor="White"
Background="DodgerBlue" Margin="0,20,0,0" Clicked="SelectPage"/>
                <Button Text="Добавить" FontSize="20" TextColor="White"
Background="DodgerBlue" Margin="0,20,0,0" Clicked="AddPage"/>
            </StackLayout>

            <StackLayout Padding="0,50,0,0">
                <Line X2="410" Stroke="Green" ScaleY="5"/>
            </StackLayout>

            <StackLayout Margin="0,20,0,0">
                <Label Text="Удаление страницы бланка" FontSize="20"
FontAttributes="Bold"/>
                <Label Text="Выберите номер страницы:" FontSize="20"
Margin="0,20,0,0"/>
                <Picker x:Name="picker_page_number_" FontSize="20"></Picker>
                <Button Text="Обновить список" FontSize="20"
TextColor="White"
Background="DodgerBlue" Margin="0,20,0,0" Clicked="GetPage"/>
                <Button Text="Удалить" FontSize="20" TextColor="White"

```

```

        Background="Red" Margin="0,20,0,0" Clicked="RemovePage"/>
    </StackLayout>
</StackLayout>
</ScrollView>
</ContentPage>

```

Листинг Г.6 – Файл BlanksPage.xaml.cs

```

using MobileProtocolApp.Models;
using MobileProtocolApp.Services;
namespace MobileProtocolApp.AdministatorPages;

public partial class BlanksPage : ContentPage
{
    byte[]? file_;
    public BlanksPage()
    {
        InitializeComponent();
    }

    private async void AddBlank(object sender, EventArgs e)
    {
        if (entry_comment_.Text == null)
        {
            await DisplayAlert("Error", "Комментарии к бланку не
заполнены", "OK");
            return;
        }

        ProtocolBlankService? service = new();

        if (service == null)
        {
            await DisplayAlert("Error", "ProtocolBlankService is null",
"OK");
            return;
        }

        var blank = new ProtocolBlank() { Comment = entry_comment_.Text };

        await service.Add(blank);
        entry_comment_.Text = string.Empty;
        await DisplayAlert("Message", "Набор успешно добавлен", "OK");
    }

    private async void GetBlanks(object sender, EventArgs e)
    {
        ProtocolBlankService? service = new();

        IEnumerable<ProtocolBlank>? blanks = await service.Get();

        if (blanks == null)
            return;

        foreach (var blank in blanks)
        {
            string line = blank.Id.ToString() + blank.Comment;
            picker_numbers_blank_.Items.Add(line);
        }
        await DisplayAlert("Message", "Список обновлен", "OK");
    }

    private async void SelectPage(object sender, EventArgs e)
    {
        var res = await FilePicker.PickAsync();
    }
}

```

```

        if (res == null)
            return;

        lbl_file_name_.Text = res.FileName;
        var stream = await res.OpenReadAsync();
        file_ = new byte[stream.Length];
        stream.Read(file_, 0, file_.Length);
    }

    private async void AddPage(object sender, EventArgs e)
    {
        PageService? service = new();

        if (service == null)
        {
            await DisplayAlert("Error", "PageService is null", "OK");
            return;
        }

        if (lbl_file_name_.Text == "Не выбрано" || lbl_file_name_.Text ==
null)
        {
            await DisplayAlert("Error", "Не выбран файл страницы
бланка", "OK");
            return;
        }

        if (picker_numbers_blank_.SelectedIndex == -1)
        {
            await DisplayAlert("Error", "Не выбран набор", "OK");
            return;
        }

        if (page_name_.SelectedIndex == -1)
        {
            await DisplayAlert("Error", "Не выбран тип страницы", "OK");
            return;
        }

        MobileProtocolApp.Models.Page page = new()
        {
            Blank_id = picker_numbers_blank_.SelectedIndex + 1,
            File = file_,
            Page_name = (PageName)page_name_.SelectedIndex
        };

        await service.Add(page);
        await DisplayAlert("Message", "Страница успешно добавлена", "OK");
    }

    private async void GetPage(object sender, EventArgs e)
    {
        PageService? service = new();

        IEnumerable<MobileProtocolApp.Models.Page>? pages = await
service.Get();

        if (pages == null)
            return;

        foreach (var page in pages)
        {
            string line = page.Id.ToString();
            picker_page_number_.Items.Add(line);
        }
    }

```

```

        await DisplayAlert("Message", "Список обновлен", "OK");
    }

    private async void RemovePage(object sender, EventArgs e)
    {
        if (picker_page_number_.SelectedIndex == -1)
        {
            await DisplayAlert("Error", "Не выбран номер страницы",
"OK");
            return;
        }

        PageService? service = new();
        await service.Delete(picker_page_number_.SelectedIndex);
        await DisplayAlert("Message", "Страница удалена", "OK");
    }
}

```

Листинг Г.7 – Файл InformationPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="MobileProtocolApp.AdministratorPages.InformationPage"
Title="Персональная информация">
    <VerticalStackLayout
        VerticalOptions="Center"
        HorizontalOptions="Center"
        Margin="60,0,0,0">
        <StackLayout>
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="60"/>
                    <RowDefinition Height="60"/>
                    <RowDefinition Height="60"/>
                    <RowDefinition Height="60"/>
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="150"/>
                    <ColumnDefinition Width="200"/>
                </Grid.ColumnDefinitions>
                <Label Text="Фамилия" FontSize="20" Grid.Row="0"
Grid.Column="0"/>
                <Label x:Name="lbl_surname_" FontSize="20" Grid.Row="0"
Grid.Column="1"/>
                <Label Text="Имя" FontSize="20" Grid.Row="1"
Grid.Column="0"/>
                <Label x:Name="lbl_name_" FontSize="20" Grid.Row="1"
Grid.Column="1"/>
                <Label Text="Отчество" FontSize="20" Grid.Row="2"
Grid.Column="0"/>
                <Label x:Name="lbl_second_name_" FontSize="20" Grid.Row="2"
Grid.Column="1"/>
                <Label Text="Телефон" FontSize="20" Grid.Row="3"
Grid.Column="0"/>
                <Label x:Name="lbl_phone_" FontSize="20" Grid.Row="3"
Grid.Column="1"/>
            </Grid>
        </StackLayout>
    </VerticalStackLayout>
</ContentPage>

```

Листинг Г.8 – Файл InformationPage.xaml.cs

```
using MobileProtocolApp.Models;

namespace MobileProtocolApp.AdministratorPages;

[QueryProperty(nameof(Admin), "Information")]
public partial class InformationPage : ContentPage
{
    private Administrator? admin_;
    public Administrator? Admin
    {
        get => admin_;
        set
        {
            admin_ = value;
            OnPropertyChanged();
        }
    }
    public InformationPage()
    {
        InitializeComponent();
        BindingContext = this;
    }

    protected override void OnNavigatedTo(NavigatedToEventArgs args)
    {
        base.OnNavigatedTo(args);

        lbl_surname_.Text = admin_?.Surname;
        lbl_name_.Text = admin_?.Name;
        lbl_second_name_.Text = admin_?.Second_name;
        lbl_phone_.Text = admin_?.Phone;
    }
}
```

Листинг Г.9 – Файл LawsPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.AdministratorPages.LawsPage"
    Title="Правила ПДД">

    <StackLayout>
        <Label Text="Добавление новых правил ПДД" FontSize="25"
            FontAttributes="Bold"/>
        <Entry x:Name="link_content_" Placeholder="Ссылка на сайт"
            FontSize="20"/>
        <Button Text="Добавить" FontSize="20"
            TextColor="White" BackgroundColor="DodgerBlue"
            Margin="0,10,0,0" Clicked="AddLaws"/>
    </StackLayout>
</ContentPage>
```

Листинг Г.10 – Файл LawsPage.xaml.cs

```
using MobileProtocolApp.Models;
using MobileProtocolApp.Services;
```

```

namespace MobileProtocolApp.AdministatorPages;

public partial class LawsPage : ContentPage
{
    string? file_;
    public LawsPage()
    {
        InitializeComponent();

        private async void AddLaws(object sender, EventArgs e)
        {
            TrafficLawsService? service = new();
            file_ = link_content_.Text;

            if (file_ == null)
            {
                await DisplayAlert("Error", "Не указан ресурс", "OK");
                return;
            }

            if (service == null)
            {
                await DisplayAlert("Error", "TrafficLaws is null", "OK");
                return;
            }

            IEnumerable<TrafficLaws>? laws_list = await service.Get();

            if (laws_list != null)
            {
                if (laws_list.Count() == 1)
                {
                    var laws = new TrafficLaws() { Id = 1, File = file_ };
                    await service.Update(laws);
                    await DisplayAlert("Message", "Ресурс сохранён", "OK");
                } else if (laws_list.Count() == 0)
                {
                    var laws = new TrafficLaws() { File = file_ };
                    await service.Add(laws);
                    await DisplayAlert("Message", "Ресурс сохранён", "OK");
                }
            }
        }
    }
}

```

Листинг Г.11 – Файл RegAdminPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.AdministatorPages.RegAdminPage"
    Title="Регистрация администратора">

    <ScrollView>
        <StackLayout>
            <Grid HorizontalOptions="Center" Margin="0,100,0,0"
                RowSpacing="50">
                <Grid.RowDefinitions>
                    <RowDefinition Height="50"/>
                    <RowDefinition Height="50"/>
                    <RowDefinition Height="50"/>
                    <RowDefinition Height="50"/>
                    <RowDefinition Height="50"/>
                </Grid.RowDefinitions>
            </Grid>
        </StackLayout>
    </ScrollView>

```

```

        <RowDefinition Height="50"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="100"/>
        <ColumnDefinition Width="200"/>
    </Grid.ColumnDefinitions>
    <Label Text="Логин" FontSize="20" Grid.Row="0"
Grid.Column="0"/>
    <Entry x:Name="login_" FontSize="20" Grid.Row="0"
Grid.Column="1"/>
    <Label Text="Пароль" FontSize="20" Grid.Row="1"
Grid.Column="0"/>
    <Entry x:Name="password_" FontSize="20" Grid.Row="1"
Grid.Column="1" IsPassword="True"/>
    <Label Text="Фамилия" FontSize="20" Grid.Row="2"
Grid.Column="0"/>
    <Entry x:Name="surname_" FontSize="20" Grid.Row="2"
Grid.Column="1"/>
    <Label Text="Имя" FontSize="20" Grid.Row="3"
Grid.Column="0"/>
    <Entry x:Name="name_" FontSize="20" Grid.Row="3"
Grid.Column="1"/>
    <Label Text="Отчество" FontSize="20" Grid.Row="4"
Grid.Column="0"/>
    <Entry x:Name="second_name_" FontSize="20" Grid.Row="4"
Grid.Column="1"/>
    <Label Text="Телефон" FontSize="20" Grid.Row="5"
Grid.Column="0"/>
    <Entry x:Name="phone_" FontSize="20" Grid.Row="5"
Grid.Column="1"/>
    </Grid>
    <Button Text="Зарегистрироваться" TextColor="White"
        BackgroundColor="DodgerBlue" FontSize="20"
Margin="0,75,0,0"
        Clicked="AddAdmin"/>
    </StackLayout>
</ScrollView>
</ContentPage>

```

Листинг Г.12 – Файл RegAdminPage.xaml.cs

```

using MobileProtocolApp.Services;
using MobileProtocolApp.Models;
using MobileProtocolApp.HelperCS;

namespace MobileProtocolApp.AdministratorPages;

public partial class RegAdminPage : ContentPage
{
    AuthPage page_;
    public RegAdminPage(AuthPage page)
    {
        InitializeComponent();
        page_ = page;
    }

    private bool FieldsIsFill()
    {
        return login_.Text != null && password_ != null
            && surname_.Text != null && name_ != null
            && second_name_.Text != null && phone_.Text != null;
    }

    private async void AddAdmin(object sender, EventArgs e)

```

```

{
    if (!FieldsIsFill())
    {
        await DisplayAlert("Error", "Не все поля заполнены", "OK");
        return;
    }

    if (!Helper.IsLogin(login_.Text))
    {
        await DisplayAlert("Error", "Логин не соответствует шаблону
электронной почты", "OK");
        return;
    }

    if (!Helper.IsPassword(password_.Text))
    {
        await DisplayAlert("Error", "Плохой пароль. Пароль должен иметь:"
+
        "\n\tdлину от 8 до 15 символов;" +
        "\n\tанглийские заглавные и прописные буквы" +
        "\n\tспециальные символы" +
        "\n\tцифры", "OK");
        return;
    }

    var user_serv = new UserService();
    var admin_serv = new AdministratorService();

    User user = new()
    {
        Traffic_laws_id = Helper.kLawsId,
        Login = login_.Text,
        Password = password_.Text
    };

    var response = await user_serv.Add(user);

    if (response == null)
    {
        await DisplayAlert("Error", "Не получилось зарегистрировать
аккаунт", "OK");
        return;
    }

    Administrator admin = new()
    {
        User_id = response.Id,
        Surname = surname_.Text,
        Name = name_.Text,
        Second_name = second_name_.Text,
        Phone = phone_.Text
    };

    var resp = await admin_serv.Add(admin);

    if (resp == null)
    {
        await DisplayAlert("Error", "Не получилось зарегистрировать
администратора", "OK");
        return;
    }

    await DisplayAlert("Message", "Пользователь зарегистрирован", "OK");
    await Navigation.PopAsync();
    page_.ToAuth();
}

```

```
}
```

Листинг Г.13 – Файл AllProtocolPage.cs

```
using MobileProtocolApp.Services;
using MobileProtocolApp.HelperCS;

namespace MobileProtocolApp.DriverPages;

public class AllProtocolPage : ContentPage
{
    Guid Guid { get; set; }
    public AllProtocolPage(MobileProtocolApp.Models.Page schema, Guid guid)
    {
        Guid = guid;
        string xml = System.Text.Encoding.UTF8.GetString(schema.File);
        this.LoadFromXaml(xml);
        Get<Button>(Helper.button_back_2_).Clicked += Back;
        Get<Button>(Helper.button_send_2_).Clicked += SaveProtocol;
    }

    public T Get<T>(string name)
    {
        return this.FindByName<T>(name);
    }

    private async void Back(object sender, EventArgs e)
    {
        var protocol_serv = new ProtocolService();
        await protocol_serv.Delete(Guid);
        await Navigation.PopModalAsync();
    }

    private async void SaveProtocol(object sender, EventArgs e)
    {
        await Navigation.PushModalAsync(new SignaturePage());
    }

    protected override async void OnAppearing()
    {
        try
        {
            ProtocolService protocol_serv = new();
            CarService car_serv = new();
            DriverService driver_serv = new();
            ImageService image_serv = new();
            ParticipantService participant_serv = new();
            ParticipantCarService participant_car_serv = new();
            WitnessService witness_serv = new();
            OwnerService owner_serv = new();

            var protocol = await protocol_serv.Get(Guid);

            if (protocol == null)
                throw new ArgumentNullException("Protocol service return null");

            var participants = (await participant_serv.Get(protocol.Id)).ToList();
            var participant_cars = (await participant_car_serv.Get(protocol.Id)).ToList();
            var drivers = (await driver_serv.Get()).ToList();
```

```

        var driver_1 = drivers?.FirstOrDefault(x => x.Id ==
participants?[0].Driver_id);
        var driver_2 = drivers?.FirstOrDefault(x => x.Id ==
participants?[1].Driver_id);

        if (driver_1 == null) throw new ArgumentNullException("driver_1
is null");
        if (driver_2 == null) throw new ArgumentNullException("driver_2
is null");

        var cars_1 = (await car_serv.Get(driver_1.Id)?.ToList());
        var cars_2 = (await car_serv.Get(driver_2.Id)?.ToList());

        if (cars_1 == null) throw new ArgumentNullException("cars_1 is
null");
        if (cars_2 == null) throw new ArgumentNullException("cars_2 is
null");

        var car_1 = cars_1.FirstOrDefault(x => x.Id ==
participant_cars?[0].Car_id);
        var car_2 = cars_2.FirstOrDefault(x => x.Id ==
participant_cars?[1].Car_id);

        if (car_1 == null) throw new ArgumentNullException("car_1 is
null");
        if (car_2 == null) throw new ArgumentNullException("car_2 is
null");

        var owner_1 = await owner_serv.Get(car_1.Owner_id);
        var owner_2 = await owner_serv.Get(car_2.Owner_id);

        var witness = await witness_serv.Get(protocol.Id);
        var res = await image_serv.Get(protocol.Id);
        var images = (res)?.ToList();

        Get<Label>(Helper.protocol_guid_lbl).Text += " " + protocol.Id;

        Helper.FillProtocol(protocol, this);
        Helper.FillParticipant(participants?[0], this);
        Helper.FillParticipant(participants?[1], this, true);
        Helper.FillParticipantCar(participant_cars?[0], this);
        Helper.FillParticipantCar(participant_cars?[1], this, true);
        Helper.FillDriver(driver_1, this);
        Helper.FillDriver(driver_2, this, true);
        Helper.FillCar(car_1, this);
        Helper.FillCar(car_2, this, true);
        Helper.FillOwner(owner_1, this);
        Helper.FillOwner(owner_2, this, true);
        Helper.FillWitness(witness, this);
        Helper.FillImages(images, this);

        base.OnAppearing();
    }
    catch (Exception ex)
    {
        await DisplayAlert("Error", ex.Message, "OK");
        return;
    }
    await DisplayAlert("Message", "Данные успешно заполнены", "OK");
}
}

```

Листинг Г.14 – Файл AlternativeWriterPage.cs

```

using MobileProtocolApp.Models;
using MobileProtocolApp.Services;
using MobileProtocolApp.HelperCS;
namespace MobileProtocolApp.DriverPages;

public class AlternativeWriterPage : ContentPage
{
    Models.Page? Page { get; set; }
    Models.Page All_blank { get; set; }
    Driver? Driver { get; set; }
    Car? Car { get; set; }
    List<FileResult> Photos { get; set; }
    FileResult? Damage_car { get; set; }
    FileResult? Reg_mark { get; set; }

    public AlternativeWriterPage(Models.Page? page, Driver driver, Owner
owner, Car car,
        Models.Page next_page)
    {
        Page = page;
        All_blank = next_page;
        Driver = driver;
        Car = car;
        Photos = new();
        string xml = System.Text.Encoding.UTF8.GetString(Page.File);
        this.LoadFromXaml(xml);
        Get<ImageButton>(Helper.button_photo_1_).Clicked += SelectPhoto;
        Get<ImageButton>(Helper.button_photo_2_).Clicked += SelectPhoto;
        Get<ImageButton>(Helper.button_photo_3_).Clicked += SelectPhoto;
        Get<ImageButton>(Helper.button_damage_car_).Clicked +=
SelectDamageCar;
        Get<ImageButton>(Helper.button_state_reg_mark_).Clicked +=
SelectRegMark;
        Get<Button>(Helper.button_send_).Clicked += AddProtocol;
        try
        {
            Helper.FillCar(car, this);
            Helper.FillDriver(driver, this);
            Helper.FillOwner(owner, this);

        }
        catch (Exception ex)
        {
            this.Background = Colors.Red;
            Get<Entry>("surname_witness_").Text = ex.Message;
        }
    }
    public T Get<T>(string name)
    {
        return this.FindByName<T>(name);
    }

    private async void SelectPhoto(object sender, EventArgs e)
    {
        var photo = await MediaPicker.PickPhotoAsync();

        if (photo == null)
        {
            await DisplayAlert("Error", "Не удалось добавить фото", "OK");
            return;
        }

        var button = sender as ImageButton;

```

```

        button.Source = ImageSource.FromFile(photo.FullPath);

        if (Photos.Count == 3)
            Photos.RemoveAt(0);

        Photos.Add(photo);
    }

    private async void SelectDamageCar(object sender, EventArgs e)
    {
        var photo = await MediaPicker.PickPhotoAsync();

        if (photo == null)
        {
            await DisplayAlert("Error", "Не удалось добавить фото", "OK");
            return;
        }

        var button = sender as ImageButton;
        button.Source = ImageSource.FromFile(photo.FullPath);
        Damage_car = photo;
    }

    private async void SelectRegMark(object sender, EventArgs e)
    {
        var photo = await MediaPicker.PickPhotoAsync();

        if (photo == null)
        {
            await DisplayAlert("Error", "Не удалось добавить фото", "OK");
            return;
        }

        var button = sender as ImageButton;
        button.Source = ImageSource.FromFile(photo.FullPath);
        Reg_mark = photo;
    }

    private async void AddProtocol(object sender, EventArgs e)
    {
        Guid protocol_id;
        try
        {
            if (Get<Entry>(Helper.protocol_number_).Text == null
                || Get<Picker>(Helper.circumstances_rta_).SelectedIndex == -1
                || Get<Editor>(Helper.explanations_).Text == null
                || Photos.Count != 3
                || Get<Picker>(Helper.hit_place_number_).SelectedIndex == -1)
            {
                await DisplayAlert("Error", "Не все поля заполнены", "OK");
                return;
            }

            List<byte[]> photos = new();
            List<Models.Image> images = new();
            byte[]? damage_car = null, reg_mark = null;

            if (!Guid.TryParse(Get<Entry>(Helper.protocol_number_).Text, out
protocol_id))
                throw new ArgumentException("Вы ввели номер неверного
формата");

            ProtocolService protocol_serv = new();
            ImageService image_serv = new();
            ParticipantCarService participant_car_serv = new();
            ParticipantService participant_serv = new();

```

```

        foreach (var photo in Photos)
        {
            var res = await Helper.ToByte(photo);
            if (res != null)
                photos.Add(res);
        }

        if (photos.Count < 3)
            throw new ArgumentNullException("Не удалось конвертировать
фотографии");

        if (Damage_car != null)
            damage_car = await Helper.ToByte(Damage_car);

        if (Reg_mark != null)
            reg_mark = await Helper.ToByte(Reg_mark);

        foreach (var photo in photos)
        {
            images.Add(
                new Models.Image()
                {
                    Protocol_id = protocol_id,
                    Content = photo
                });
        }

        Participant participant = new()
        {
            Protocol_id = protocol_id,
            Driver_id = Driver.Id,
            Circumstances_rta =
Get<Picker>(Helper.circumstances_rta_).SelectedIndex,
            Explanations = Get<Editor>(Helper.explanations_).Text
        };

        ParticipantCar p_car = new()
        {
            Protocol_id = protocol_id,
            Car_id = Car.Id,
            Damage_place =
Get<Picker>(Helper.hit_place_number_).SelectedIndex,
            Damage_image = damage_car,
            Reg_number_image = reg_mark
        };

        foreach (var photo in images)
        {
            var image_res = await image_serv.Add(photo);
            if (image_res == null)
                throw new ArgumentNullException("Image service return
null");
        }

        var participant_res = await participant_serv.Add(participant);
        if (participant_res == null)
            throw new ArgumentNullException("Participant service return
null");

        var car_res = await participant_car_serv.Add(p_car);
        if (car_res == null)
            throw new ArgumentNullException("Participant car service
return null");

    } catch (Exception ex)

