

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»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Мобильное приложение для сервиса покупки
и продажи автомобилей**

Студент _____ Мещанов Сергей Владимирович
(подпись, дата) (ФИО полностью)

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

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

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

Заведующий кафедрой _____ Макарычев П.П.
(подпись) (фамилия, инициалы)

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

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

Пенза, 2020

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

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ П.П. Макарычев
«__» _____ 2020 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Мещанов Сергей Владимирович гр. 16ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Мобильное приложение для сервиса покупки и продажи автомобилей

Тема утверждена приказом ПГУ № 1488/о от "30" декабря 2019

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 мобильное приложение, позволяющее владельцу публиковать сведения о своем автомобиле, чтобы другие пользователи могли найти необходимое им транспортное средство и договориться о его покупке.*

2) *Среда выполнения: операционная система Android 9.0 и выше, Windows 10*

Языковые средства разработки: C++, UML, SQL

Инструментальные средства разработки: Qt Creator 4.11.0,

Android Software Development Kit (SDK), jdk1.8.0_201, смартфон Samsung Galaxy A40,

Microsoft Project, Microsoft SQL Server Management Studio 2017, ERwin Data Modeler, ArgoUML

Технологии разработки: RUP, ООП, MVC

3) *Основные функции мобильного приложения:*

а) прохождение аутентификации;

б) прохождение авторизации;

в) добавление нового объявления;

г) отображение списка объявлений;

д) возможность редактирования и удаления объявления;

е) возможность добавить объявление в избранное.

5. Перечень подлежащих разработке вопросов

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

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	<i>Анализ предметной области и постановка задачи</i>	15%	13.04.2020– 21.04.2020	
2	<i>Проектирование</i>	25%	22.04.2020– 05.05.2020	
3	<i>Реализация</i>	40%	06.05.2020– 30.05.2020	
4	<i>Контроль качества программного обеспечения</i>	5%	31.05.2020– 02.06.2020	
5	<i>Оформление пояснительной записки</i>	15%	03.06.2020– 13.06.2020	

Дата выдачи задания "30" декабря 2019

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 95 листов, 37 рисунков, 12 таблиц, 22 использованных источника, 4 приложения.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ПРОДАЖА И ПОКУПКА АВТОМОБИЛЕЙ, ANDROID, RUP, UML, IDEF1, IDEF1X, QT-CREATOR, QT, C++.

Цель работы – разработка мобильного приложения для сервиса покупки и продажи автомобилей.

В процессе разработки программного продукта применялись технологии RUP и MVC. Проектирование проводилось с использованием языка моделирования UML.

Разработка клиентской и серверной частей мобильного приложения проводилась на языке программирования C++ с использованием библиотек Qt, в качестве среды разработки использовались среда программирования Qt Creator 4.11.0. Клиентская часть функционирует под управлением операционной системы Android 9.0 и выше, а серверная часть Windows 10. Работа с базой данных осуществлялась при помощи СУБД Microsoft SQL Server Management Studio 2017.

В результате выполнения выпускной квалификационной работы было разработано мобильное приложение для сервиса покупки и продажи автомобилей, отвечающее всем требованиям технического задания.

					ПГУ 09.03.04 – 08BP161.20 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата				
Разраб.		Мещанов С.В			Мобильное приложение для сервиса покупки и продажи автомобилей. Пояснительная записка	Лит.	Лист	Листов
Пров.		Князев В.Н					4	95
						Гр. 16ВП1		
Н. контр.		Такташкин Д.В						
Утв.		Макарычев П.П.						

Содержание

Содержание	5
Введение	7
1 Анализ предметной области и требований	9
1.1 Анализ предметной области и постановка задачи.....	9
1.2 Анализ аналогичных решений.....	11
1.2.1 Анализ мобильного приложения Auto.ru	11
1.2.2 Анализ мобильного приложения Drom.....	12
1.2.3 Анализ мобильного приложения Avito	13
1.3 Анализ целевой аудитории	14
1.4 Анализ требований	16
1.5 Анализ функциональных требований	16
1.6 Анализ нефункциональных требований	24
2 Проектирование мобильного приложения.....	25
2.1 Планирование разработки и оценка бюджета.....	25
2.2 Проектирование пользовательского интерфейса	34
2.3 Проектирование архитектуры приложения	39
2.4 Логическое проектирование базы данных	40
2.5 Проектирование классов клиентской части приложения	42
2.6 Проектирование классов серверной части приложения	43
3 Реализация мобильного приложения	44
3.1 Реализация базы данных	44
3.2 Реализация клиентской и серверной частей приложения	44
4 Контроль качества программного продукта	47
4.1 Функциональное тестирование	47
4.1.1 Метод эквивалентного разбиения	47
4.1.2 Метод границ.....	48
4.2 Метрики кода.....	49
4.2.1 Метрика Холстеда	50
4.2.2 Цикломатическая метрика Мак-Кейба.....	53
4.2.3 Метрика Чепина	54
Заключение	56
Список использованных источников	58
Приложение А. Глоссарий.....	61