```

```

        {
            await DisplayAlert("Error", ex.Message, "OK");
            return;
        }
        await DisplayAlert("Message", "Успешно добавлено", "OK");
        WaitParticipants(protocol_id);
    }

private async void WaitParticipants(Guid guid)
{
    ContentPage page = new ContentPage()
    {
        Content = new StackLayout()
        {
            {
                VerticalOptions = LayoutOptions.Center,
                HorizontalOptions = LayoutOptions.Center,
                Children =
                {
                    new Label() {Text="Ждем второго участника", FontSize
= 25},
                    new ActivityIndicator() { IsRunning = true, Color =
Colors.SeaGreen }
                }
            }
        };
        ParticipantService participant_serv = new();
        List<Participant>? participants = new();

        while (participants?.Count != 2)
        {
            participants = (await participant_serv.Get(guid))?.ToList();
            System.Threading.Thread.Sleep(15000);
        }
        await Navigation.PushModalAsync(new AllProtocolPage(All_blank,
guid));
    }
}

```

Листинг Г.15 – Файл CarInformPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.DriverPages.CarInformPage"
    Title="Информация о ТС">
    <ScrollView>

        <StackLayout Margin="0,20,0,0">
            <Label Text="Сведения о транспортном средстве" FontSize="25"
FontAttributes="Bold"/>
            <StackLayout Padding="0,5,0,0">
                <Line X2="410" Stroke="Red" ScaleY="5"/>
            </StackLayout>

            <StackLayout Padding="0,20,0,40">
                <StackLayout>
                    <Label Text="Карта ТС" FontSize="20"/>
                    <Entry x:Name="car_brand_" FontSize="20"
Margin="0,0,0,15"/>
                    <Label Text="Модель ТС" FontSize="20"/>
                    <Entry x:Name="car_model_" FontSize="20"
Margin="0,0,0,15"/>
                    <Label Text="Государственный регистрационный знак ТС"
FontSize="20"/>
                </StackLayout>
            </StackLayout>
        </ScrollView>
    </ContentPage>

```

```

        Margin="0,0,0,15"/>
        <Entry x:Name="state_reg_mark_" FontSize="20"
        <Label Text="Идентификационный номер (VIN) ТС"
        FontSize="20"/>
        <Entry x:Name="vin_number_" FontSize="20"
        Margin="0,0,0,15"/>
        <Label Text="Свидетельство о регистрации ТС"
        FontSize="20"/>
        <Entry x:Name="sts_number_" FontSize="20"
        Margin="0,0,0,15"/>
        <Label Text="Собственник ТС" FontSize="20"/>
        <Entry x:Name="car_owner_" FontSize="20"
        Margin="0,0,0,15"/>
        <Label Text="Адрес собственника" FontSize="20"/>
        <Entry x:Name="address_owner_" FontSize="20"
        Margin="0,0,0,15"/>
        <Label Text="Телефон собственника" FontSize="20"/>
        <Entry x:Name="phone_owner_" FontSize="20"
        Margin="0,0,0,15"/>
        <Label Text="Страховщик" FontSize="20"/>
        <Picker x:Name="insurance_company_" Margin="0,0,0,15"
        FontSize="20">
            <Picker.Items>
                <x:String>АльфаСтрахование</x:String>
                <x:String>Ингосстрах</x:String>
                <x:String>Росгосстрах</x:String>
                <x:String>Согаз</x:String>
                <x:String>Ренессанс</x:String>
            </Picker.Items>
        </Picker>
        <Label Text="Страховой полис" FontSize="20"/>
        <Grid Margin="0,0,0,15">
            <Grid.RowDefinitions>
                <RowDefinition Height="50"/>
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="65"/>
                <ColumnDefinition Width="90"/>
                <ColumnDefinition Width="80"/>
                <ColumnDefinition Width="170"/>
            </Grid.ColumnDefinitions>
            <Label Text="Серия" FontSize="20" Grid.Row="0"
            Grid.Column="0"/>
            <Entry x:Name="osago_series_" FontSize="20"
            Grid.Row="0" Grid.Column="1"/>
            <Label Text="Номер" FontSize="20" Grid.Row="0"
            Grid.Column="2"/>
            <Entry x:Name="osago_number_" FontSize="20"
            Grid.Row="0" Grid.Column="3"/>
        </Grid>
    </StackLayout>
    <Button x:Name="add_button" Text="Добавить" TextColor="White"
        BackgroundColor="DodgerBlue" FontSize="20"
        Clicked="AddCar"/>
    </StackLayout>
</StackLayout>
</ScrollView>
</ContentPage>

```

Листинг Г.16 – Файл CarInformPage.xaml.cs

```
using MobileProtocolApp.Models;
```

```

using MobileProtocolApp.Services;

namespace MobileProtocolApp.DriverPages;

public partial class CarInformPage : ContentPage
{
    DriverCarPage? parent_page_;
    public CarInformPage(DriverCarPage page)
    {
        InitializeComponent();
        parent_page_ = page;
    }
    public CarInformPage(Car car, Owner owner)
    {
        InitializeComponent();
        add_button.IsVisible = false;
        car_brand_.Text = car.Brand;
        car_model_.Text = car.Model;
        state_reg_mark_.Text = car.Reg_number;
        vin_number_.Text = car.VIN_number;
        sts_number_.Text = car.Reg_certificate;
        car_owner_.Text = $"{owner?.Surname} {owner?.Name}
{owner?.Second_name}";
        address_owner_.Text = owner?.Address;
        phone_owner_.Text = owner?.Phone;
        insurance_company_.SelectedIndex = car.Insurer;
        osago_series_.Text = car.OSAGO?[..3];
        osago_number_.Text = car.OSAGO?[3..];
    }

    private async void AddCar(object sender, EventArgs e)
    {
        var fio = car_owner_.Text.Split(' ');

        if (car_owner_.Text == null
            || address_owner_.Text == null
            || phone_owner_.Text == null)
        {
            await DisplayAlert("Error", "Не введены данные о
собственнике", "OK");
            return;
        }

        var owner = new Owner()
        {
            Surname = fio[0],
            Name = fio[1],
            Second_name = fio[2],
            Address = address_owner_.Text,
            Phone = phone_owner_.Text
        };

        if (car_brand_.Text == null || car_model_.Text == null
            || state_reg_mark_.Text == null || vin_number_.Text == null
            || sts_number_.Text == null ||
insurance_company_.SelectedIndex == -1
            || osago_number_.Text == null || osago_number_.Text == null)
        {
            await DisplayAlert("Error", "Не введены данные о машине", "OK");
            return;
        }

        var car = new Car()
        {
            Brand = car_brand_.Text,
            Model = car_model_.Text,

```

```

        Reg_number = state_reg_mark_.Text,
        VIN_number = vin_number_.Text,
        Reg_certificate = sts_number_.Text,
        Insurer = insurance_company_.SelectedIndex,
        OSAGO = osago_series_.Text + osago_number_.Text,
    };

    var owner_serv = new OwnerService();
    var car_serv = new CarService();

    var owner_response = await owner_serv.Add(owner);
    if (owner_response == null)
    {
        await DisplayAlert("Error", "Собственник не добавлен",
"OK");
        return;
    }
    car.Driver_id = parent_page_.Driver.Id;
    car.Owner_id = owner_response.Id;
    var car_response = await car_serv.Add(car);
    if (car_response == null)
    {
        await DisplayAlert("Error", "Машина не добавлена", "OK");
        return;
    }
    parent_page_.Cars.Add(car);
    await Navigation.PopAsync();
}
}

```

Листинг Г.17 – Файл ChoiceWritterPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.DriverPages.ChoiceWritterPage">

    <VerticalStackLayout VerticalOptions="Center">
        <Label Text="Выберите кто будет заполнять, а кто будет дополнять
европротокол"
            FontSize="20" Margin="0,0,0,0"/>
        <Picker x:Name="picker_role">
            <Picker.Items>
                <x:String>Я буду заполнять, а второй участник будет дополнять
европротокол</x:String>
                <x:String>Я буду дополнять, а второй участник будет
заполнять европротокол</x:String>
            </Picker.Items>
        </Picker>
        <Label Text="Выберите своё ТС:" Margin="0,20,0,0"/>
        <Picker x:Name="picker_ts_"></Picker>
        <Button Text="Продолжить" FontSize="20" TextColor="White"
            Background="DodgerBlue" Margin="0,20,0,0"
            Clicked="ToNextPage"/>
    </VerticalStackLayout>
</ContentPage>

```

Листинг Г.18 – Файл ChoiceWritterPage.xaml.cs

```

using MobileProtocolApp.Models;

namespace MobileProtocolApp.DriverPages;

```



```

        Detail="{Binding Reg_number}"
        TextColor="White"
        DetailColor="LightGreen">
    </TextCell>
</DataTemplate>
</ListView.ItemTemplate>
</ListView>
<Button Text="Обновить" TextColor="White"
        BackgroundColor="Orange" FontSize="20" Clicked="UpdateCar"/>
<Button Text="Добавить авто" TextColor="White"
        BackgroundColor="DodgerBlue" FontSize="20"
Clicked="AddCar"/>
</StackLayout>

</ScrollView>
</ContentPage>

```

Листинг Г.20 – Файл DriverCarPage.xaml.cs

```

using MobileProtocolApp.Models;
using System.Collections.ObjectModel;
namespace MobileProtocolApp.DriverPages;

[QueryProperty(nameof(Cars), "Cars")]
[QueryProperty(nameof(Owners), "Owner")]
[QueryProperty(nameof(Driver), "Driver")]
public partial class DriverCarPage : ContentPage
{
    List<Car>? cars_;
    List<Owner>? owners_;
    Driver? driver_;

    public ObservableCollection<Car>? CarList { get; set; }

    public List<Car>? Cars
    {
        get => cars_;
        set
        {
            cars_ = value;
            OnPropertyChanged();
        }
    }
    public List<Owner>? Owners
    {
        get => owners_;
        set
        {
            owners_ = value;
            OnPropertyChanged();
        }
    }

    public Driver? Driver
    {
        get => driver_;
        set
        {
            driver_ = value;
            OnPropertyChanged();
        }
    }

    public DriverCarPage()

```

```

{
    InitializeComponent();
    CarList = new();
    BindingContext = this;
}

private async void CarTapped(object sender, ItemTappedEventArgs e)
{
    try
    {
        Car? car = cars_[e.ItemIndex];
        Owner? owner = owners_[e.ItemIndex];
        if (car == null || owner == null)
        {
            await DisplayAlert("Error", "Информации нет", "OK");
            return;
        }
        var car_page = new CarInformPage(car, owner);
        await Navigation.PushAsync(car_page);
    } catch (Exception ex)
    {
        await DisplayAlert("Error", $"{ex.Message}", "OK");
        return;
    }
}

private async void AddCar(object sender, EventArgs e)
{
    if (cars_ != null && driver_ != null)
    {
        var page = new CarInformPage(this);
        await Navigation.PushAsync(page);
    }
}

private void UpdateCar(object sender, EventArgs e)
{
    if (Cars == null)
        return;
    CarList.Clear();
    foreach (var car in Cars)
    {
        CarList.Add(car);
    }
}
}

```

Листинг Г.21 – Файл DriverInformationPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.DriverPages.DriverInformationPage"
    Title="Мои данные">

    <ScrollView>
        <StackLayout>

            <StackLayout Padding="5,0,0,0">
                <Label Text="Персональная информация" FontSize="25"
                    FontAttributes="Bold"/>

                <StackLayout Padding="0,5,0,20">

```

```

        <Line X2="410" Stroke="Red" ScaleY="5"/>
    </StackLayout>

    <StackLayout>
        <Label Text="Фамилия" FontSize="20"/>
        <Entry x:Name="surname_driver_" FontSize="20"
Margin="0,0,0,15"
            TextChanged="EnableButtons"/>
        <Label Text="Имя" FontSize="20"/>
        <Entry x:Name="name_driver_" FontSize="20"
Margin="0,0,0,15"
            TextChanged="EnableButtons"/>
        <Label Text="Отчество" FontSize="20"/>
        <Entry x:Name="second_name_driver_" FontSize="20"
Margin="0,0,0,15"
            TextChanged="EnableButtons"/>
        <Label Text="Телефон" FontSize="20"/>
        <Entry x:Name="phone_driver_" FontSize="20"
Margin="0,0,0,15"
            TextChanged="EnableButtons"/>
        <Label Text="Дата рождения" FontSize="20"/>
        <DatePicker x:Name="birthday_driver_" FontSize="20"
Margin="0,0,0,15" IsEnabled="False"/>
        <Label Text="Адрес проживания" FontSize="20"/>
        <Entry x:Name="address_driver_" FontSize="20"
Margin="0,0,0,15"
            TextChanged="EnableButtons"/>
        <Label Text="Права" FontSize="20"/>
        <Grid Margin="0,0,0,15">
            <Grid.RowDefinitions>
                <RowDefinition Height="50"/>
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="65"/>
                <ColumnDefinition Width="90"/>
                <ColumnDefinition Width="80"/>
                <ColumnDefinition Width="170"/>
            </Grid.ColumnDefinitions>
            <Label Text="Серия" FontSize="20" Grid.Row="0"
Grid.Column="0"/>
                <Entry x:Name="series_lic_driver_" FontSize="20"
Grid.Row="0" Grid.Column="1"
                    TextChanged="EnableButtons"/>
                <Label Text="Номер" FontSize="20" Grid.Row="0"
Grid.Column="2"/>
                <Entry x:Name="number_lic_driver_" FontSize="20"
Grid.Row="0" Grid.Column="3"
                    TextChanged="EnableButtons"/>
            </Grid>
        <Label Text="Действительны до" FontSize="20"/>
        <DatePicker x:Name="date_lic_driver_" FontSize="20"
DateSelected="DateChanged"/>
    </StackLayout>

    </StackLayout>
    <Button x:Name="edit_button_" Text="Исправить" TextColor="White"
FontSize="20"
        Background="Gray" Margin="0,20,0,0" IsEnabled="False"
Clicked="UpdateDriver"/>
    <Button x:Name="back_button_" Text="Отменить изменения"
TextColor="White" FontSize="20"
        Margin="0,5,0,0" Background="Gray" IsEnabled="False"
Clicked="BackFields"/>

</StackLayout>

```

```

</ScrollView>

</ContentPage>

```

Листинг Г.22 – Файл DriverInformationPage.xaml.cs

```

using MobileProtocolApp.Models;
using MobileProtocolApp.Services;

namespace MobileProtocolApp.DriverPages;
[QueryProperty(nameof(Driver), "Driver")]
public partial class DriverInformationPage : ContentPage
{
    Driver? driver_;

    public Driver? Driver
    {
        get => driver_;
        set
        {
            driver_ = value;
            OnPropertyChanged();
        }
    }

    public DriverInformationPage()
    {
        InitializeComponent();
        BindingContext = this;
    }

    private void SetFields()
    {
        surname_driver_.Text = driver_.Surname;
        name_driver_.Text = driver_.Name;
        second_name_driver_.Text = driver_.Second_name;
        phone_driver_.Text = driver_.Phone;
        birthday_driver_.Date = driver_.Birthday;
        address_driver_.Text = driver_.Address;
        series_lic_driver_.Text = driver_.Driver_lic?[..4];
        number_lic_driver_.Text = driver_.Driver_lic?[4..];
        date_lic_driver_.Date = driver_.Lic_date;

        edit_button_.IsEnabled = false;
        edit_button_.Background = Colors.Gray;
        back_button_.IsEnabled = false;
        back_button_.Background = Colors.Gray;
    }

    private bool FieldsIsFill()
    {
        return surname_driver_.Text != null && name_driver_.Text != null &&
second_name_driver_.Text != null
        && phone_driver_.Text != null && address_driver_.Text != null
        && series_lic_driver_.Text != null && number_lic_driver_.Text !=
null;
    }

    protected override void OnNavigatedTo(NavigatedToEventArgs args)
    {
        base.OnNavigatedTo(args);

        if (driver_ == null)
            return;
    }
}

```

```

        SetFields();
    }

    private void EnableButtons(object sender, TextChangedEventArgs e)
    {
        edit_button_.IsEnabled = true;
        edit_button_.Background = Colors.Green;
        back_button_.IsEnabled = true;
        back_button_.Background = Colors.Crimson;
    }

    private void BackFields(object sender, EventArgs e)
    {
        SetFields();
    }

    private async void UpdateDriver(object sender, EventArgs e)
    {
        if (!FieldsIsFill())
        {
            await DisplayAlert("Error", "Не все поля заполнены", "OK");
            return;
        }

        var driver_serv = new DriverService();

        driver_.Surname = surname_driver_.Text;
        driver_.Name = name_driver_.Text;
        driver_.Second_name = second_name_driver_.Text;
        driver_.Phone = phone_driver_.Text;
        driver_.Birthday = birthday_driver_.Date;
        driver_.Address = address_driver_.Text;
        driver_.Driver_lic = series_lic_driver_.Text +
number_lic_driver_.Text;
        driver_.Lic_date = date_lic_driver_.Date;

        var response = await driver_serv.Update(driver_);

        if (response == null)
        {
            await DisplayAlert("Error", "Не получилось обновить данные
водителя", "OK");
            return;
        }
        await DisplayAlert("Message", "Данные водителя обновлены", "OK");
        SetFields();
    }

    private void DateChanged(object sender, DateChangedEventArgs e)
    {
        edit_button_.IsEnabled = true;
        edit_button_.Background = Colors.Green;
        back_button_.IsEnabled = true;
        back_button_.Background = Colors.Crimson;
    }
}

```

Листинг Г.23 – Файл DriverLawsPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.DriverPages.DriverLawsPage"
    Title="Правила ПДД">

```

```

        <ScrollView>
            <WebView x:Name="web_view_" />
        </ScrollView>

</ContentPage>

```

Листинг Г.24 – Файл DriverLawsPage.xaml.cs

```

using MobileProtocolApp.Models;
using System.Windows.Input;
using Microsoft.Maui.Controls;

namespace MobileProtocolApp.DriverPages;
[QueryProperty(nameof(Laws), "Laws")]
public partial class DriverLawsPage : ContentPage
{
    TrafficLaws? laws_;

    public TrafficLaws? Laws
    {
        get => laws_;
        set
        {
            laws_ = value;
            OnPropertyChanged();
        }
    }

    public DriverLawsPage()
    {
        InitializeComponent();
        BindingContext = this;
    }

    protected override void OnNavigatedFrom(NavigatedFromEventArgs args)
    {
        base.OnNavigatedFrom(args);
        web_view_.Source = new UrlWebViewSource { Url = Laws?.File };
    }
}

```

Листинг Г.25 – Файл DriverShellPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<Shell
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.DriverPages.DriverShellPage"
    xmlns:local="clr-namespace:MobileProtocolApp.DriverPages"
    Navigated="ShellNavigated">

    <TabBar>

        <ShellContent
            Title="Правила ПДД"
            ContentTemplate="{DataTemplate local:DriverLawsPage}"
            Route="DriverLawsPage"/>

        <ShellContent
            Title="Мои данные"
            ContentTemplate="{DataTemplate local:DriverInformationPage}"
            Route="DriverInformationPage"/>
    </TabBar>
</Shell>

```

```

        <ShellContent
            Title="Мои ТС"
            ContentTemplate="{DataTemplate local:DriverCarPage}"
            Route="DriverCarPage"/>

        <ShellContent
            Title="ДТП"
            ContentTemplate="{DataTemplate local:RTAPage}"
            Route="RTAPage"/>

    </TabBar>
</Shell>

```

Листинг Г.26 – Файл DriverShellPage.xaml.cs

```

using MobileProtocolApp.Models;
namespace MobileProtocolApp.DriverPages;

public partial class DriverShellPage : Shell
{
    const string kInformationPath = "//DriverInformationPage";
    const string kLaws = "//DriverLawsPage";
    const string kDriverCarPage = "//DriverCarPage";
    const string kRTAPage = "//RTAPage";
    TrafficLaws laws_;
    Driver driver_;
    List<Car> cars_;
    List<Owner> owner_;
    List<MobileProtocolApp.Models.Page> pages_;

    public DriverShellPage(TrafficLaws laws, Driver driver, List<Car> cars,
        List<Owner> owner,
        List<MobileProtocolApp.Models.Page> pages)
    {
        InitializeComponent();
        laws_ = laws;
        driver_ = driver;
        cars_ = cars;
        owner_ = owner;
        pages_ = pages;
    }

    private async void ShellNavigated(object sender, ShellNavigatedEventArgs
e)
    {
        if (e.Current.Location.ToString() == kLaws)
        {
            var parameter = new Dictionary<string, object> { { "Laws",
laws_ } };
            await Shell.Current.GoToAsync(kLaws, parameter);
        } else if (e.Current.Location.ToString() == kInformationPath)
        {
            var parameter = new Dictionary<string, object> { { "Driver",
driver_ } };
            await Shell.Current.GoToAsync(kInformationPath, parameter);
        } else if (e.Current.Location.ToString() == kDriverCarPage)
        {
            var parameter = new Dictionary<string, object> { { "Cars", cars_
},
{ "Owner", owner_ }, { "Driver", driver_ } };
            await Shell.Current.GoToAsync(kDriverCarPage, parameter);
        } else if (e.Current.Location.ToString() == kRTAPage)
        {

```

```

        var parameter = new Dictionary<string, object> { { "Pages",
pages_ },
        { "Cars", cars_ }, { "Owner", owner_ }, { "Driver", driver_ }
};
        await Shell.Current.GoToAsync(kRTAPage, parameter);
    }
}
}

```

Листинг Г.27 – Файл MainWriterPage.cs

```

using MobileProtocolApp.Models;
using MobileProtocolApp.Services;
using MobileProtocolApp.HelperCS;

namespace MobileProtocolApp.DriverPages;

public class MainWriterPage : ContentPage
{
    Models.Page? Page { get; set; }
    Models.Page? All_blank { get; set; }
    Driver? Driver { get; set; }
    Car? Car { get; set; }
    List<FileResult> Photos { get; set; }
    FileResult? Schema { get; set; }
    FileResult? Damage_car { get; set; }
    FileResult? Reg_mark { get; set; }

    Guid Guid { get; set; }

    public MainWriterPage(Models.Page? page, Driver driver, Owner owner, Car
car, Models.Page next_page)
    {
        Page = page;
        All_blank = next_page;
        Driver = driver;
        Car = car;
        Photos = new();
        string xml = System.Text.Encoding.UTF8.GetString(Page.File);
        this.LoadFromXaml(xml);
        Guid = Guid.NewGuid();
        Get<Label>(Helper.protocol_guid_lbl).Text += " - " + Guid.ToString();
        Get<ImageButton>(Helper.button_photo_1_).Clicked += SelectPhoto;
        Get<ImageButton>(Helper.button_photo_2_).Clicked += SelectPhoto;
        Get<ImageButton>(Helper.button_photo_3_).Clicked += SelectPhoto;
        Get<ImageButton>(Helper.button_photo_schema_).Clicked +=
SelectSchema;
        Get<ImageButton>(Helper.button_damage_car_).Clicked +=
SelectDamageCar;
        Get<ImageButton>(Helper.button_state_reg_mark_).Clicked +=
SelectRegMark;
        Get<Button>(Helper.button_send_).Clicked += AddProtocol;
        try
        {
            Helper.FillCar(car, this);
            Helper.FillDriver(driver, this);
            Helper.FillOwner(owner, this);
        }
        catch (Exception ex)
        {
            this.Background = Colors.Red;
            Get<Entry>("surname_witness_").Text = ex.Message;
        }
    }
}

```

```

    }

    public T Get<T>(string name)
    {
        return this.FindByName<T>(name);
    }

    private async void SelectPhoto(object sender, EventArgs e)
    {
        var photo = await MediaPicker.PickPhotoAsync();

        if (photo == null)
        {
            await DisplayAlert("Error", "Не удалось добавить фото", "OK");
            return;
        }

        var button = sender as ImageButton;
        button.Source = ImageSource.FromFile(photo.FullPath);

        if (Photos.Count == 3)
            Photos.RemoveAt(0);

        Photos.Add(photo);
    }

    private async void SelectDamageCar(object sender, EventArgs e)
    {
        var photo = await MediaPicker.PickPhotoAsync();

        if (photo == null)
        {
            await DisplayAlert("Error", "Не удалось добавить фото", "OK");
            return;
        }

        var button = sender as ImageButton;
        button.Source = ImageSource.FromFile(photo.FullPath);
        Damage_car = photo;
    }

    private async void SelectSchema(object sender, EventArgs e)
    {
        var photo = await MediaPicker.PickPhotoAsync();

        if (photo == null)
        {
            await DisplayAlert("Error", "Не удалось добавить фото", "OK");
            return;
        }

        var button = sender as ImageButton;
        button.Source = ImageSource.FromFile(photo.FullPath);
        Schema = photo;
    }

    private async void SelectRegMark(object sender, EventArgs e)
    {
        var photo = await MediaPicker.PickPhotoAsync();

        if (photo == null)
        {
            await DisplayAlert("Error", "Не удалось добавить фото", "OK");
            return;
        }
    }

```

```

        var button = sender as ImageButton;
        button.Source = ImageSource.FromFile(photo.FullPath);
        Reg_mark = photo;
    }

    private async void AddProtocol(object sender, EventArgs e)
    {
        try
        {
            if (Get<Entry>(Helper.address_rta_).Text == null
                || Get<Picker>(Helper.circumstances_rta_).SelectedIndex == -1
                || Get<Editor>(Helper.explanations_).Text == null
                || Photos.Count != 3 || Schema == null
                || Get<Picker>(Helper.disagreements_).SelectedIndex == -1
                || Get<Entry>(Helper.surname_witness_).Text == null
                || Get<Entry>(Helper.name_witness_).Text == null
                || Get<Entry>(Helper.second_name_witness_).Text == null
                || Get<Entry>(Helper.phone_witness_).Text == null
                || Get<Entry>(Helper.address_witness_).Text == null
                || Get<Picker>(Helper.hit_place_number_).SelectedIndex == -1)
            {
                await DisplayAlert("Error", "Не все поля заполнены", "OK");
                return;
            }

            List<byte[]> photos = new();
            List<Models.Image> images = new();
            byte[]? schema, damage_car = null, reg_mark = null;

            ProtocolService protocol_serv = new();
            WitnessService witness_serv = new();
            ImageService image_serv = new();
            ParticipantCarService participant_car_serv = new();
            ParticipantService participant_serv = new();

            foreach (var photo in Photos)
            {
                var res = await Helper.ToByte(photo);
                if (res != null)
                    photos.Add(res);
            }

            if (photos.Count < 3)
            {
                await DisplayAlert("Error", "Не удалось конвертировать фотографии", "OK");
                return;
            }

            schema = await Helper.ToByte(Schema);

            if (schema == null)
            {
                await DisplayAlert("Error", "Не удалось конвертировать схему", "OK");
                return;
            }

            if (Damage_car != null)
                damage_car = await Helper.ToByte(Damage_car);

            if (Reg_mark != null)
                reg_mark = await Helper.ToByte(Reg_mark);

            var disagreements =
                Get<Picker>(Helper.disagreements_).SelectedIndex == 0

```

```

        ? true : false;

Protocol protocol = new()
{
    Id = Guid,
    Date_Time = Get<DatePicker>(Helper.date_rta_).Date
    + Get<TimePicker>(Helper.time_rta_).Time,
    Address = Get<Entry>(Helper.address_rta_).Text,
    Schema = schema,
    Disagreements = disagreements
};

Witness witness = new()
{
    Surname = Get<Entry>(Helper.surname_witness_).Text,
    Name = Get<Entry>(Helper.name_witness_).Text,
    Second_name = Get<Entry>(Helper.second_name_witness_).Text,
    Phone = Get<Entry>(Helper.phone_witness_).Text,
    Address = Get<Entry>(Helper.address_witness_).Text,
    Protocol_id = protocol.Id
};

foreach (var photo in photos)
{
    images.Add(
        new Models.Image()
        {
            Protocol_id = protocol.Id,
            Content = photo
        });
}

Participant participant = new()
{
    Protocol_id = protocol.Id,
    Driver_id = Driver.Id,
    Circumstances_rta =
Get<Picker>(Helper.circumstances_rta_).SelectedIndex,
    Explanations = Get<Editor>(Helper.explanations_).Text
};

ParticipantCar p_car = new()
{
    Protocol_id = protocol.Id,
    Car_id = Car.Id,
    Damage_place =
Get<Picker>(Helper.hit_place_number_).SelectedIndex,
    Damage_image = damage_car,
    Reg_number_image = reg_mark
};

var protocol_res = await protocol_serv.Add(protocol);

if (protocol_res == null)
    throw new ArgumentNullException("Protocol service return
null");

var witness_res = await witness_serv.Add(witness);

if (witness_res == null)
    throw new ArgumentNullException("Witness service return
null");

foreach (var photo in images)
{
    var image_res = await image_serv.Add(photo);

```

```

        if (image_res == null)
            throw new ArgumentNullException("Image service return
null");
    }

    var participant_res = await participant_serv.Add(participant);
    if (participant_res == null)
        throw new ArgumentNullException("Participant service return
null");

    var car_res = await participant_car_serv.Add(p_car);
    if (car_res == null)
        throw new ArgumentNullException("Participant car service
return null");
    }
    catch (Exception ex)
    {
        await DisplayAlert("Error", ex.Message, "OK");
        return;
    }
    await DisplayAlert("Message", "Успешно добавлено", "OK");
    WaitParticipants();
}

private async void WaitParticipants()
{
    ContentPage page = new ContentPage()
    {
        Content = new StackLayout()
        {
            {
                VerticalOptions = LayoutOptions.Center,
                HorizontalOptions = LayoutOptions.Center,
                Children =
                {
                    new Label() {Text="Ждем второго участника", FontSize
= 25},
                    new ActivityIndicator() { IsRunning = true, Color =
Colors.SeaGreen }
                }
            }
        }
    };
    await Navigation.PushAsync(page);
    ParticipantService participant_serv = new();
    List<Participant>? participants = new();

    while (participants?.Count != 2)
    {
        participants = (await participant_serv.Get(Guid))?.ToList();
        System.Threading.Thread.Sleep(15000);
    }
    await Navigation.PushModalAsync(new AllProtocolPage(All_blank,
Guid));
}
}

```

Листинг Г.28 – Файл RegDriverPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.DriverPages.RegDriverPage"
    Title="Регистрация водителя">
    <ScrollView>
        <StackLayout>

```

```

<Label Text="Логин" FontSize="20"/>
<Entry x:Name="login_" FontSize="20" Margin="0,0,0,15"/>
<Label Text="Пароль" FontSize="20"/>
<Entry x:Name="password_" FontSize="20" Margin="0,0,0,15"
IsPassword="True"/>
<Label Text="Фамилия" FontSize="20"/>
<Entry x:Name="surname_" FontSize="20" Margin="0,0,0,15"/>
<Label Text="Имя" FontSize="20"/>
<Entry x:Name="name_" FontSize="20" Margin="0,0,0,15"/>
<Label Text="Отчество" FontSize="20"/>
<Entry x:Name="second_name_" FontSize="20" Margin="0,0,0,15"/>
<Label Text="Телефон" FontSize="20"/>
<Entry x:Name="phone_" FontSize="20" Margin="0,0,0,15"/>
<Label Text="Дата рождения" FontSize="20"/>
<DatePicker x:Name="birthday_" FontSize="20" Margin="0,0,0,15"/>
<Label Text="Адрес проживания" FontSize="20"/>
<Entry x:Name="address_" FontSize="20" Margin="0,0,0,15"/>
<Label Text="Права" FontSize="20"/>
<Grid Margin="0,0,0,15">
    <Grid.RowDefinitions>
        <RowDefinition Height="50"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="65"/>
        <ColumnDefinition Width="90"/>
        <ColumnDefinition Width="80"/>
        <ColumnDefinition Width="170"/>
    </Grid.ColumnDefinitions>
    <Label Text="Серия" FontSize="20" Grid.Row="0"
Grid.Column="0"/>
    <Entry x:Name="series_lic_" FontSize="20" Grid.Row="0"
Grid.Column="1"/>
    <Label Text="Homep" FontSize="20" Grid.Row="0"
Grid.Column="2"/>
    <Entry x:Name="number_lic_" FontSize="20" Grid.Row="0"
Grid.Column="3"/>
</Grid>
<Label Text="Действительны до" FontSize="20"/>
<DatePicker x:Name="date_lic_" FontSize="20" Margin="0,0,0,20"/>
<Button Text="Зарегистрироваться" FontSize="20" TextColor="White"
    Background="DodgerBlue" Clicked="AddDriver"/>
</StackLayout>
</ScrollView>

</ContentPage>

```

Листинг Г.29 – Файл RegDriverPage.xaml.cs

```

using MobileProtocolApp.Models;
using MobileProtocolApp.Services;
using MobileProtocolApp.HelperCS;

namespace MobileProtocolApp.DriverPages;

public partial class RegDriverPage : ContentPage
{
    AuthPage page_;
    public RegDriverPage(AuthPage page)
    {
        InitializeComponent();
        page_ = page;
    }

    private bool FieldsIsFill()

```

```

    {
        return login_.Text != null && password_.Text != null
            && surname_.Text != null && name_.Text != null &&
second_name_.Text != null
            && phone_.Text != null && address_.Text != null
            && series_lic_.Text != null && number_lic_.Text != null;
    }

private async void AddDriver(object sender, EventArgs e)
{
    if (!FieldsIsFill())
    {
        await DisplayAlert("Error", "Не все поля заполнены", "OK");
        return;
    }

    if (!Helper.IsLogin(login_.Text))
    {
        await DisplayAlert("Error", "Логин не соответствует шаблону
электронной почты", "OK");
        return;
    }

    if (!Helper.IsPassword(password_.Text))
    {
        await DisplayAlert("Error", "Плохой пароль. Пароль должен иметь:"
+
            "\n\tдлину от 8 до 15 символов;" +
            "\n\tанглийские заглавные и прописные буквы" +
            "\n\tспециальные символы" +
            "\n\tцифры", "OK");
        return;
    }

    var user_serv = new UserService();
    var driver_serv = new DriverService();

    User user = new()
    {
        Traffic_laws_id = Helper.kLawsId,
        Login = login_.Text,
        Password = password_.Text
    };

    var response = await user_serv.Add(user);

    if (response == null)
    {
        await DisplayAlert("Error", "Не получилось зарегистрировать
аккаунт", "OK");
        return;
    }

    Driver driver = new()
    {
        User_id = response.Id,
        Surname = surname_.Text,
        Name = name_.Text,
        Second_name = second_name_.Text,
        Phone = phone_.Text,
        Birthday = birthday_.Date,
        Address = address_.Text,
        Driver_lic = series_lic_.Text + number_lic_.Text,
        Lic_date = date_lic_.Date
    };

```

```

var resp = await driver_serv.Add(driver);

if (resp == null)
{
    await DisplayAlert("Error", "Не получилось зарегистрировать
водителя", "OK");
    return;
}

await DisplayAlert("Message", "Пользователь зарегистрирован", "OK");
await Navigation.PopAsync();
page_.ToAuth();
}
}

```

Листинг Г.30 – Файл RTAPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.DriverPages.RTAPage"
    Title="ДТП">
    <VerticalStackLayout>
        <Label
            Text=
"Оформить ДТП без вызова ГАИ
по европротоколу можно при следующих условиях:"
            FontSize="20"/>
        <Label Text="1. Люди не пострадали" FontSize="20" Margin="0,30,0,0"/>
        <Label Text="2. Участвуют только два ТС" FontSize="20"
Margin="0,20,0,0"/>
        <Label Text="3. У обоих водителей есть учетная запись в данном
приложении" FontSize="20" Margin="0,20,0,0"/>
        <Label Text="4. У обоих водителей есть полис ОСАГО" FontSize="20"
Margin="0,20,0,0"/>
        <Label Text="5. Не пострадали третьи лица или имущество"
FontSize="20" Margin="0,20,0,0"/>
        <Label Text="6. Не повреждена инфраструктура, например ограждения,
столбы" FontSize="20" Margin="0,20,0,0"/>
        <Button Text="Оформить ДТП" FontSize="20" TextColor="White"
            Background="DodgerBlue" Margin="0,150,0,0"
Clicked="NextStep"/>
    </VerticalStackLayout>
</ContentPage>

```

Листинг Г.31 – Файл RTAPage.xaml.cs

```

using MobileProtocolApp.Models;

namespace MobileProtocolApp.DriverPages;

[QueryProperty(nameof(Pages), "Pages")]
[QueryProperty(nameof(Cars), "Cars")]
[QueryProperty(nameof(Owners), "Owner")]
[QueryProperty(nameof(Driver), "Driver")]
public partial class RTAPage : ContentPage
{
    List<Models.Page>? pages_;
    List<Car>? cars_;
    List<Owner>? owners_;
    Driver? driver_;
}

```

```

public List<Models.Page>? Pages
{
    get => pages_;
    set
    {
        pages_ = value;
        OnPropertyChanged();
    }
}

public List<Car>? Cars
{
    get => cars_;
    set
    {
        cars_ = value;
        OnPropertyChanged();
    }
}

public List<Owner>? Owners
{
    get => owners_;
    set
    {
        owners_ = value;
        OnPropertyChanged();
    }
}

public Driver? Driver
{
    get => driver_;
    set
    {
        driver_ = value;
        OnPropertyChanged();
    }
}

public RTAPage()
{
    InitializeComponent();
    BindingContext = this;
}

private async void NextStep(object sender, EventArgs e)
{
    if (pages_ == null)
    {
        await DisplayAlert("Message", "NULL", "OK");
        return;
    }
    await Navigation.PushAsync(new ChoiceWriterPage(pages_, cars_,
owners_, driver_));
}
}

```

Листинг Г.32 – Файл SignaturePage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.DriverPages.SignaturePage"
    Title="Подтвердите свои данные">
    <StackLayout VerticalOptions="Center">

```

```

<StackLayout Margin="50,0,0,0">
    <Grid Margin="0,0,0,15">
        <Grid.RowDefinitions>
            <RowDefinition Height="50"/>
            <RowDefinition Height="30"/>
            <RowDefinition Height="50"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="80"/>
            <ColumnDefinition Width="180"/>
            <ColumnDefinition Width="100"/>
            <ColumnDefinition Width="170"/>
        </Grid.ColumnDefinitions>
        <Label Text="Логин" FontSize="20" Grid.Row="0"
Grid.Column="0"/>
        <Entry x:Name="login_" FontSize="20" Grid.Row="0"
Grid.Column="1"/>
        <Label Text="Пароль" FontSize="20" Grid.Row="2"
Grid.Column="0"/>
        <Entry x:Name="password_" FontSize="20" Grid.Row="2"
Grid.Column="1"
            IsPassword="False"/>
        </Grid>
        <Button Text="Подписать и отправить" BackgroundColor="DodgerBlue"
            TextColor="White" FontSize="20" Margin="0,0,50,10"
Clicked="Signature"/>
    </StackLayout>
</ContentPage>

```

Листинг Г.33 – Файл SignaturePage.xaml.cs

```

namespace MobileProtocolApp.DriverPages;

public partial class SignaturePage : ContentPage
{
    public SignaturePage()
    {
        InitializeComponent();
    }

    private async void Signature(object sender, EventArgs e)
    {
        if (login_.Text == null || password_.Text == null)
        {
            await DisplayAlert("Error", "Вы ввели не все данные", "OK");
            return;
        }

        if (AuthPage.Current_user.Login == login_.Text
            && AuthPage.Current_user.Password == password_.Text)
        {
            await DisplayAlert("Message", "Успешно подписано.\nДанные
успешно отправлены", "OK");
            Application.Current?.MainPage.Navigation.PopToRootAsync();
        } else
        {
            await DisplayAlert("Message", "Вы ввели неверные данные", "OK");
        }
    }
}

```

Листинг Г.34 – Файл Helper.cs

```

using MobileProtocolApp.DriverPages;
using MobileProtocolApp.Models;
using System.ComponentModel;
using System.Net.Mail;
using System.Text.RegularExpressions;

namespace MobileProtocolApp.HelperCS
{
    internal class Helper
    {
        public static int kLawsId = 1;
        public const string protocol_guid_lbl = "protocol_guid_lbl";
        public const string protocol_number_ = "protocol_number_";
        public const string date_rta_ = "date_rta_";
        public const string time_rta_ = "time_rta_";
        public const string address_rta_ = "address_rta_";
        public const string circumstances_rta_ = "circumstances_rta_";
        public const string explanations_ = "explanations_";
        public const string button_photo_1_ = "button_photo_1_";
        public const string button_photo_2_ = "button_photo_2_";
        public const string button_photo_3_ = "button_photo_3_";
        public const string button_photo_4_ = "button_photo_4_";
        public const string button_photo_5_ = "button_photo_5_";
        public const string button_photo_6_ = "button_photo_6_";
        public const string button_photo_schema_ = "button_photo_schema_";
        public const string disagreements_ = "disagreements_";
        public const string surname_witness_ = "surname_witness_";
        public const string name_witness_ = "name_witness_";
        public const string second_name_witness_ = "second_name_witness_";
        public const string phone_witness_ = "phone_witness_";
        public const string address_witness_ = "address_witness_";
        public const string surname_driver_ = "surname_driver_";
        public const string name_driver_ = "name_driver_";
        public const string second_name_driver_ = "second_name_driver_";
        public const string phone_driver_ = "phone_driver_";
        public const string birthday_driver_ = "birthday_driver_";
        public const string address_driver_ = "address_driver_";
        public const string series_lic_driver_ = "series_lic_driver_";
        public const string number_lic_driver_ = "number_lic_driver_";
        public const string date_lic_driver_ = "date_lic_driver_";
        public const string car_brand_ = "car_brand_";
        public const string car_model_ = "car_model_";
        public const string state_reg_mark_ = "state_reg_mark_";
        public const string vin_number_ = "vin_number_";
        public const string sts_number_ = "sts_number_";
        public const string car_owner_ = "car_owner_";
        public const string address_owner_ = "address_owner_";
        public const string phone_owner_ = "phone_owner_";
        public const string insurance_company_ = "insurance_company_";
        public const string osago_series_ = "osago_series_";
        public const string osago_number_ = "osago_number_";
        public const string button_car_schema = "button_car_schema";
        public const string hit_place_number_ = "hit_place_number_";
        public const string button_damage_car_ = "button_damage_car_";
        public const string button_state_reg_mark_ =
"button_state_reg_mark_";
        public const string button_send_ = "button_send_";
        public const string button_send_2_ = "button_send_2_";
        public const string button_back_2_ = "button_back_2_";

        public static bool IsLogin(string value)
        {
            if (value == null) return false;

```

```

        if (value.Length > 30) return false;

        try
        {
            MailAddress mail = new(value);
            return true;
        } catch (Exception)
        {
            return false;
        }
    }

    public static bool IsPassword(string value)
    {
        if (value == null) return false;
        Regex regex = new(@"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[^\da-zA-Z]).{8,15}$");
        return regex.IsMatch(value);
    }

    public static void FillCar(Car car, ContentPage page, bool is_second
= false)
    {
        string ps = is_second ? "2_" : ""; // postscript
        if (car == null)
            throw new ArgumentNullException("Car is null");

        page.FindByName<Entry>(car_brand_ + ps).Text = car.Brand;
        page.FindByName<Entry>(car_model_ + ps).Text = car.Model;
        page.FindByName<Entry>(state_reg_mark_ + ps).Text =
car.Reg_number;
        page.FindByName<Entry>(vin_number_ + ps).Text = car.VIN_number;
        page.FindByName<Entry>(sts_number_ + ps).Text =
car.Reg_certificate;
        page.FindByName<Picker>(insurance_company_ + ps).SelectedIndex =
car.Insurer;
        page.FindByName<Entry>(osago_series_ + ps).Text =
car.OSAGO?[..3];
        page.FindByName<Entry>(osago_number_ + ps).Text =
car.OSAGO?[3..];
    }

    public static void FillOwner(Owner owner, ContentPage page, bool
is_second = false)
    {
        string ps = is_second ? "2_" : ""; // postscript
        if (owner == null)
            throw new ArgumentNullException("Owner is null");

        page.FindByName<Entry>(car_owner_ + ps).Text = owner.Surname + '
' + owner.Name + ' '
            + owner.Second_name;
        page.FindByName<Entry>(phone_owner_ + ps).Text = owner.Phone;
        page.FindByName<Entry>(address_owner_ + ps).Text = owner.Address;
    }

    public static void FillDriver(Driver driver, ContentPage page, bool
is_second = false)
    {
        string ps = is_second ? "2_" : ""; // postscript
        if (driver == null)
            throw new ArgumentNullException("Driver is null");
    }