Приложение Б. Исходный код приложения	63
Приложение В. Исходный код создания базы данных	79
Приложение Г. Результаты тестирования.....	81

Введение

С каждым годом информационные технологии все прочнее и прочнее оседают в различных сферах деятельности людей. Это связано с быстрым темпом развития этой отрасли, о чем свидетельствует утверждение программы «Цифровая экономика Российской Федерации» в 2017 году. Основной задачей этой программы является «создание необходимых условий для развития цифровой экономики Российской Федерации, в которой данные в цифровой форме являются ключевым фактором производства во всех сферах социально-экономической деятельности, что повышает конкурентоспособность страны, качество жизни граждан, обеспечивает экономический рост и национальный суверенитет» [1].

Развитие информационных технологий поспособствовало усовершенствованию смартфонов без особого увеличения их стоимости, что привело росту количества их владельцев. В целях реализации Стратегии развития информационного общества в Российской Федерации на 2017 – 2030 годы, вышеупомянутая программа направлена на «... повышение благосостояния и качества жизни граждан нашей страны путем повышения доступности и качества товаров и услуг, произведенных в цифровой экономике с использованием современных цифровых технологий ...» [2]. Это означает повсеместное доступное использование интернета и мобильных сетевых приложений.

Действительно, в настоящее время интернет представляет собой глобальную сеть для обмена информации. Люди, в свою очередь, начали использовать это в коммерческих целях, например, за счет создания различных торговых площадок. К ним можно отнести сервисы для продажи б/у автомобилей. Благодаря их развитию на второй план отошли объявления в газетах, журналах и реклама на телевидении. Начали появляться сайты и мобильные приложения с большим количеством объявлений. Несмотря на сложную экономическую ситуацию, продажи автомобилей с пробегом увеличиваются из года в год. Основная причина увеличения рынка

заключается в том, что реальные доходы населения не успевают за ростом цен на новые машины. Только за первое полугодие 2018 года цены увеличились от 2% до 10%. Этому способствовала совокупность факторов: повышение утилизационного сбора, рост акцизов; кроме того, у многих производителей летом традиционно меняется модельный год, и в связи с этим повышаются цены. Компании также стремятся компенсировать накопившуюся курсовую разницу [3].

Покупатель приходит к выводу, что желаемый автомобиль стоит слишком дорого и принимает решение дальше эксплуатировать имеющийся. Средний возраст автомобиля в России – свыше 13 лет по состоянию на 1 января 2018 года, в 2017 году этот показатель был равен 12,9 года. Парк легковых машин медленно, но верно стареет.

Рано или поздно покупатель встает перед выбором: или приобрести новый автомобиль в кредит, или купить авто с пробегом в рамках имеющегося бюджета. Можно сделать вывод, что рост цен на новые авто благоволит продажам б/у транспорта, как следствие – увеличению объема машин с пробегом в общей структуре продаж. Очевидно, в обозримом эта тенденция будет сохраняться.

Исходя из вышесказанного, целью выпускной квалификационной работы является создание актуального мобильного приложения для сервиса покупки и продажи автомобилей.

1 Анализ предметной области и требований

1.1 Анализ предметной области и постановка задачи

Автомобильный рынок представляет собой сферу деятельности, в которой происходит взаимодействие участников рынка по поводу обмена готовой автомобильной продукцией на денежные средства или их эквиваленты. Актуальность тематики подчеркивается широким распространением автомобильной продукции по всему миру. Автомобиль плотно вошел в жизнь человека, и современный транспорт уже невообразим без автомобильной техники.

По данным Ernst&Young мировой рынок автомобилей в начале XXI века развивается весьма динамично. За последние 10 лет продажи автомобилей прибавили порядка 30%. Характерно, что в 2008-2013 гг. рост на авторынке обеспечивался за счет развивающихся стран, а с 2013 года основная доля роста приходится на развитые страны, а также на Индию и Китай. В 2015 году глобальный автомобильный рынок вырос на 2%, а в 2016 году – на 4% [4].

Для дилеров рынок б/у автомобилей является зоной роста, поскольку доходность тут выше, чем от продажи новых машин – около 10%. Меньше срок оборачиваемости на складе, в среднем от 20 до 30 дней; а также быстрый цикл продажи от звонка до выдачи. Это обеспечивает повышенное внимание со стороны крупных игроков, что постепенно трансформирует рынок. Все больше клиентов ищут подержанные автомобили на интернет-площадках и через онлайн-сервисы, рассказала «Ведомостям» руководитель группы автомобильного сектора Deloitte Татьяна Кофанова. Именно в интернете, по ее мнению, группируется самая большая база вторичных машин [5].