```

```

        page.FindByName<Entry>(surname_driver_ + ps).Text =
driver.Surname;
        page.FindByName<Entry>(name_driver_ + ps).Text = driver.Name;
        page.FindByName<Entry>(second_name_driver_ + ps).Text =
driver.Second_name;
        page.FindByName<Entry>(phone_driver_ + ps).Text = driver.Phone;
        page.FindByName<DatePicker>(birthday_driver_ + ps).Date =
driver.Birthday;
        page.FindByName<Entry>(address_driver_ + ps).Text =
driver.Address;
        page.FindByName<Entry>(series_lic_driver_ + ps).Text =
driver.Driver_lic?[..4];
        page.FindByName<Entry>(number_lic_driver_ + ps).Text =
driver.Driver_lic?[4..];
        page.FindByName<DatePicker>(date_lic_driver_ + ps).Date =
driver.Lic_date;
    }

    public static void FillParticipant(Participant participant,
ContentPage page,
        bool is_second = false)
    {
        string ps = is_second ? "2_" : ""; // postscript
        if (participant == null)
            throw new ArgumentNullException("Participant is null");

        page.FindByName<Picker>(circumstances_rta_ + ps).SelectedIndex =
participant.Circumstances_rta;
        page.FindByName<Editor>(explanations_ + ps).Text =
participant.Explanations;
    }

    public static void FillParticipantCar(ParticipantCar car, ContentPage
page, bool is_second = false)
    {
        string ps = is_second ? "2_" : ""; // postscript
        if (car == null) throw new ArgumentNullException("Participant car
is null");

        page.FindByName<Picker>(hit_place_number_ + ps).SelectedIndex =
car.Damage_place;
        if (car.Damage_image != null)
        {
            page.FindByName<ImageButton>(button_damage_car_ + ps).Source
= ImageSource.FromStream(() => new
MemoryStream(car.Damage_image));
        }

        if (car.Reg_number_image != null)
        {
            page.FindByName<ImageButton>(button_state_reg_mark_ +
ps).Source
= ImageSource.FromStream(() => new
MemoryStream(car.Reg_number_image));
        }
    }

    public static void FillWitness(Witness witness, ContentPage page)
    {
        if (witness == null)
            throw new ArgumentNullException("Witness is null");

        page.FindByName<Entry>(surname_witness_).Text = witness.Surname;
        page.FindByName<Entry>(name_witness_).Text = witness.Name;
    }

```

```

        page.FindByName<Entry>(second_name_witness_).Text =
witness.Second_name;
        page.FindByName<Entry>(phone_witness_).Text = witness.Phone;
        page.FindByName<Entry>(address_witness_).Text = witness.Address;
    }

    public static void FillProtocol(Protocol protocol, ContentPage page)
    {
        if (protocol == null)
            throw new ArgumentNullException("Protocol is null");

        page.FindByName<DatePicker>(date_rta_).Date =
protocol.Date_Time.Date;
        page.FindByName<TimePicker>(time_rta_).Time =
protocol.Date_Time.TimeOfDay;
        page.FindByName<Entry>(address_rta_).Text = protocol.Address;
        page.FindByName<Picker>(disagreements_).SelectedIndex =
protocol.Disagreements ? 0 : 1;
        page.FindByName<ImageButton>(button_photo_schema_).Source
            = ImageSource.FromStream(() => new
MemoryStream(protocol.Schema));
    }

    public static void FillImages(List<Models.Image> images, ContentPage
page)
    {
        if (images == null || images.Count != 6)
            throw new ArgumentNullException("Invalid images");

        foreach (var image in images)
        {
            if (image == null || image.Content == null)
                throw new ArgumentNullException("Null images");
        }

        page.FindByName<ImageButton>(button_photo_1_).Source
            = ImageSource.FromStream(() => new
MemoryStream(images[0].Content));
        page.FindByName<ImageButton>(button_photo_2_).Source
            = ImageSource.FromStream(() => new
MemoryStream(images[1].Content));
        page.FindByName<ImageButton>(button_photo_3_).Source
            = ImageSource.FromStream(() => new
MemoryStream(images[2].Content));
        page.FindByName<ImageButton>(button_photo_4_).Source
            = ImageSource.FromStream(() => new
MemoryStream(images[3].Content));
        page.FindByName<ImageButton>(button_photo_5_).Source
            = ImageSource.FromStream(() => new
MemoryStream(images[4].Content));
        page.FindByName<ImageButton>(button_photo_6_).Source
            = ImageSource.FromStream(() => new
MemoryStream(images[5].Content));
    }

    public async static Task<byte[]?> ToByte(FileResult file)
    {
        try
        {
            byte[] data;
            string local_file_path =
Path.Combine(FileSystem.CacheDirectory, file.FileName);
            using Stream stream = await file.OpenReadAsync();
            using (FileStream local_file_stream =
File.OpenWrite(local_file_path))
            {

```

```

        await stream.CopyToAsync(local_file_stream);
    }
    data = File.ReadAllBytes(local_file_path);
    return data;
}
catch (Exception)
{
    return null;
}
}
}
}

```

Листинг Г.35 – Файл AgentInformPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.InsuranceAgentPages.AgentInformPage"
    Title="Мои данные">

    <VerticalStackLayout>

        <StackLayout>

            <StackLayout>
                <Label Text="Фамилия" FontSize="20"/>
                <Entry x:Name="surname_agent_" FontSize="20"
Margin="0,0,0,15"/>
                <Label Text="Имя" FontSize="20"/>
                <Entry x:Name="name_agent_" FontSize="20" Margin="0,0,0,15"/>
                <Label Text="Отчество" FontSize="20"/>
                <Entry x:Name="second_name_agent_" FontSize="20"
Margin="0,0,0,15"/>
                <Label Text="Телефон" FontSize="20"/>
                <Entry x:Name="phone_agent_" FontSize="20"
Margin="0,0,0,15"/>
                <Label Text="Название компании" FontSize="20"/>
                <Picker x:Name="insurance_company_" FontSize="20">
                    <Picker.Items>
                        <x:String>АльфаСтрахование</x:String>
                        <x:String>Ингосстрах</x:String>
                        <x:String>Росгосстрах</x:String>
                        <x:String>Сораз</x:String>
                        <x:String>Ренессанс</x:String>
                    </Picker.Items>
                </Picker>
            </StackLayout>

        </StackLayout>

    </VerticalStackLayout>

</ContentPage>

```

Листинг Г.36 – Файл AgentInformPage.xaml.cs

```

using MobileProtocolApp.Models;

namespace MobileProtocolApp.InsuranceAgentPages;

```

```

[QueryProperty(nameof(Agent), "Agent")]
public partial class AgentInformPage : ContentPage
{
    InsuranceAgent? agent_;
    public InsuranceAgent? Agent
    {
        get => agent_;
        set
        {
            agent_ = value;
            OnPropertyChanged();
        }
    }
    public AgentInformPage()
    {
        InitializeComponent();
        BindingContext = this;
    }

    private void SetFields()
    {
        surname_agent_.Text = Agent.Surname;
        name_agent_.Text = Agent.Name;
        second_name_agent_.Text = Agent.Second_name;
        phone_agent_.Text = Agent.Phone;
        insurance_company_.SelectedIndex = (int)Agent.Company;
    }

    protected override void OnNavigatedTo(NavigatedToEventArgs args)
    {
        base.OnNavigatedTo(args);

        if (Agent == null)
            return;

        SetFields();
    }
}

```

Листинг Г.37 – Файл AgentShellPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<Shell
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.InsuranceAgentPages.AgentShellPage"
    xmlns:local="clr-namespace:MobileProtocolApp.InsuranceAgentPages"
    Navigated="ShellNavigated">

    <TabBar>

        <ShellContent
            Title="Список протоколов"
            ContentTemplate="{DataTemplate local:ProtocolsListPage}"
            Route="ProtocolsListPage"/>

        <ShellContent
            Title="Мои данные"
            ContentTemplate="{DataTemplate local:AgentInformPage}"
            Route="AgentInformPage"/>

    </TabBar>

```

```
</Shell>
```

Листинг Г.38 – Файл AgentShellPage.xaml.cs

```
using MobileProtocolApp.Models;

namespace MobileProtocolApp.InsuranceAgentPages;

public partial class AgentShellPage : Shell
{
    const string kAgentInform = "//AgentInformPage";
    InsuranceAgent Agent { get; set; }
    public AgentShellPage(InuranceAgent agent)
    {
        Agent = agent;
        InitializeComponent();
    }

    private async void ShellNavigated(object sender, ShellNavigatedEventArgs
e)
    {
        if (e.Current.Location.ToString() == kAgentInform)
        {
            var parameter = new Dictionary<string, object> { { "Agent",
Agent } };
            await Shell.Current.GoToAsync(kAgentInform, parameter);
        }
    }
}
```

Листинг Г.39 – Файл ProtocolsListPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="MobileProtocolApp.InsuranceAgentPages.ProtocolsListPage"
    Title="Список протоколов">
    <ScrollView>

        <StackLayout>

            <ListView x:Name="list_view" ItemsSource="{Binding Protocols}"
ItemTapped="ProtocolTapped">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <TextCell Text="{Binding Id}"
                            TextColor="Honeydew">
                        </TextCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>

            <Button Text="Обновить данные" FontSize="20" TextColor="White"
                Background="DodgerBlue" Clicked="UpdateList"/>
        </StackLayout>

    </ScrollView>
</ContentPage>
```

Листинг Г.40 – Файл ProtocolsListPage.xaml.cs

```

using MobileProtocolApp.Models;
using MobileProtocolApp.Services;
using System.Collections.ObjectModel;
using MobileProtocolApp.HelperCS;

namespace MobileProtocolApp.InsuranceAgentPages;

public partial class ProtocolsListPage : ContentPage
{
    public ObservableCollection<Protocol> Protocols { get; set; }
    Models.Page? All_blank { get; set; }
    public ProtocolsListPage()
    {
        InitializeComponent();
        Protocols = new();
        BindingContext = this;
    }

    private async void UpdateList(object sender, EventArgs e)
    {
        ProtocolService serv = new();
        PageService page_serv = new();
        var pages = await page_serv.Get(1);
        if (pages != null)
            All_blank = pages.FirstOrDefault(x => x.Page_name ==
PageName.kFullProtocol);

        var response = await serv.Get();
        if (response == null)
        {
            await DisplayAlert("Error", "Не удалось получить список
протоколов", "OK");
            return;
        }
        Protocols.Clear();
        foreach (var resp in response)
        {
            Protocols.Add(resp);
        }
    }

    private async void ProtocolTapped(object sender, ItemTappedEventArgs e)
    {
        try
        {
            Protocol? protocol = Protocols[e.ItemIndex];
            if (All_blank == null)
            {
                await DisplayAlert("Error", "Не получилось получить
страницу", "OK");
                return;
            }
            DriverPages.AllProtocolPage page = new(All_blank, protocol.Id);
            page.FindByName<Button>(Helper.button_send_2_).IsVisible =
false;
            page.FindByName<Button>(Helper.button_back_2_).IsVisible =
false;
            await Navigation.PushAsync(page);
        } catch (Exception ex)
        {
            await DisplayAlert("Error", ex.Message, "OK");
            return;
        }
    }
}

```

```

    }
}

```

Листинг Г.41 – Файл RegAgentPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.InsuranceAgentPages.RegAgentPage"
    Title="Регистрация страхового агента">

    <VerticalStackLayout>
        <Grid HorizontalOptions="Center" Margin="0,40,0,0" RowSpacing="50">
            <Grid.RowDefinitions>
                <RowDefinition Height="50"/>
                <RowDefinition Height="50"/>
                <RowDefinition Height="50"/>
                <RowDefinition Height="50"/>
                <RowDefinition Height="50"/>
                <RowDefinition Height="50"/>
                <RowDefinition Height="60"/>
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="115"/>
                <ColumnDefinition Width="200"/>
            </Grid.ColumnDefinitions>
            <Label Text="Логин" FontSize="20" Grid.Row="0" Grid.Column="0"/>
            <Entry x:Name="login_" FontSize="20" Grid.Row="0"
Grid.Column="1"/>
            <Label Text="Пароль" FontSize="20" Grid.Row="1" Grid.Column="0"/>
            <Entry x:Name="password_" FontSize="20" Grid.Row="1"
Grid.Column="1" IsPassword="True"/>
            <Label Text="Фамилия" FontSize="20" Grid.Row="2"
Grid.Column="0"/>
            <Entry x:Name="surname_" FontSize="20" Grid.Row="2"
Grid.Column="1"/>
            <Label Text="Имя" FontSize="20" Grid.Row="3" Grid.Column="0"/>
            <Entry x:Name="name_" FontSize="20" Grid.Row="3"
Grid.Column="1"/>
            <Label Text="Отчество" FontSize="20" Grid.Row="4"
Grid.Column="0"/>
            <Entry x:Name="second_name_" FontSize="20" Grid.Row="4"
Grid.Column="1"/>
            <Label Text="Телефон" FontSize="20" Grid.Row="5"
Grid.Column="0"/>
            <Entry x:Name="phone_" FontSize="20" Grid.Row="5"
Grid.Column="1"/>
            <Label Text="Название компании" FontSize="20" Grid.Row="6"
Grid.Column="0"/>
            <Picker x:Name="insurance_company_" FontSize="20" Grid.Row="6"
Grid.Column="1">
                <Picker.Items>
                    <x:String>АльфаСтрахование</x:String>
                    <x:String>Ингосстрах</x:String>
                    <x:String>Росгосстрах</x:String>
                    <x:String>Согаз</x:String>
                    <x:String>Ренессанс</x:String>
                </Picker.Items>
            </Picker>
        </Grid>
        <Button Text="Зарегистрироваться" TextColor="White"
            BackgroundColor="DodgerBlue" FontSize="20" Margin="0,25,0,0"
            Clicked="AddAgent"/>
    </VerticalStackLayout>

```

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

Листинг Г.42 – Файл RegAgentPage.xaml.cs

```
using MobileProtocolApp.Models;
using MobileProtocolApp.Services;
using MobileProtocolApp.HelperCS;

namespace MobileProtocolApp.InsuranceAgentPages;

public partial class RegAgentPage : ContentPage
{
    AuthPage page_;
    public RegAgentPage(AuthPage page)
    {
        InitializeComponent();
        page_ = page;
    }

    private bool FieldsIsFill()
    {
        return login_.Text != null && password_.Text != null
            && surname_.Text != null && name_.Text != null
            && second_name_.Text != null && phone_.Text != null
            && insurance_company_.SelectedIndex != -1;
    }

    private async void AddAgent(object sender, EventArgs e)
    {
        if (!FieldsIsFill())
        {
            await DisplayAlert("Error", "Не все поля заполнены", "OK");
            return;
        }

        if (!Helper.IsLogin(login_.Text))
        {
            await DisplayAlert("Error", "Логин не соответствует шаблону электронной почты", "OK");
            return;
        }

        if (!Helper.IsPassword(password_.Text))
        {
            await DisplayAlert("Error", "Плохой пароль. Пароль должен иметь:"
+
                "\n\tдлину от 8 до 15 символов;" +
                "\n\tанглийские заглавные и прописные буквы" +
                "\n\tспециальные символы" +
                "\n\tцифры", "OK");
            return;
        }

        var user_serv = new UserService();
        var agent_serv = new InsuranceAgentService();

        User user = new()
        {
            Traffic_laws_id = Helper.kLawsId,
            Login = login_.Text,
            Password = password_.Text
        };
    }
}
```

```

var response = await user_serv.Add(user);

if (response == null)
{
    await DisplayAlert("Error", "Не получилось зарегистрировать
аккаунт", "OK");
    return;
}

InsuranceAgent agent = new()
{
    User_id = response.Id,
    Surname = surname_.Text,
    Name = name_.Text,
    Second_name = second_name_.Text,
    Phone = phone_.Text,
    Company = (InsuranceCompany)insurance_company_.SelectedIndex
};

var resp = await agent_serv.Add(agent);
if (resp == null)
{
    await DisplayAlert("Error", "Не получилось зарегистрировать
агента", "OK");
    return;
}

await DisplayAlert("Message", "Пользователь зарегистрирован",
"OK");

await Navigation.PopAsync();
page_.ToAuth();
}
}

```

Листинг Г.43 – Файл App.xaml

```

<?xml version = "1.0" encoding = "UTF-8" ?>
<Application xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:local="clr-namespace:MobileProtocolApp"
    x:Class="MobileProtocolApp.App">
    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="Resources/Styles/Colors.xaml" />
                <ResourceDictionary Source="Resources/Styles/Styles.xaml" />
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Application.Resources>
</Application>

```

Листинг Г.44 – Файл App.xaml.cs

```

using MobileProtocolApp.AdministatorPages;
using MobileProtocolApp.DriverPages;

namespace MobileProtocolApp
{
    public partial class App : Application
    {
        public App()
        {

```

```

        InitializeComponent();
        MainPage = new AuthPage(this);
    }
}

```

Листинг Г.45 – Файл AuthPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.AuthPage">
    <StackLayout>
        <Frame BackgroundColor="DodgerBlue" Padding="20">
            <Label Text="Мобильное приложение для поддержки процессов
регистрации дорожно-транспортных происшествий"
                HorizontalTextAlignment="Center" FontSize="20"/>
        </Frame>
        <StackLayout Margin="80, 110,0,0">
            <Label Text="Вход в систему" FontSize="25" Margin="0,0,0,30"
FontAttributes="Bold"/>
            <Grid Margin="0,0,0,15">
                <Grid.RowDefinitions>
                    <RowDefinition Height="50"/>
                    <RowDefinition Height="30"/>
                    <RowDefinition Height="50"/>
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="80"/>
                    <ColumnDefinition Width="180"/>
                    <ColumnDefinition Width="100"/>
                    <ColumnDefinition Width="170"/>
                </Grid.ColumnDefinitions>
                <Label Text="Логин" FontSize="20" Grid.Row="0"
Grid.Column="0"/>
                <Entry x:Name="login_" FontSize="20" Grid.Row="0"
Grid.Column="1"/>
                <Label Text="Пароль" FontSize="20" Grid.Row="2"
Grid.Column="0"/>
                <Entry x:Name="password_" FontSize="20" Grid.Row="2"
Grid.Column="1"
                    IsPassword="False"/>
            </Grid>
            <Button Text="Войти" BackgroundColor="DodgerBlue"
                TextColor="White" FontSize="20" Margin="0,0,50,10"
                Clicked="SignIn"/>
            <Button Text="Регистрация" BackgroundColor="DodgerBlue"
                TextColor="White" FontSize="20" Margin="0,0,50,0"
                Clicked="Registration"/>
        </StackLayout>
    </StackLayout>
</ContentPage>

```

Листинг Г.46 – Файл AuthPage.xaml.cs

```

using MobileProtocolApp.Models;
using MobileProtocolApp.Services;
using MobileProtocolApp.DriverPages;
namespace MobileProtocolApp;

public partial class AuthPage : ContentPage

```

```

{
    Application app_;
    int current_set_pages_;
    public static User Current_user;
    public AuthPage(Application app)
    {
        InitializeComponent();
        app_ = app;
        current_set_pages_ = 1;
    }

    public async void SignIn(object sender, EventArgs e)
    {
        UserService service = new();

        if (login_.Text == null || password_.Text == null)
        {
            await DisplayAlert("Error", "Не все поля заполнены", "OK");
            return;
        }

        UserType? type = await service.Get(login_.Text, password_.Text);

        if (type == null)
        {
            await DisplayAlert("Error", "Такого пользователя нет", "OK");
            return;
        }

        var users = await service.Get();
        if (users == null)
        {
            await DisplayAlert("Error", "Список пользователей пуст", "OK");
            return;
        }
        var user = users.ToList().Find(x => x.Id == type.User_id);

        if (user == null)
        {
            await DisplayAlert("Error", "Пользователь не найден", "OK");
            return;
        }
        Current_user = user;

        if (type.Type == Types.kAdmin)
        {
            AdministratorService admin_serv = new();
            Administrator? admin = await admin_serv.Get(type.User_id);
            if (admin == null)
            {
                await DisplayAlert("Error", "admin service return null",
"OK");
                return;
            }
            app_.MainPage = new
MobileProtocolApp.AdministatorPages.AdminMainPage(admin);
        }
        else if (type.Type == Types.kAgent)
        {
            InsuranceAgentService agent_serv = new();
            var agent = await agent_serv.Get(type.User_id);
            if (agent == null)
            {
                await DisplayAlert("Error", "agent service return null",
"OK");
                return;
            }

```

```

    }
    app_.MainPage = new InsuranceAgentPages.AgentShellPage(agent);
}
else if (type.Type == Types.kDriver)
{
    DriverService driver_serv = new();
    TrafficLawsService laws_serv = new();
    CarService car_serv = new();
    PageService page_serv = new();
    OwnerService owner_serv = new();

    TrafficLaws? laws = await laws_serv.Get(user.Traffic_laws_id);
    if (laws == null)
    {
        await DisplayAlert("Error", "Правила не найдены", "OK");
        return;
    }

    Driver? driver = await driver_serv.Get(user.Id);
    if (driver == null)
    {
        await DisplayAlert("Error", "Водители не найдены", "OK");
        return;
    }

    List<Car>? cars = (await car_serv.Get(driver.Id)).ToList();
    if (cars == null)
    {
        cars = new List<Car>();
        await DisplayAlert("Message", "Машины не найдены", "OK");
    }

    List<Owner> owners = new();
    foreach (var car in cars)
    {
        Owner? owner = await owner_serv.Get(car.Owner_id);
        if (owner != null)
            owners.Add(owner);
    }

    List<MobileProtocolApp.Models.Page>? pages = (await
page_serv.Get(current_set_pages)).ToList();
    if (pages == null)
    {
        await DisplayAlert("Error", "Страницы не найдены", "OK");
        return;
    }
    app_.MainPage = new DriverShellPage(laws, driver, cars, owners,
pages);
}
}

public void Registration(object sender, EventArgs e)
{
    app_.MainPage = new NavigationPage(new RegistrationPage(this));
}

public void ToAuth()
{
    app_.MainPage = this;
}
}

```

Листинг Г.47 – Файл MauiProgram.cs

```
using Microsoft.Extensions.Logging;

namespace MobileProtocolApp
{
    public static class MauiProgram
    {
        public static MauiApp CreateMauiApp()
        {
            var builder = MauiApp.CreateBuilder();
            builder
                .UseMauiApp<App>()
                .ConfigureFonts(fonts =>
                {
                    fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
                    fonts.AddFont("OpenSans-Semibold.ttf",
"OpenSansSemibold");
                });

            #if DEBUG
                builder.Logging.AddDebug();
            #endif

            return builder.Build();
        }
    }
}
```

Листинг Г.48 – Файл RegistrationPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MobileProtocolApp.RegistrationPage"
    Title="Регистрация">
    <ContentView>
        <StackLayout HorizontalOptions="Center" VerticalOptions="Center">
            <Label Text="Выберите свою роль в приложении:" FontSize="20"
Margin="0,0,0,20"/>
            <Button Text="Администратор" FontSize="20"
                TextColor="White" Background="DodgerBlue" Clicked="GoToAdmin"
Margin="0,0,0,10"/>
            <Button Text="Водитель ТС*" FontSize="20"
                TextColor="White" Background="DodgerBlue" Clicked="GoToDriver"
Margin="0,0,0,10"/>
            <Button Text="Представитель страховой компании" FontSize="20"
                TextColor="White" Background="DodgerBlue" Clicked="GoToAgent"/>
            <Label Text="*ТС - Транспортное средство" FontSize="10"
                TextColor="Gray" Margin="10"/>
        </StackLayout>
    </ContentView>
</ContentPage>
```

Листинг Г.49 – Файл RegistrationPage.xaml.cs

```
using MobileProtocolApp.AdministratorPages;
```

```

using MobileProtocolApp.DriverPages;
using MobileProtocolApp.InsuranceAgentPages;

namespace MobileProtocolApp;

public partial class RegistrationPage : ContentPage
{
    const string kAdmin = "admin";
    const string kDriver = "driver";
    const string kAgent = "agent";
    AuthPage page_;

    public RegistrationPage(AuthPage page)
    {
        InitializeComponent();
        page_ = page;
    }

    private async void GoToAdmin(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new RegAdminPage(page_));
    }

    private async void GoToDriver(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new RegDriverPage(page_));
    }

    private async void GoToAgent(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new RegAgentPage(page_));
    }
}

```

Листинг Г.50 – Файл AdministratorService.cs

```

using MobileProtocolApp.Models;

namespace MobileProtocolApp.Services
{
    class AdministratorService : HttpService<Administrator>
    {
        public async Task<IEnumerable<Administrator>?> Get()
        {
            return await RequestGet(Constants.kUrlAdmins);
        }

        public async Task<Administrator?> Get(int id)
        {
            return await RequestGet(Constants.kUrlAdmins, id);
        }

        public async Task<Administrator?> Add(Administrator admin)
        {
            return await RequestPost(Constants.kUrlAdmins, admin);
        }
    }
}

```

Листинг Г.51 – Файл BaseService.cs

```

using System.Text;

```

```

using System.Text.Json;

namespace MobileProtocolApp.Services
{
    enum Services
    {
        Administrator,
        Car,
        Driver,
        Image,
        Agent,
        Owner,
        Page,
        Participant,
        ProtocolBlank,
        Protocol,
        TrafficLaws,
        UserBlank,
        User,
        Witness
    }

    class BaseService
    {
        JsonSerializerOptions options_ = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true,
        };

        protected HttpClient GetClient()
        {
            var client = new HttpClient();
            client.DefaultRequestHeaders.Add("Accept", "application/json");
            return client;
        }

        protected T? Deserialize<T>(string result)
        {
            return JsonSerializer.Deserialize<T>(result, options_);
        }

        protected StringContent Serialize<T>(T value)
        {
            return new StringContent(JsonSerializer.Serialize(value),
                Encoding.UTF8, "application/json");
        }

        public static BaseService? GetService(Services services)
        {
            if (services == Services.Administrator)
            {
                return new AdministratorService();
            } else if (services == Services.Car)
            {
                return new CarService();
            } else if (services == Services.Driver)
            {
                return new DriverService();
            } else if (services == Services.Image)
            {
                return new ImageService();
            } else if (services == Services.Agent)
            {
                return new InsuranceAgentService();
            } else if (services == Services.Owner)
            {

```

```

        return new OwnerService();
    } else if (services == Services.Page)
    {
        return new PageService();
    } else if (services == Services.Participant)
    {
        return new ParticipantService();
    } else if (services == Services.ProtocolBlank)
    {
        return new ProtocolBlankService();
    } else if (services == Services.Protocol)
    {
        return new ProtocolService();
    } else if (services == Services.TrafficLaws)
    {
        return new TrafficLawsService();
    } else if (services == Services.UserBlank)
    {
        return new UserBlankService();
    } else if (services == Services.User)
    {
        return new UserService();
    } else if (services == Services.Witness)
    {
        return new WitnessService();
    } else
    {
        return null;
    }
}
}
}

```

Листинг Г.52 – Файл CarService.cs

```

using MobileProtocolApp.Models;

namespace MobileProtocolApp.Services
{
    class CarService : HttpService<Car>
    {
        public async Task<IEnumerable<Car>?> Get()
        {
            return await RequestGet(Constants.kUrlCars);
        }

        public async Task<IEnumerable<Car>?> Get(int id)
        {
            string query = Constants.kUrlCars + id;
            return await RequestGet(query);
        }

        public async Task<Car?> Add(Car car)
        {
            return await RequestPost(Constants.kUrlCars, car);
        }

        public async Task<Car?> Update(Car car)
        {
            return await RequestPut(Constants.kUrlCars, car);
        }
    }
}

```

Листинг Г.53 – Файл Constants.cs

```
namespace MobileProtocolApp.Services
{
    static class Constants
    {
        private const string kIPAddress = "192.168.1.5";
        public const string kUrlAdmins =
$"http://{kIPAddress}:5232/api/administrators/";
        public const string kUrlCars = $"http://{kIPAddress}:5232/api/cars/";
        public const string kUrlDrivers =
$"http://{kIPAddress}:5232/api/drivers/";
        public const string kUrlImages =
$"http://{kIPAddress}:5232/api/images/";
        public const string kUrlAgents =
$"http://{kIPAddress}:5232/api/insuranceagents/";
        public const string kUrlOwners =
$"http://{kIPAddress}:5232/api/owners/";
        public const string kUrlPages =
$"http://{kIPAddress}:5232/api/page/";
        public const string kUrlParticipants =
$"http://{kIPAddress}:5232/api/participants/";
        public const string kUrlCarParticipants =
$"http://{kIPAddress}:5232/api/participantcars/";
        public const string kUrlProtocols =
$"http://{kIPAddress}:5232/api/protocols/";
        public const string kUrlProtocolBlanks =
$"http://{kIPAddress}:5232/api/protocolblanks/";
        public const string kUrlTrafficLaws =
$"http://{kIPAddress}:5232/api/trafficlaws/";
        public const string kUrlUsers =
$"http://{kIPAddress}:5232/api/users/";
        public const string kUrlUserBlanks =
$"http://{kIPAddress}:5232/api/userblanks/";
        public const string kUrlWitnesses =
$"http://{kIPAddress}:5232/api/witness/";
    }
}
```

Листинг Г.54 – Файл DriverService.cs

```
using MobileProtocolApp.Models;

namespace MobileProtocolApp.Services
{
    class DriverService : HttpService<Driver>
    {
        public async Task<IEnumerable<Driver>??> Get()
        {
            return await RequestGet(Constants.kUrlDrivers);
        }

        public async Task<Driver?> Get(int id)
        {
            return await RequestGet(Constants.kUrlDrivers, id);
        }

        public async Task<Driver?> Add(Driver driver)
        {
            return await RequestPost(Constants.kUrlDrivers, driver);
        }

        public async Task<Driver?> Update(Driver driver)
```

```

        {
            return await RequestPut(Constants.kUrlDrivers, driver);
        }
    }
}

```

Листинг Г.55 – Файл HttpService.cs

```

using System.Net;

namespace MobileProtocolApp.Services
{
    class HttpService<T> : BaseService
    {
        protected async Task<IEnumerable<T>?> RequestGet(string kUrl)
        {
            var client = GetClient();
            var res = await client.GetAsync(kUrl);
            if (res.IsSuccessStatusCode)
            {
                return Deserialize<IEnumerable<T>>(await
res.Content.ReadAsStringAsync());
            } else
            {
                return null;
            }
        }

        protected async Task<T?> RequestGet(string kUrl, int id)
        {
            var client = GetClient();
            var res = await client.GetAsync(kUrl + id);
            if (res.IsSuccessStatusCode)
            {
                return Deserialize<T>(await res.Content.ReadAsStringAsync());
            } else
            {
                return default;
            }
        }

        protected async Task<T?> RequestGet(string kUrl, Guid id)
        {
            var client = GetClient();
            var res = await client.GetAsync(kUrl + id);
            if (res.IsSuccessStatusCode)
            {
                return Deserialize<T>(await res.Content.ReadAsStringAsync());
            }
            else
            {
                return default;
            }
        }

        protected async Task<T?> RequestPost(string kUrl, T value)
        {
            var client = GetClient();
            var res = await client.PostAsync(kUrl, Serialize(value));

            if (res.StatusCode != HttpStatusCode.OK)
                return default;

            return Deserialize<T>(await res.Content.ReadAsStringAsync());
        }
    }
}

```

```

    }

    protected async Task<T?> RequestPut(string kUrl, T value)
    {
        var client = GetClient();
        var res = await client.PutAsync(kUrl, Serialize(value));

        if (res.StatusCode != HttpStatusCode.OK)
            return default;

        return Deserialize<T>(await res.Content.ReadAsStringAsync());
    }

    protected async Task<T?> RequestDelete(string kUrl, int id)
    {
        var client = GetClient();
        var res = await client.DeleteAsync(kUrl + id);

        if (res.StatusCode != HttpStatusCode.OK)
            return default;

        return Deserialize<T>(await res.Content.ReadAsStringAsync());
    }
}

```

Листинг Г.56 – Файл ImageService.cs

```

namespace MobileProtocolApp.Services
{
    class ImageService : HttpService<MobileProtocolApp.Models.Image>
    {
        public async Task<IEnumerable<MobileProtocolApp.Models.Image>?> Get()
        {
            return await RequestGet(Constants.kUrlImages);
        }

        public async Task<IEnumerable<MobileProtocolApp.Models.Image>?>
        Get(Guid id)
        {
            return await RequestGet(Constants.kUrlImages + id);
        }

        public async Task<MobileProtocolApp.Models.Image?>
        Add(MobileProtocolApp.Models.Image image)
        {
            return await RequestPost(Constants.kUrlImages, image);
        }

        public async Task<MobileProtocolApp.Models.Image?>
        Update(MobileProtocolApp.Models.Image image)
        {
            return await RequestPut(Constants.kUrlImages, image);
        }
    }
}

```

Листинг Г.57 – Файл InsuranceAgentService.cs

```

using MobileProtocolApp.Models;

namespace MobileProtocolApp.Services

```

```

{
    class InsuranceAgentService : HttpService<InsuranceAgent>
    {
        public async Task<IEnumerable<InsuranceAgent>?> Get()
        {
            return await RequestGet(Constants.kUrlAgents);
        }

        public async Task<InsuranceAgent?> Get(int id)
        {
            return await RequestGet(Constants.kUrlAgents, id);
        }

        public async Task<InsuranceAgent?> Add(InsuranceAgent agent)
        {
            return await RequestPost(Constants.kUrlAgents, agent);
        }
    }
}

```

Листинг Г.58 – Файл OwnerService.cs

```

using MobileProtocolApp.Models;

namespace MobileProtocolApp.Services
{
    class OwnerService : HttpService<Owner>
    {
        public async Task<IEnumerable<Owner>?> Get()
        {
            return await RequestGet(Constants.kUrlOwners);
        }

        public async Task<Owner?> Get(int id)
        {
            return await RequestGet(Constants.kUrlOwners, id);
        }

        public async Task<Owner?> Add(Owner owner)
        {
            return await RequestPost(Constants.kUrlOwners, owner);
        }

        public async Task<Owner?> Update(Owner owner)
        {
            return await RequestPut(Constants.kUrlOwners, owner);
        }
    }
}

```

Листинг Г.59 – Файл PageService.cs

```

namespace MobileProtocolApp.Services
{
    class PageService : HttpService<MobileProtocolApp.Models.Page>
    {
        public async Task<IEnumerable<MobileProtocolApp.Models.Page>?> Get()
        {
            return await RequestGet(Constants.kUrlPages);
        }
    }
}

```

```

        public async Task<IEnumerable<MobileProtocolApp.Models.Page?>>
Get(int id)
    {
        var query = Constants.kUrlPages + id;
        return await RequestGet(query);
    }

        public async Task<MobileProtocolApp.Models.Page?>
Add(MobileProtocolApp.Models.Page page)
    {
        return await RequestPost(Constants.kUrlPages, page);
    }

        public async Task<MobileProtocolApp.Models.Page?>
Update(MobileProtocolApp.Models.Page page)
    {
        return await RequestPut(Constants.kUrlPages, page);
    }

        public async Task<MobileProtocolApp.Models.Page?> Delete(int id)
    {
        return await RequestDelete(Constants.kUrlPages, id);
    }
    }
}

```

Листинг Г.60 – Файл ParticipantCarService.cs

```

using MobileProtocolApp.Models;

namespace MobileProtocolApp.Services
{
    internal class ParticipantCarService : HttpService<ParticipantCar>
    {
        public async Task<IEnumerable<ParticipantCar?>> Get(Guid id)
        {
            string query = Constants.kUrlCarParticipants + id.ToString();
            return await RequestGet(query);
        }

        public async Task<ParticipantCar?> Add(ParticipantCar car)
        {
            return await RequestPost(Constants.kUrlCarParticipants, car);
        }
    }
}

```

Листинг Г.61 – Файл ParticipantService.cs

```

using MobileProtocolApp.Models;

namespace MobileProtocolApp.Services
{
    class ParticipantService : HttpService<Participant>
    {
        public async Task<IEnumerable<Participant?>> Get()
        {
            return await RequestGet(Constants.kUrlParticipants);
        }

        public async Task<IEnumerable<Participant?>> Get(Guid guid)
        {

```

```

        return await RequestGet(Constants.kUrlParticipants + guid);
    }

    public async Task<Participant?> Add(Participant participant)
    {
        return await RequestPost(Constants.kUrlParticipants,
participant);
    }
}

```

Листинг Г.62 – Файл ProtocolBlankService.cs

```

using MobileProtocolApp.Models;

namespace MobileProtocolApp.Services
{
    class ProtocolBlankService : HttpService<ProtocolBlank>
    {
        public async Task<IEnumerable<ProtocolBlank?>> Get()
        {
            return await RequestGet(Constants.kUrlProtocolBlanks);
        }

        public async Task<ProtocolBlank?> Get(int id)
        {
            return await RequestGet(Constants.kUrlProtocolBlanks, id);
        }

        public async Task<ProtocolBlank?> Add(ProtocolBlank blank)
        {
            return await RequestPost(Constants.kUrlProtocolBlanks, blank);
        }
    }
}

```

Листинг Г.63 – Файл ProtocolService.cs

```

using MobileProtocolApp.Models;
using System.Net;

namespace MobileProtocolApp.Services
{
    class ProtocolService : BaseService
    {
        public async Task<IEnumerable<Protocol?>> Get()
        {
            var client = GetClient();
            var res = await client.GetAsync(Constants.kUrlProtocols);
            if (res.IsSuccessStatusCode)
            {
                return Deserialize<IEnumerable<Protocol>>(await
res.Content.ReadAsStringAsync());
            } else
            {
                return null;
            }
        }

        public async Task<Protocol?> Get(Guid guid)
        {
            var client = GetClient();

```

```

        var res = await client.GetAsync(Constants.kUrlProtocols + guid);
        if (res.IsSuccessStatusCode)
        {
            return Deserialize<Protocol>(await
res.Content.ReadAsStringAsync());
        } else
        {
            return null;
        }
    }

    public async Task<Protocol?> Add(Protocol protocol)
    {
        var client = GetClient();
        var res = await client.PostAsync(Constants.kUrlProtocols,
Serialize(protocol));

        if (res.StatusCode != HttpStatusCode.OK)
            return null;

        return Deserialize<Protocol>(await
res.Content.ReadAsStringAsync());
    }

    public async Task<Protocol?> Update(Protocol protocol)
    {
        var client = GetClient();
        var res = await client.PutAsync(Constants.kUrlProtocols,
Serialize(protocol));

        if (res.StatusCode != HttpStatusCode.OK)
            return null;

        return Deserialize<Protocol>(await
res.Content.ReadAsStringAsync());
    }

    public async Task<Protocol?> Delete(Guid guid)
    {
        var client = GetClient();
        var res = await client.DeleteAsync(Constants.kUrlProtocols +
guid);

        if (res.StatusCode != HttpStatusCode.OK)
            return null;

        return Deserialize<Protocol>(await
res.Content.ReadAsStringAsync());
    }
}

```

Листинг Г.64 – Файл TrafficLawsService.cs

```

using MobileProtocolApp.Models;

namespace MobileProtocolApp.Services
{
    class TrafficLawsService : HttpService<TrafficLaws>
    {
        public async Task<IEnumerable<TrafficLaws>?> Get()
        {
            return await RequestGet(Constants.kUrlTrafficLaws);
        }
    }
}

```

```

public async Task<TrafficLaws?> Get(int id)
{
    return await RequestGet(Constants.kUrlTrafficLaws, id);
}

public async Task<TrafficLaws?> Add(TrafficLaws laws)
{
    return await RequestPost(Constants.kUrlTrafficLaws, laws);
}

public async Task<TrafficLaws?> Update(TrafficLaws laws)
{
    return await RequestPut(Constants.kUrlTrafficLaws, laws);
}

public async Task<TrafficLaws?> Delete(int id)
{
    return await RequestDelete(Constants.kUrlTrafficLaws, id);
}
}

```

Листинг Г.65 – Файл UserBlankService.cs

```

using MobileProtocolApp.Models;

namespace MobileProtocolApp.Services
{
    class UserBlankService : HttpService<UserBlank>
    {
        public async Task<UserBlank?> Get(int id)
        {
            return await RequestGet(Constants.kUrlUserBlanks, id);
        }

        public async Task<UserBlank?> Add(UserBlank blank)
        {
            return await RequestPost(Constants.kUrlUserBlanks, blank);
        }
    }
}

```

Листинг Г.66 – Файл UserService.cs

```

using MobileProtocolApp.Models;

namespace MobileProtocolApp.Services
{
    class UserService : HttpService<User>
    {
        public async Task<IEnumerable<User>?> Get()
        {
            return await RequestGet(Constants.kUrlUsers);
        }

        public async Task<UserType?> Get(string login, string password)
        {
            var client = GetClient();
            string query = Constants.kUrlUsers
                + $"{login}/{password}";
            var res = await client.GetAsync(query);
        }
    }
}

```

```

        if (res.IsSuccessStatusCode)
        {
            return Deserialize<UserType>(await
res.Content.ReadAsStringAsync());
        } else
        {
            return null;
        }
    }

    public async Task<User?> Add(User user)
    {
        return await RequestPost(Constants.kUrlUsers, user);
    }
}

```

Листинг Г.67 – Файл WitnessService.cs

```

using MobileProtocolApp.Models;

namespace MobileProtocolApp.Services
{
    class WitnessService : HttpService<Witness>
    {
        public async Task<IEnumerable<Witness>?> Get()
        {
            return await RequestGet(Constants.kUrlWitnesses);
        }

        public async Task<Witness?> Get(Guid id)
        {
            return await RequestGet(Constants.kUrlWitnesses, id);
        }

        public async Task<Witness?> Add(Witness witness)
        {
            return await RequestPost(Constants.kUrlWitnesses, witness);
        }
    }
}

```

Приложение Д
(рекомендуемое)
Результаты тестирования серверной части

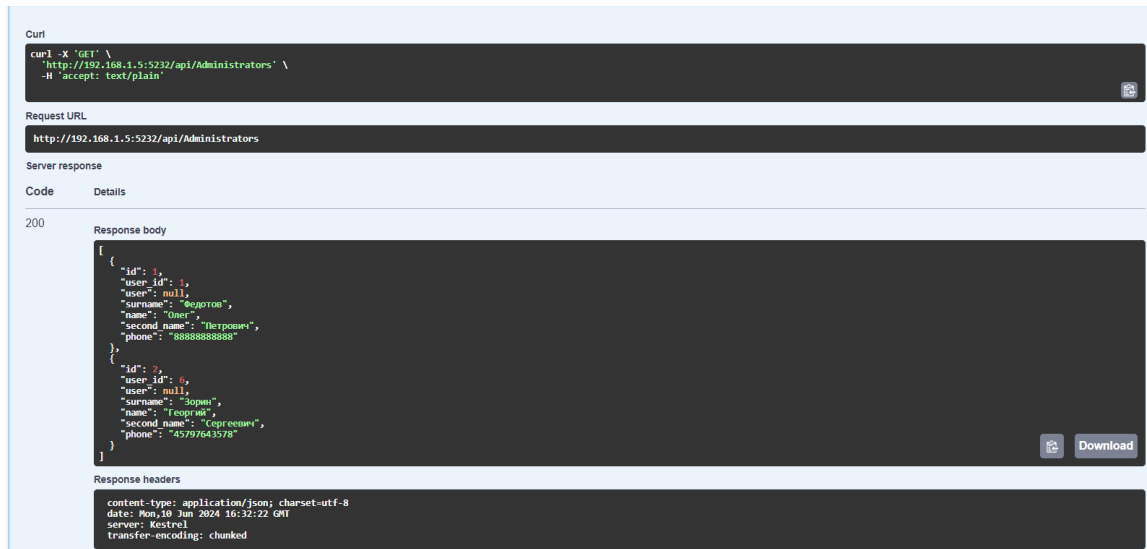


Рисунок Д.1 – Получение списка всех администраторов

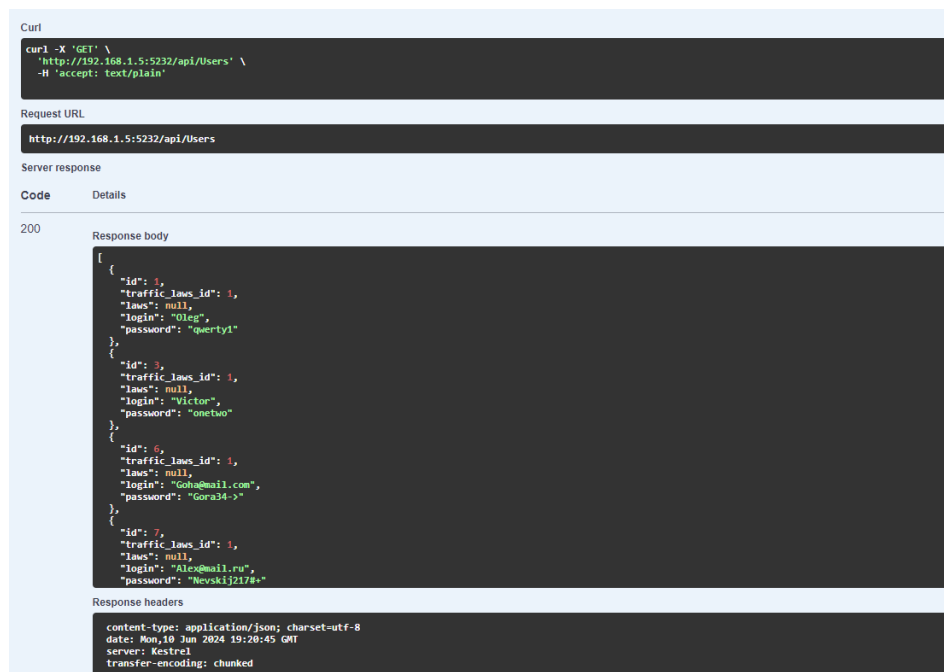


Рисунок Д.2 – Получение списка всех пользователей

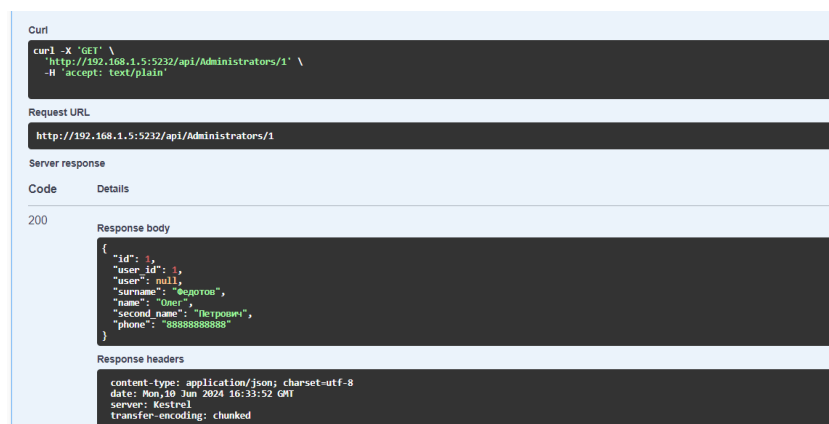


Рисунок Д.3 – Получение конкретного администратора



Рисунок Д.4 – Получение конкретного собственника авто



Рисунок Д.5 – Обновление данных собственника авто

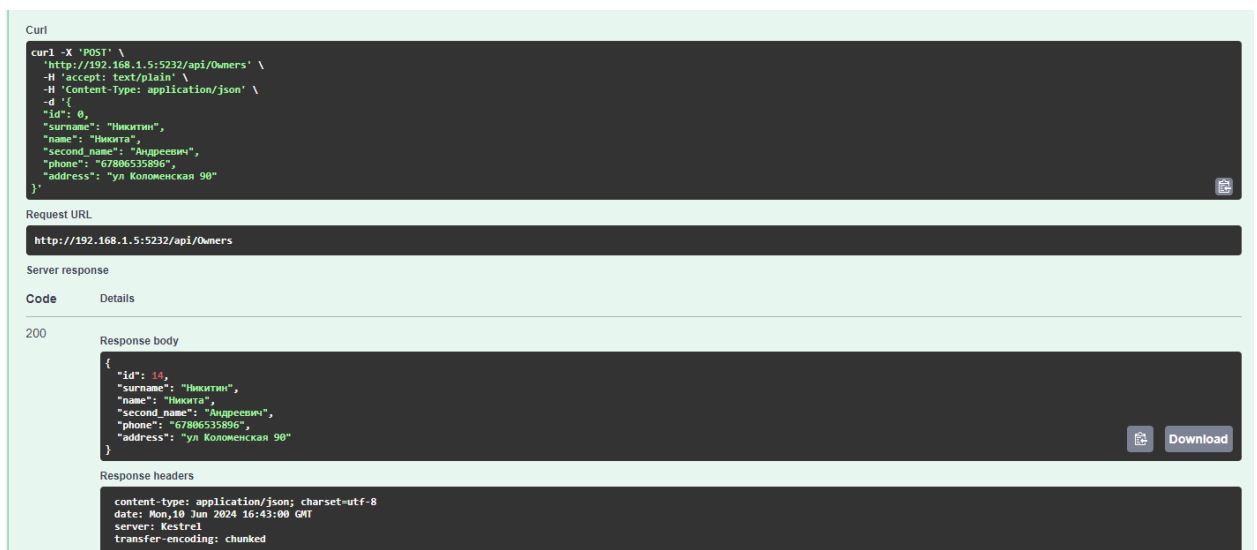


Рисунок Д.6 – Добавление нового собственника авто



Рисунок Д.7 – Получение списка водителей



Рисунок Д.8 – Получение одного водителя

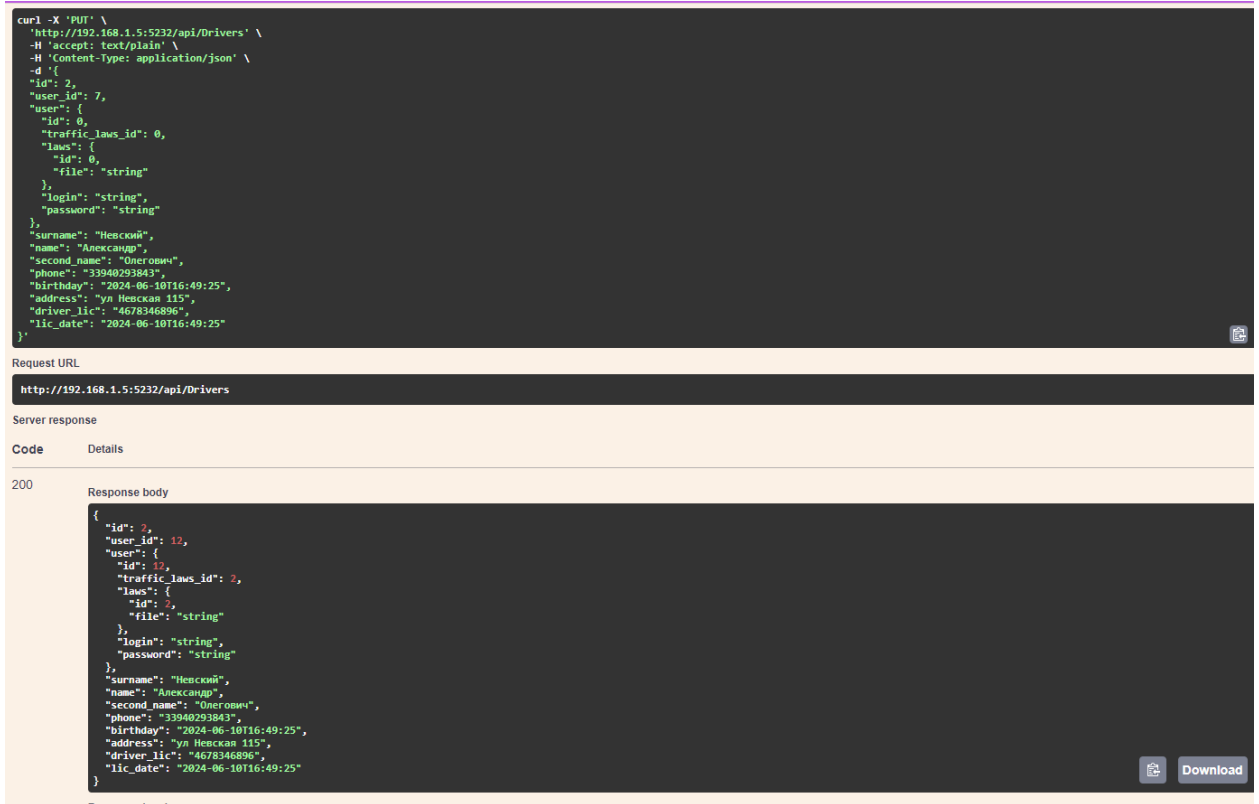


Рисунок Д.9 – Обновление данных о водителе

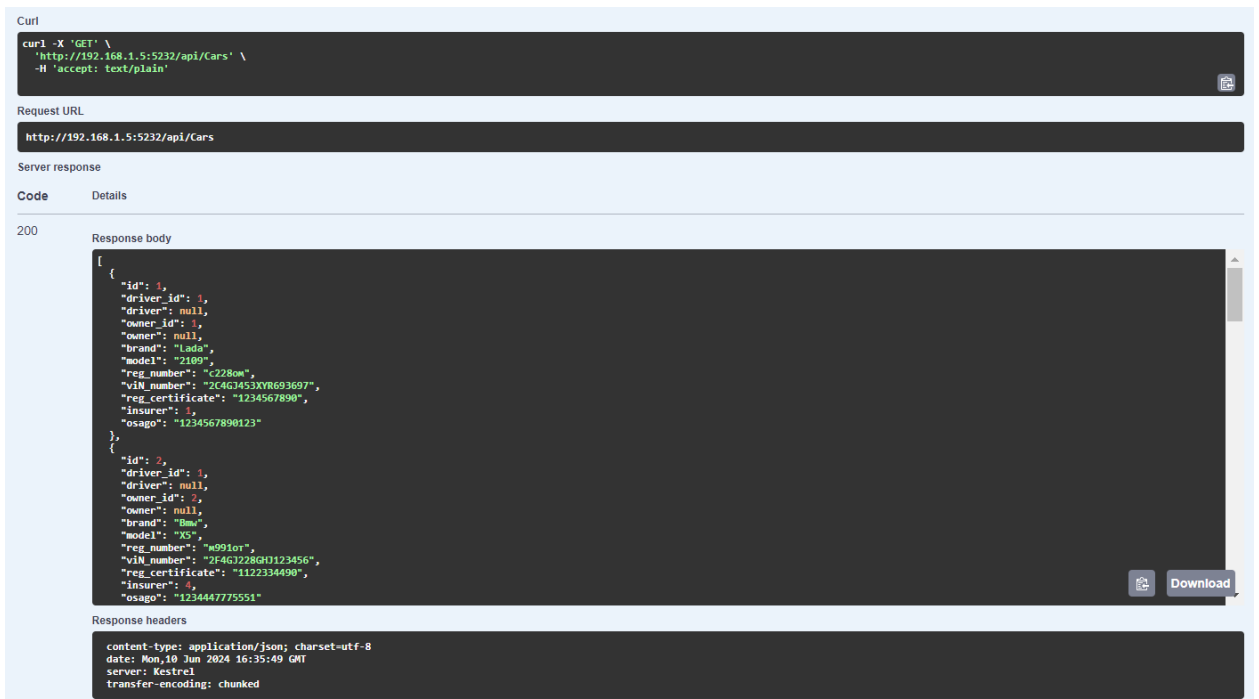


Рисунок Д.10 – Получение списка машин

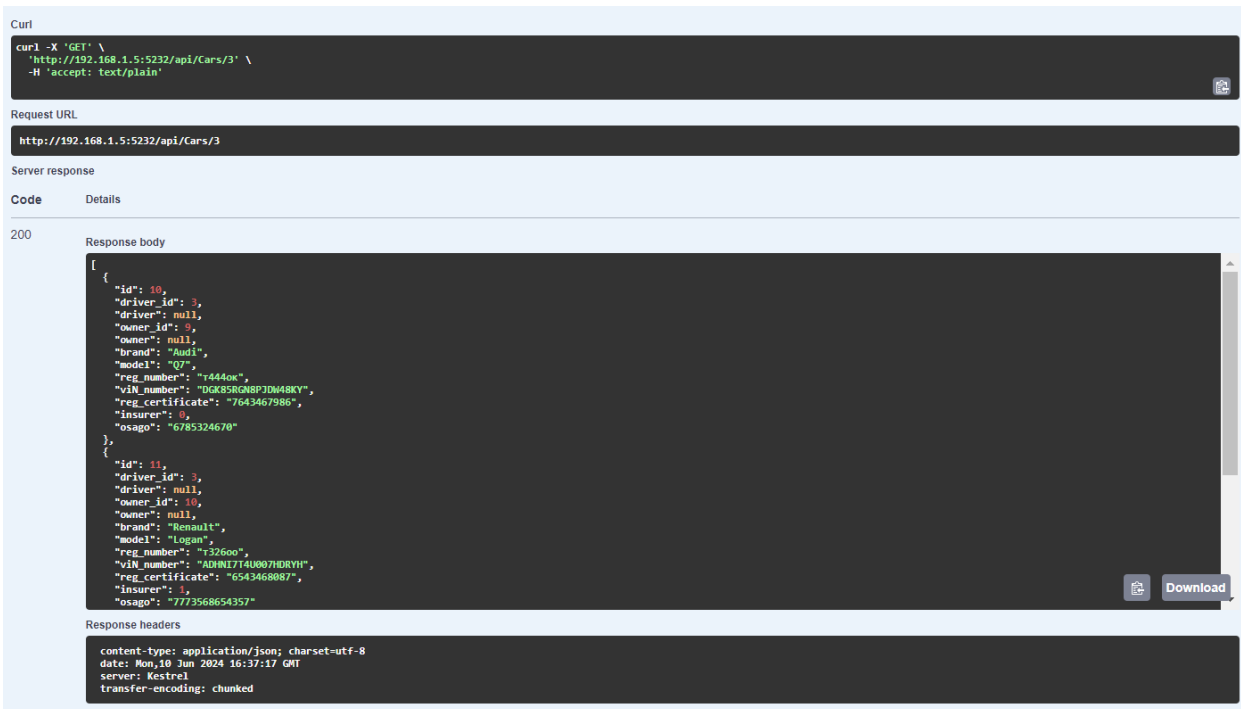


Рисунок Д.11 – Получение списка машин, принадлежащих конкретному водителю



Рисунок Д.12 – Получение списка правил ПДД

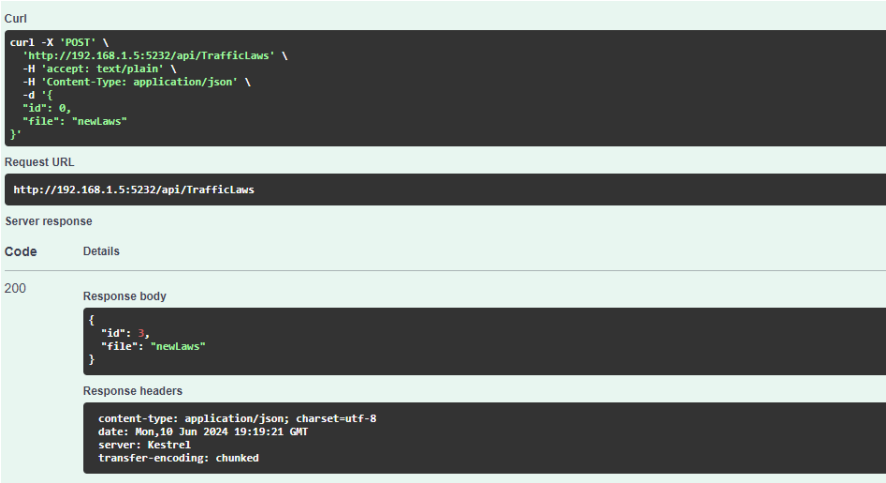


Рисунок Д.13 – Добавление новых правил ПДД



Рисунок Д.14 – Удаление правил ПДД



Рисунок Д.15 – Добавление нового набора бланков

Name	Description
login * required string (path)	Victor
password * required string (path)	onetwo

Execute **Clear**

Responses

Curl

```
curl -X 'GET' \
'http://192.168.1.5:5232/api/Users/Victor/onetwo' \
-H 'accept: text/plain'
```

Request URL

```
http://192.168.1.5:5232/api/Users/Victor/onetwo
```

Server response

Code **Details**

200

Response body

```
{
  "type": 1,
  "user_id": 3
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Mon, 10 Jun 2024 19:21:44 GMT
server: Kestrel
transfer-encoding: chunked
```

Рисунок Д.16 – Авторизация пользователя

Curl

```
curl -X 'GET' \
'http://192.168.1.5:5232/api/Protocols' \
-H 'accept: text/plain'
```

Request URL

```
http://192.168.1.5:5232/api/Protocols
```

Server response

Code **Details**

200

Response body

```
{
  "id": "c23a51d6-5a84-4886-af70-57a2094a9822",
  "date_time": "2024-06-08T12:20:00",
  "address": "192.168.1.5",
  "schema": "192.168.1.5:5232/api/Protocols"
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Mon, 10 Jun 2024 17:25:35 GMT
server: Kestrel
transfer-encoding: chunked
```

Рисунок Д.17 – Получение списка европротоколов

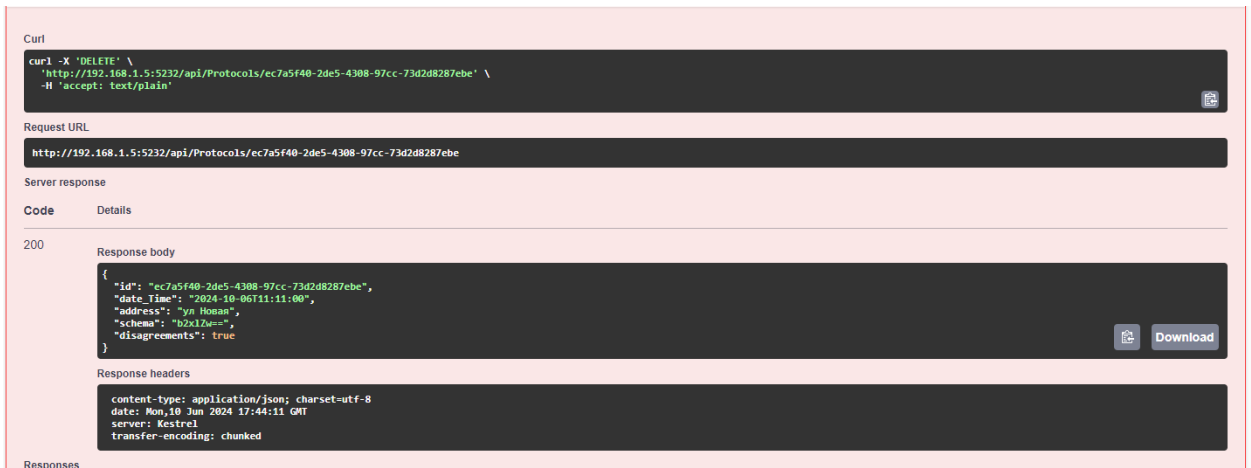


Рисунок Д.18 – Удаление европротокола

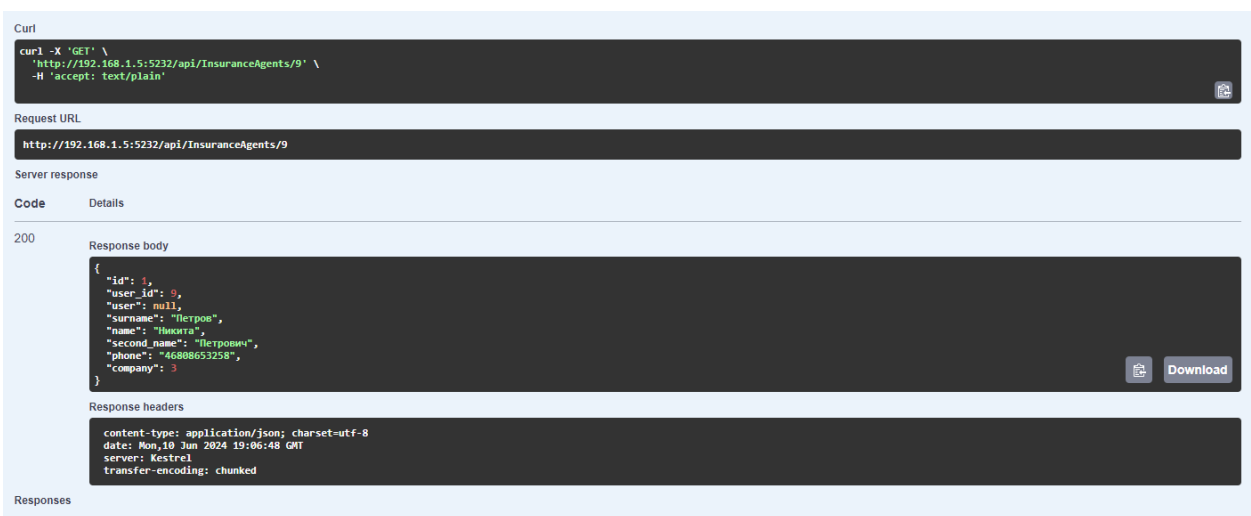


Рисунок Д.19 – Получение конкретного представителя страховой компании

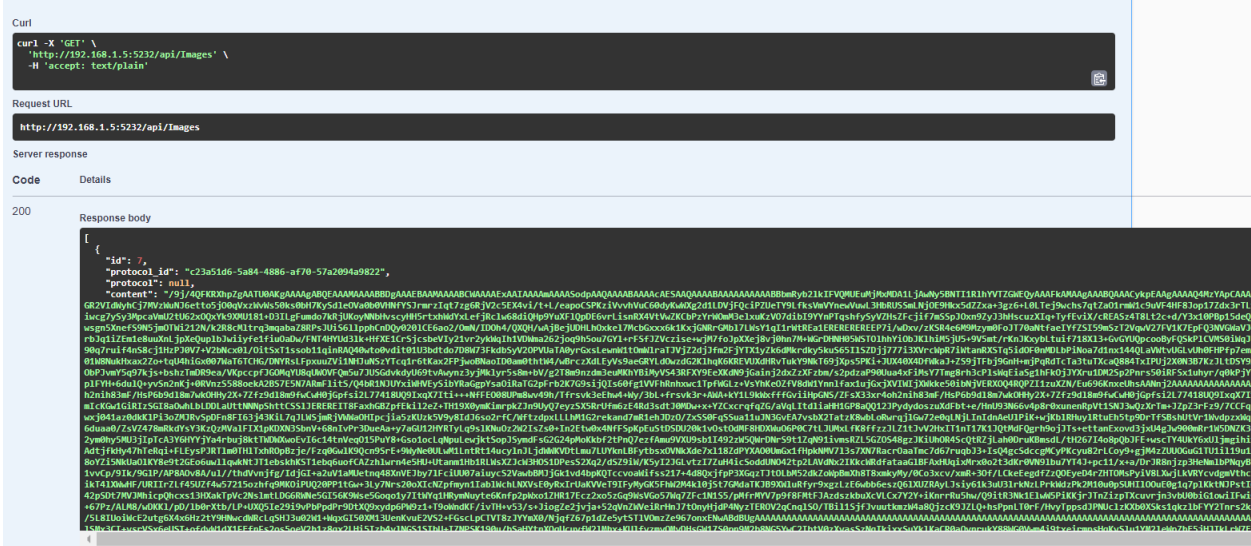


Рисунок Д.20 – Получение списка всех фотографий с места ДТП

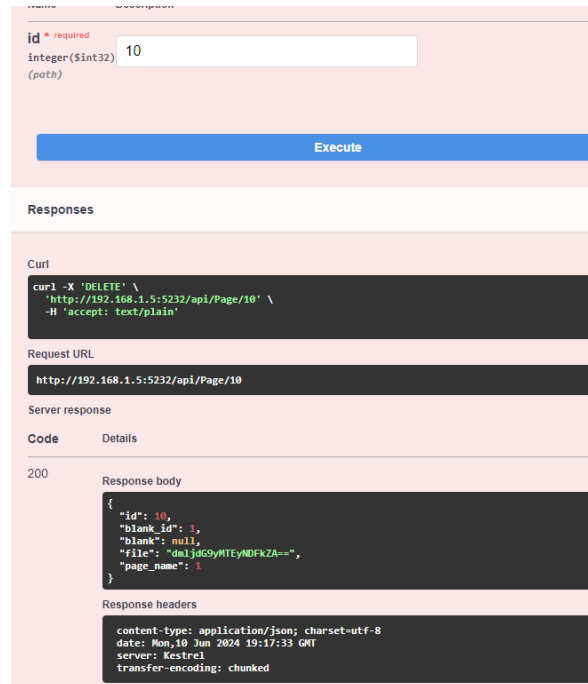


Рисунок Д.21 – Удаление бланка европротокола

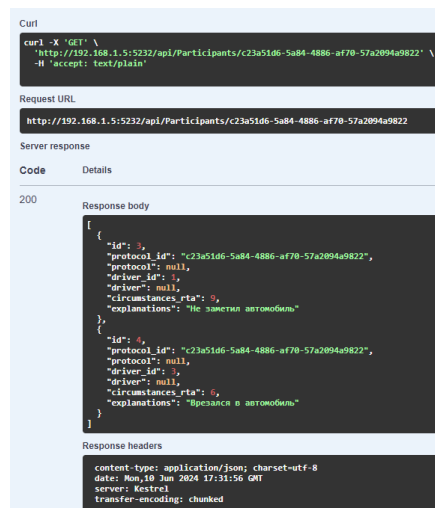


Рисунок Д.22 – Получение конкретного участника ДТП

Приложение Е
(рекомендуемое)
Результаты тестирования мобильного приложения

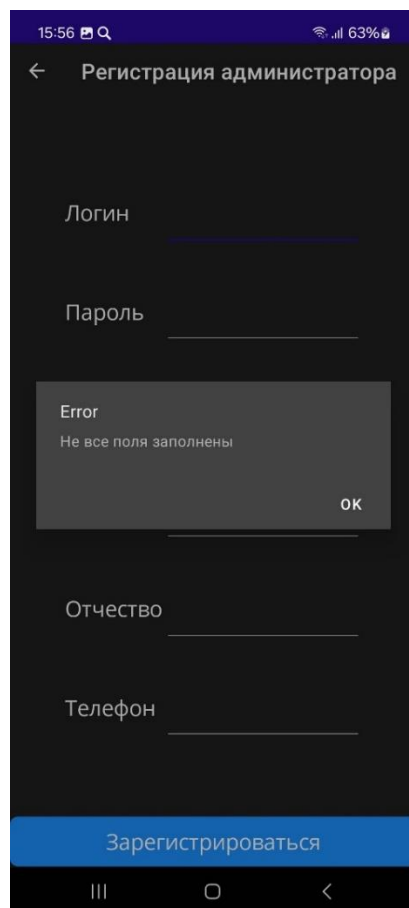


Рисунок Е.1 – Регистрация без введенной информации

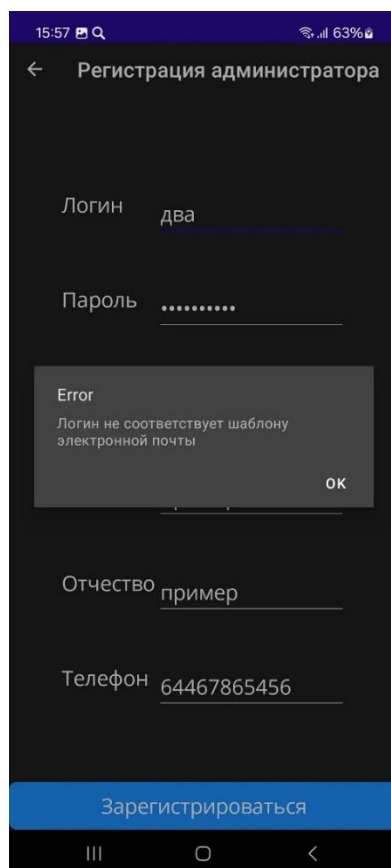


Рисунок Е.2 – Регистрация с недопустимым логином

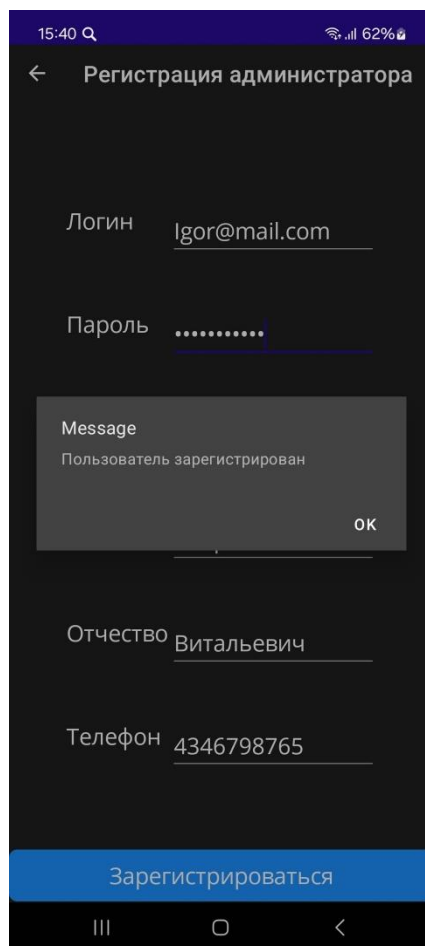


Рисунок Е.3 – Регистрация администратора

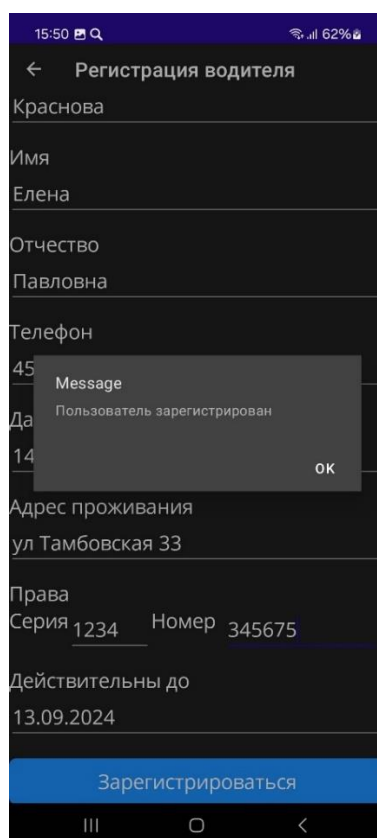


Рисунок Е.4 – Регистрация водителя

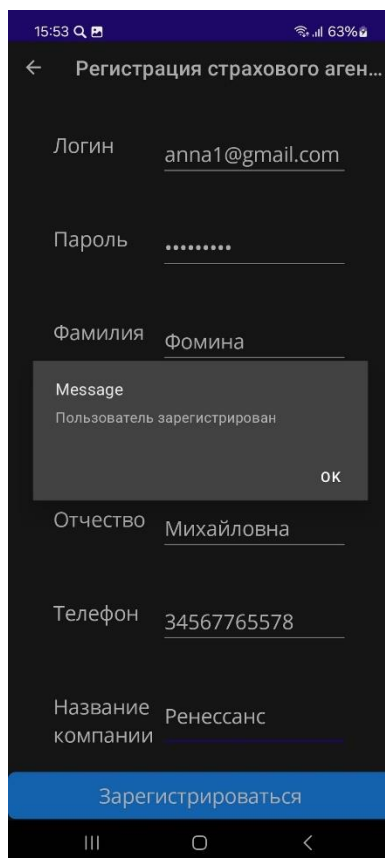


Рисунок Е.5 – Регистрация представителя страховой компании

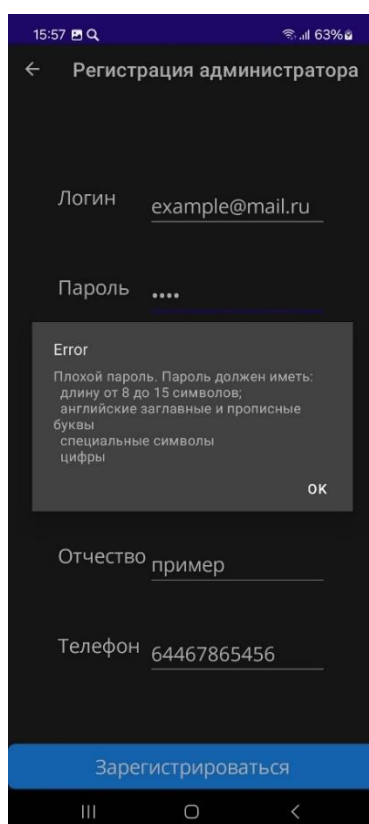


Рисунок Е.6 – Регистрация с недопустимым паролем

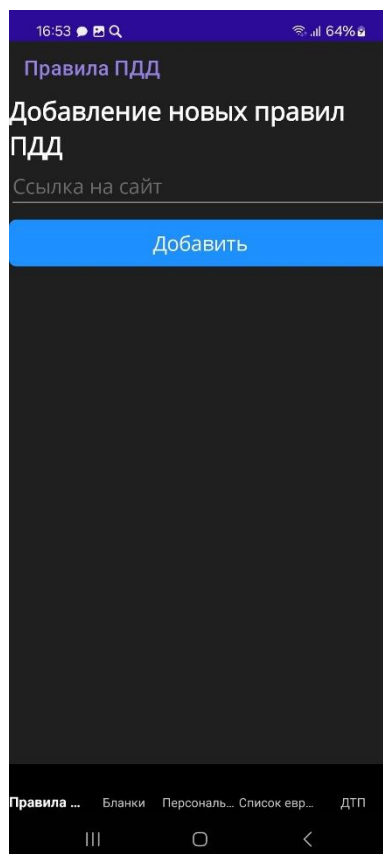


Рисунок Е.7 – Войти в систему как администратор

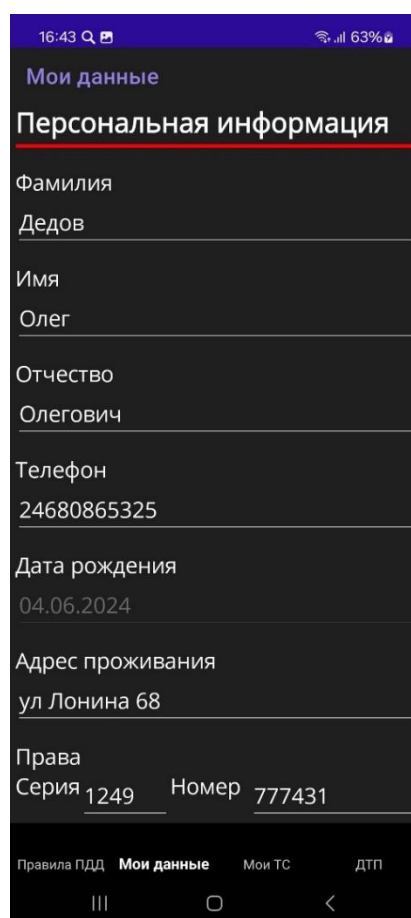


Рисунок Е.8 – Войти в систему как водитель

16:48 🔍 63%

Мои данные

Фамилия
Петров

Имя
Никита

Отчество
Петрович

Телефон
46808653258

Название компании
Согаз

Список протоколов Мои данные

Рисунок Е.9 – Войти в систему как представитель страховой компании

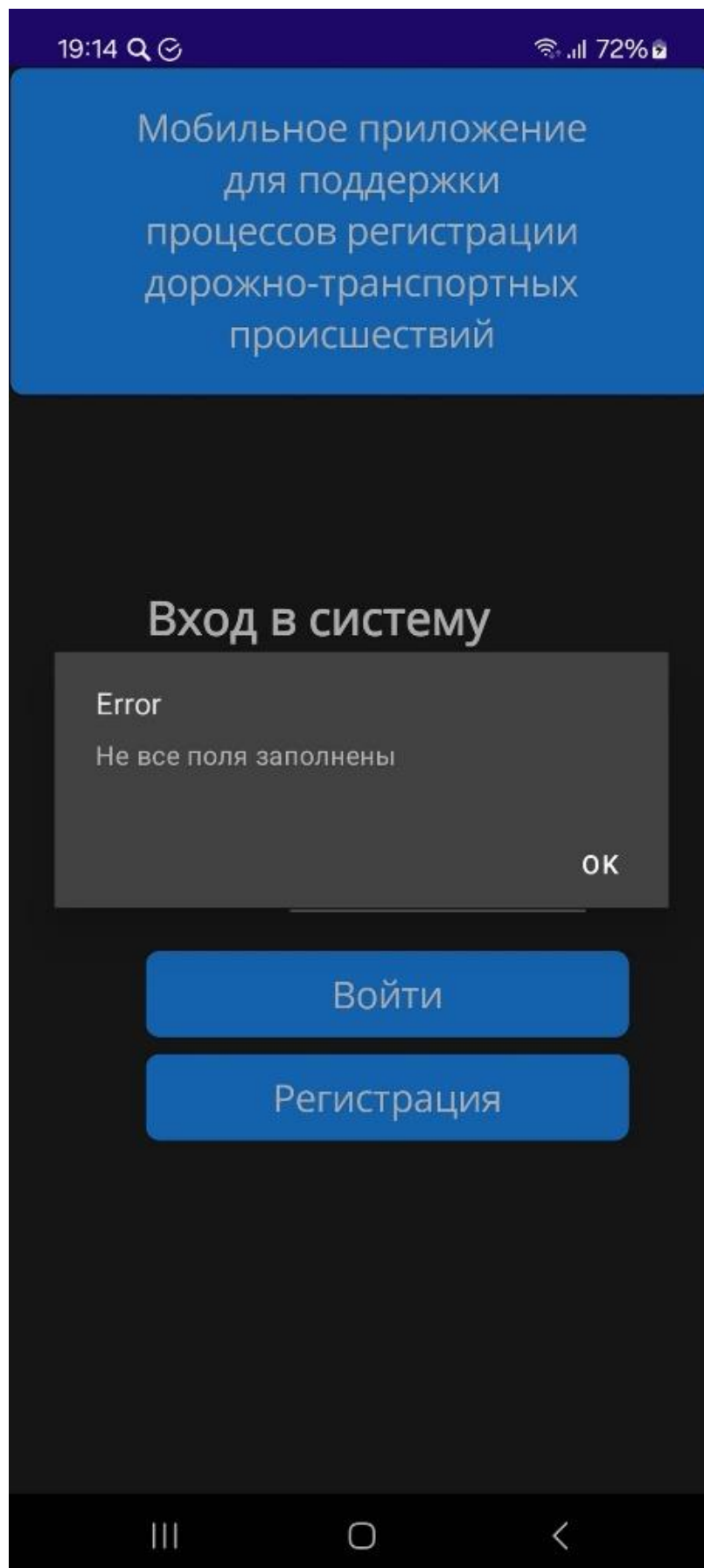


Рисунок Е.10 – Вход в систему без введенных данных

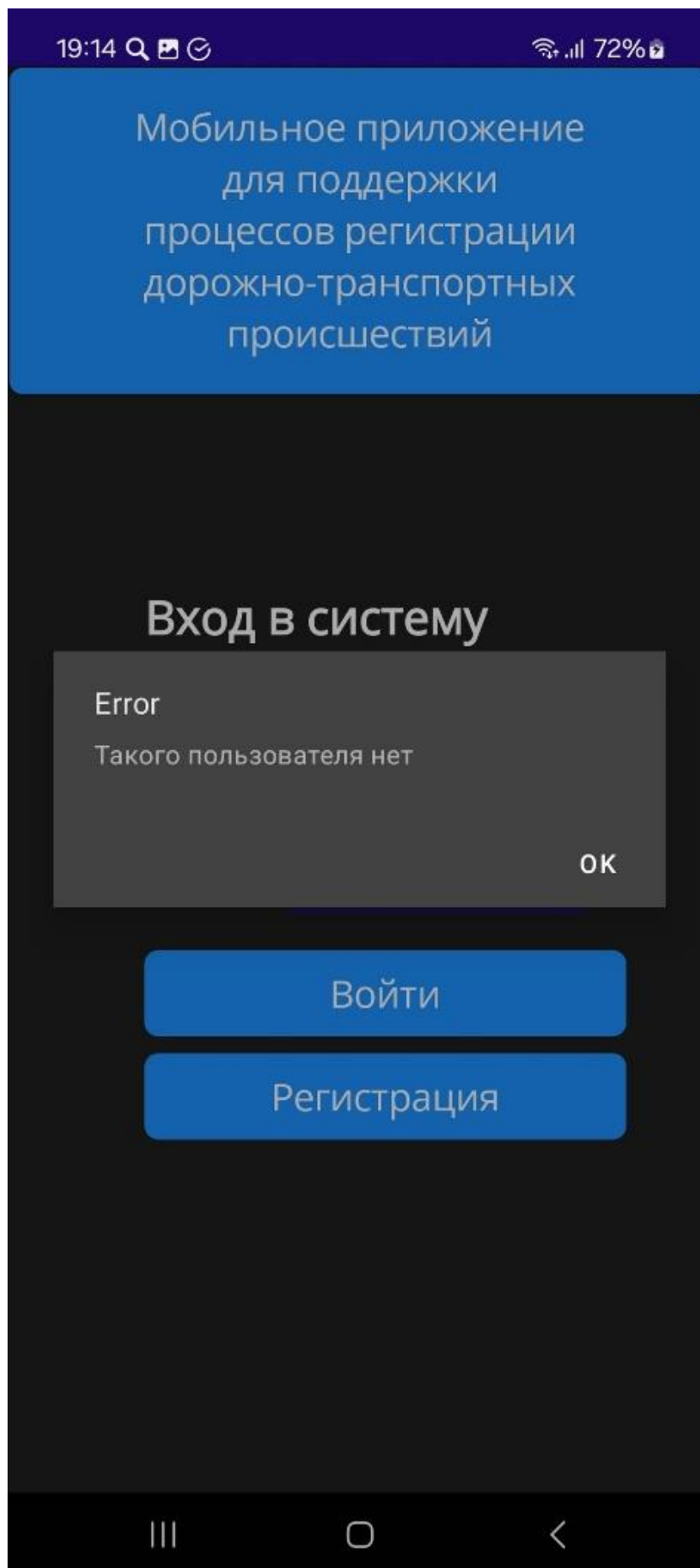


Рисунок Е.11 – Вход в отсутствующий личный кабинет

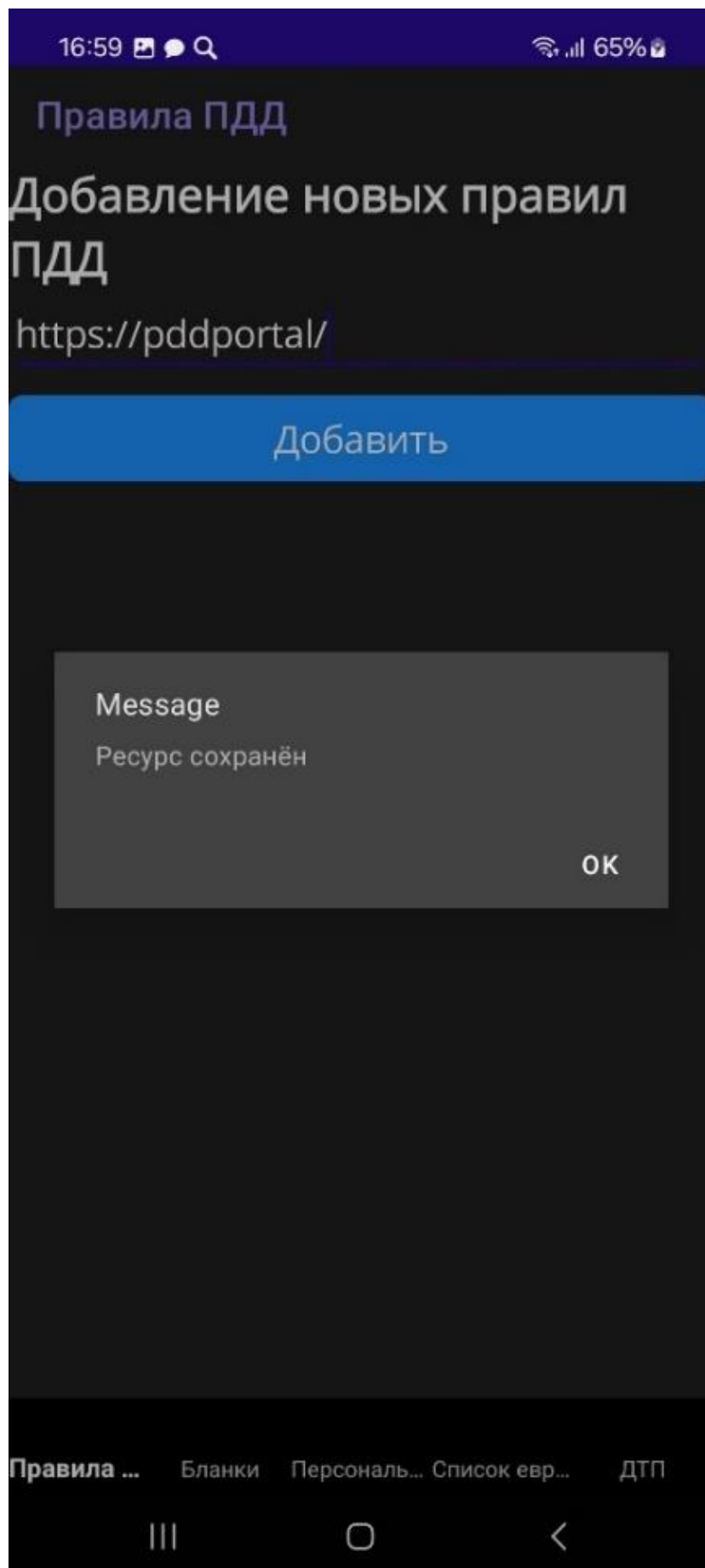


Рисунок Е.12 – Добавление новых правил ПДД

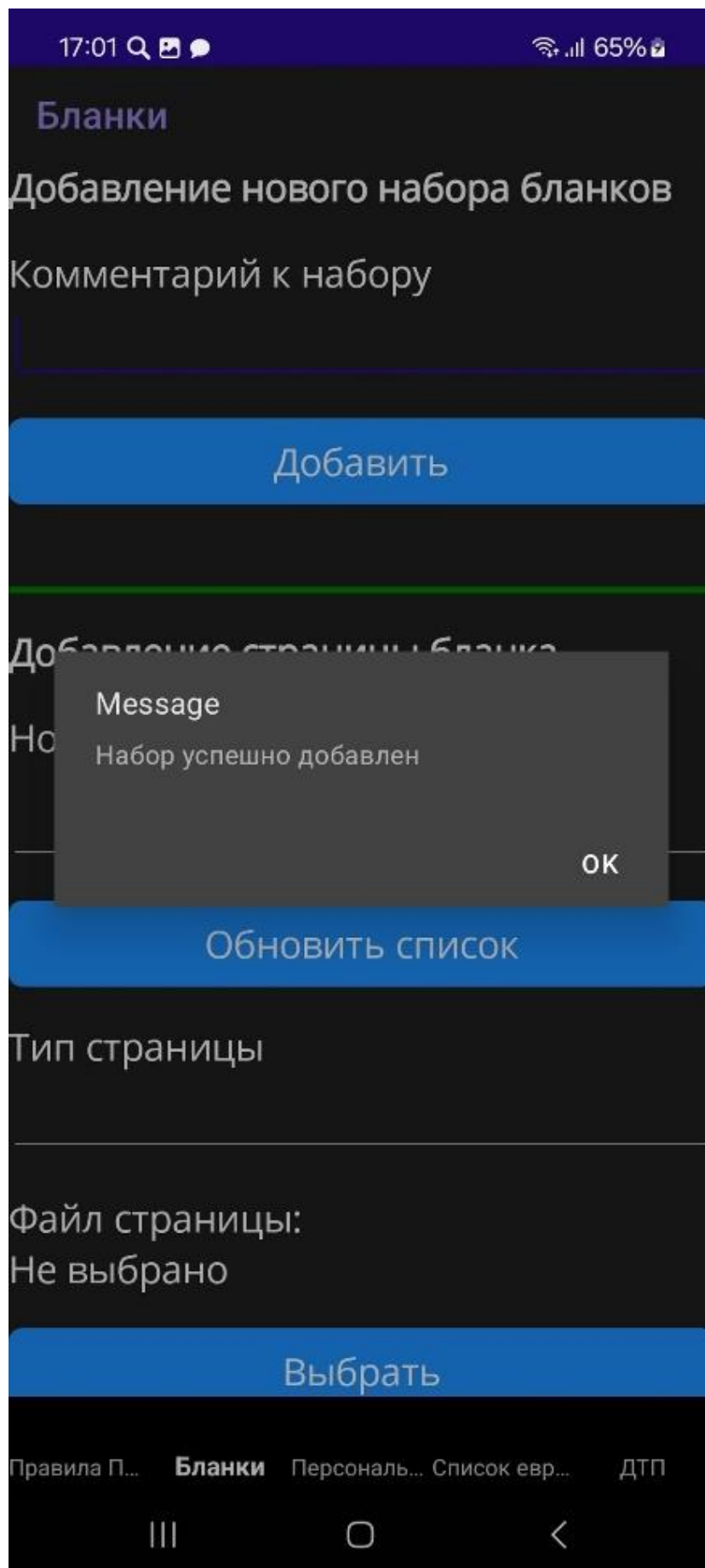


Рисунок Е.13 – Добавление нового набора бланков для европротокола

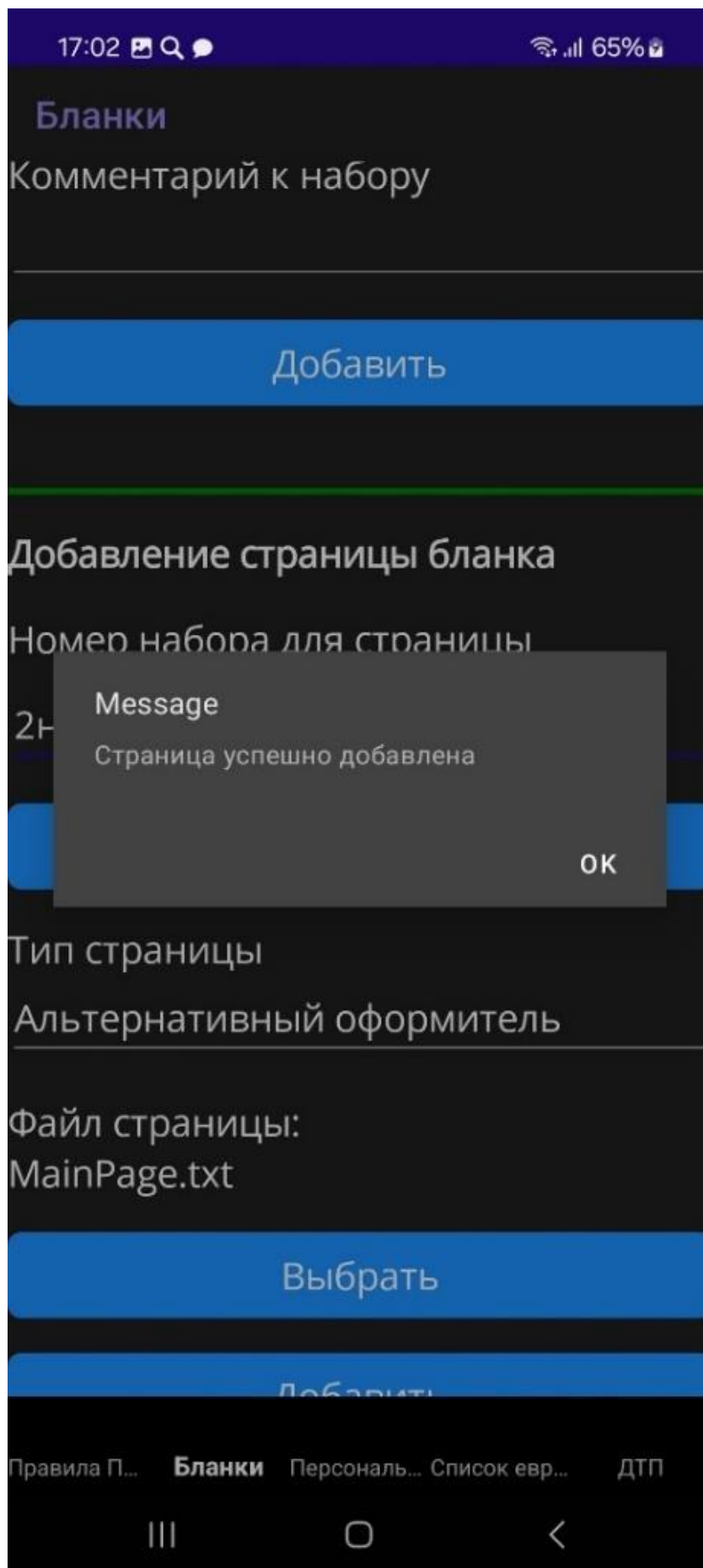


Рисунок Е.14 – Добавление нового бланка европротокола

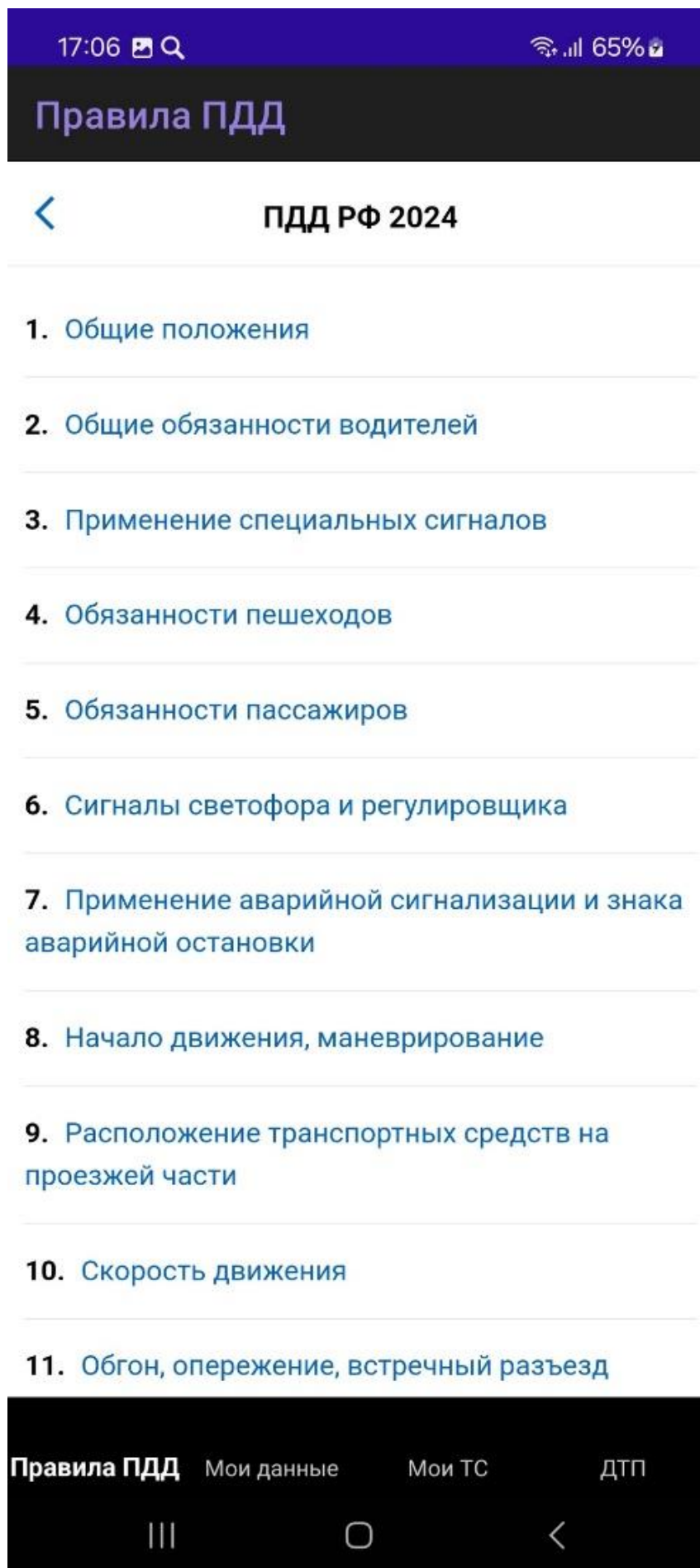


Рисунок Е.15 – Просмотр правил ПДД

17:06 65%

Мои данные

Имя Виктор

Отчество Петрович

Телефон 89898989000

Дата рождения 05.12.1999

Адрес ул. ...

Права Серия 1111 Номер 111111

Действительны до 22.03.2025

Исправить

Отменить изменения

Правила ПДД **Мои данные** Мои ТС ДТП

III O <

Message

Данные водителя обновлены

OK

Рисунок Е.16 – Обновление данных водителя

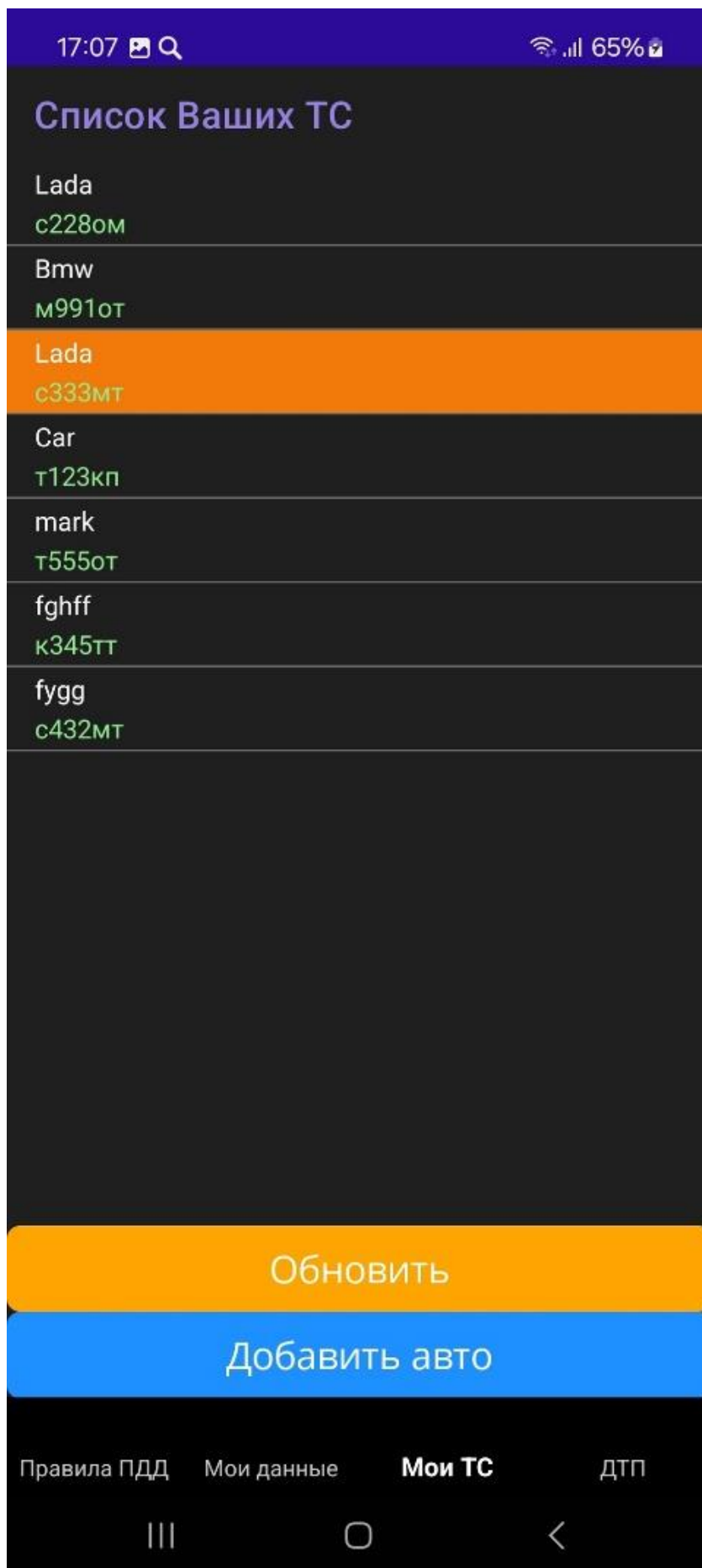


Рисунок Е.17 – Просмотр списка транспортных средств

The screenshot shows a mobile application interface for registration. At the top, the status bar displays the time 17:10, a search icon, and a battery level of 66%. The app's header is dark blue with a back arrow. The main form has a dark background. A label 'Пароль' (Password) is followed by a field containing four dots. Below this, a grey error message box is displayed, stating: 'Error', 'Плохой пароль. Пароль должен иметь: длину от 8 до 15 символов; английские заглавные и прописные буквы; специальные символы; цифры'. Below the error box is a grey message box with the text 'Message' and 'Успешно добавлено' (Successfully added), with an 'OK' button. Below the message box is a label 'Телефон' (Phone) followed by a field containing the number '64467865456'. At the bottom of the form is a large blue button labeled 'Зарегистрироваться' (Register). Below the form is a dark blue bar with three icons: a hamburger menu, a circle, and a back arrow. Below this bar is another large blue button labeled 'Продолжить' (Continue). At the very bottom is a dark blue bar with four links: 'Правила ПДД', 'Мои данные', 'Мои ТС', and 'ДТП'. Below this bar is another dark blue bar with the same three icons as the one above.

17:10 🔍 66%

←

Пароль

Error
Плохой пароль. Пароль должен иметь:
длину от 8 до 15 символов;
английские заглавные и прописные
буквы
специальные символы
цифры

Message
Успешно добавлено

OK

Телефон 64467865456

Зарегистрироваться

☰ ○ <

Продолжить

Правила ПДД Мои данные Мои ТС ДТП

☰ ○ <

Рисунок Е.18 – Заполнение европротокола

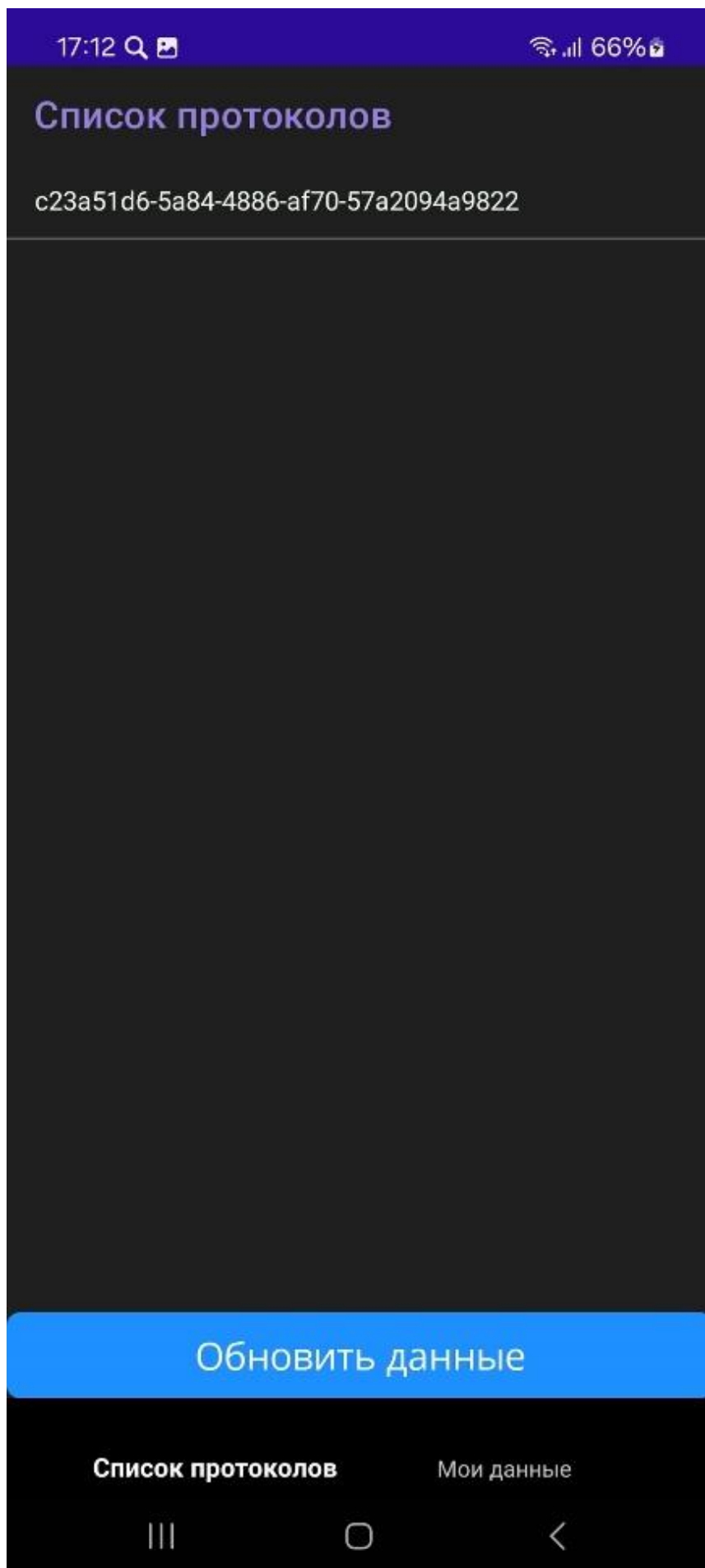


Рисунок Е.19 – Просмотр списка европротоколов

Электронный европротокол

Уникальный номер электронного европротокола - c23a51d6-5a84-4886-af70-57a2094a9822

Информация о ДТП

Дата

суббота, 8 июня 2024 г.

Время

12:20

Адрес ДТП

ул Дорожная 22

Обстоятельства ДТП. Участник 1

Совершал разворот

Пояснения. Участник 1

Не заметил автомобиль

Обстоятельства ДТП. Участник 2

Двигался прямо

Пояснения. Участник 2

Врезался в автомобиль

Фотографии с места ДТП

19:21 Q ↓

100%

Электронный

Список протоколов

Мои данные

Рисунок Е.20 – Просмотр оформленного европротокола

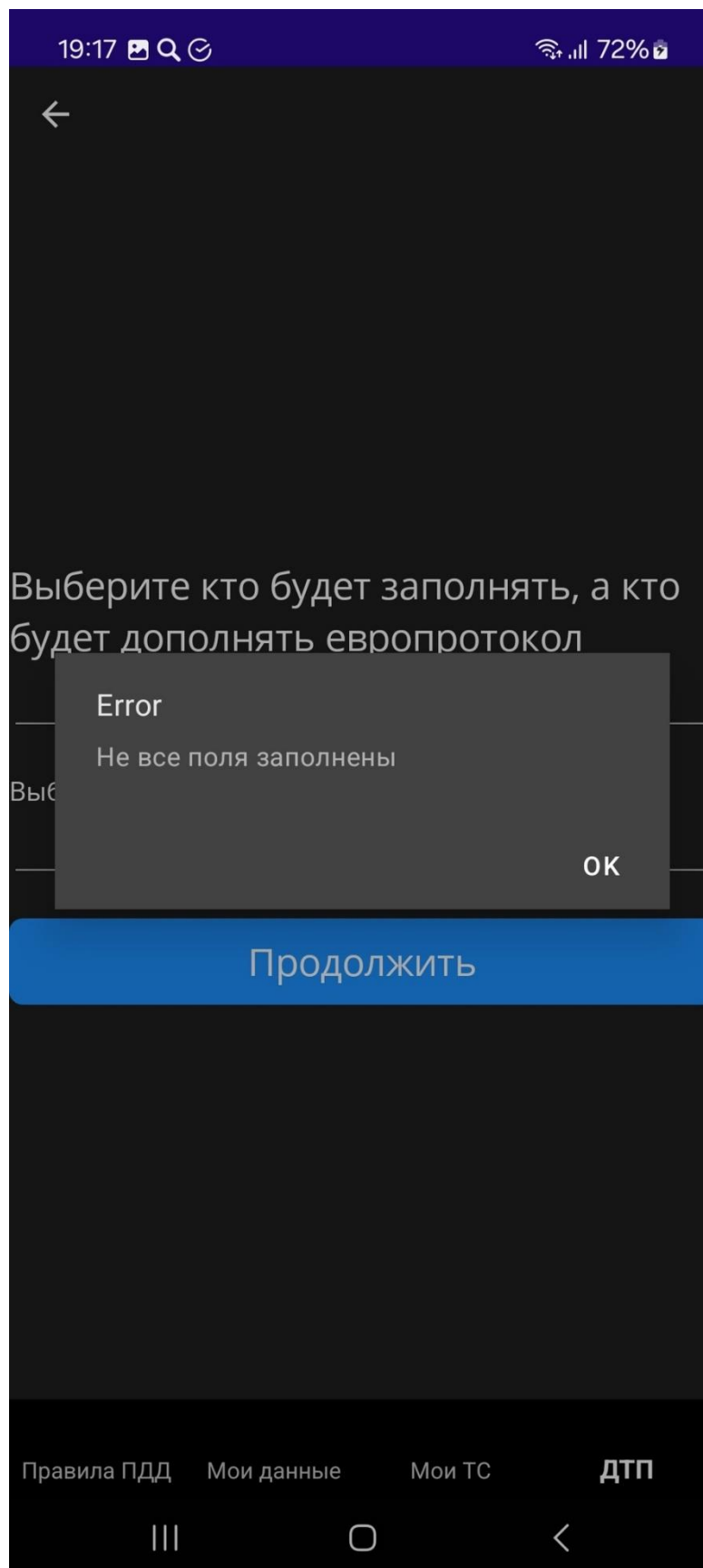


Рисунок Е.21 – Переход к оформлению европротокола без выбора роли