Согласно данным аналитического агентства «АВТОСТАТ», в феврале 2020 года объем рынка легковых автомобилей с пробегом в России составил 407,5 тыс. единиц. Это на 11,1% больше по сравнению с аналогичным периодом 2019 года [6]. В таблице 1 представлен список самых продаваемых подержанных автомобилей в России на февраль за 2020 год в сравнении с тем же периодом за 2019 год.

Таблица 1 – Топ-10 марок легковых автомобилей с пробегом в феврале 2020 и 2019 годов

№	Марка	Февраль 2020	Февраль 2019	Изм., %	2 мес. 2020	2 мес. 2019	Изм., %
1	LADA	98 542	92 649	6,4	187 380	176 302	6,3
2	TOYOTA	45 527	40 848	11,5	87 040	79 541	9,4
3	NISSAN	23 344	20 831	12,1	44 460	40 472	9,9
4	HYUNDAI	21 687	18 201	19,2	41 058	35 141	16,8
5	KIA	20 694	16 602	24,6	39 150	31 905	22,7
6	CHEVROLET	17 749	16 431	8,0	33 547	31 608	6,1
7	VOLKSWAGEN	16 917	14 537	16,4	31 935	28 347	12,7
8	RENAULT	15 906	13 399	18,7	30 401	26 102	16,5
9	FORD	15 418	14 461	6,6	29 542	27 862	6,0
10	MITSUBISHI	12 188	11 189	8,9	23 231	21 979	5,7
	Всего по России	407 455	366 802	11,1	773 830	706 145	9,6

С развитием онлайн-сервисов клиенты стали тратить меньше времени на поиск автомобиля в разных салонах. Кроме того, жители других городов теперь меньше ездят за вторичным транспортом в Москву: цены в регионах и столице уравниваются, а местное предложение не уступает московскому. Почти все автомобили с пробегом сейчас продаются через сайты объявлений. Согласно аналитическому агентству «Автостат» весь рынок вторичных машин вырос на 2,1% до 5,3 млн шт.

Данная тенденция не удивительна, мобильные приложения и онлайн-сервисы по продаже автомобилей предоставляют большую базу автомобилей, широкий спектр критериев сортировки объявлений, возможность просмотреть фотографии авто и многое другое.

Разрабатываемое приложение предназначено для того, чтобы владельцы своих автомобилей могли подать объявление о их продаже, а люди, желающие приобрести транспортное средство, могли просматривать эти объявления или подать свое объявление о покупке. Данная концепция позволит потенциальным продавцам и покупателям взаимодействовать в одном приложении, не разграничивая роли. Пользователь может подать объявление о продаже своего текущего автомобиля и одновременно присматривать себе другие варианты.

Таким образом, мобильное приложение для сервиса покупки и продажи автомобилей должно предоставлять пользователям следующие основные функции:

- создать аккаунт;
- пройти авторизацию;
- добавить объявление;
- изменить объявление;
- удалить объявление;
- добавить объявления в избранное;
- просмотреть список объявлений.

1.2 Анализ аналогичных решений

В данный момент существует множество сервисов по продаже автомобилей. Из всего многообразия можно выделить следующие компании: Auto.ru, Дром и Авито. Они являются лидерами на Российском рынке и значительно превосходят своих конкурентов по количеству активных пользователей и просмотров.

1.2.1 Анализ мобильного приложения Auto.ru

15 июля 1996 года Михаилом Рогальским был создан сайт Auto.ru (Авто.ру). В 2014 году все права на сервис были приобретены компанией «Яндекс» [7]. Главный экран мобильного приложения приведен на рисунке 1.

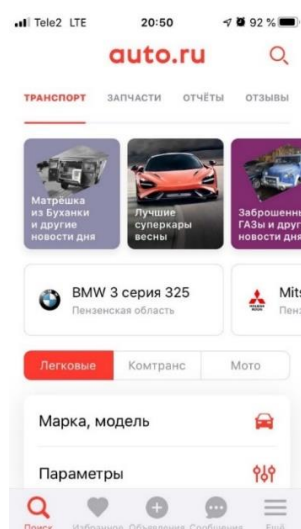


Рисунок 1 – Главный экран мобильного приложения Auto.ru

Приложение выполнено очень лаконично как в плане структуры, так и по содержанию. В нём содержится всего несколько разделов, не перегруженных функционалом.

При работе с приложением можно выделить несколько основных достоинств и недостатков.

Из достоинств следует отметить охват регионов по всей России, не зависимо от своего места проживания, будь то Москва или провинция, пользователь может найти объявление о продаже в любом регионе страны. Также присутствует возможность сравнения как объявлений между собой, так и отдельных моделей, что упрощает выбор покупателю. Сервис представлен мобильным приложением и веб-сайтом.

Однако, при наличии мощного инструмента сравнения объявлений, существует недостаток параметров их фильтрации, на что неоднократно жаловались пользователи. Частые замечания возникали и по поводу того, что отсутствуют какие-либо индикаторы в «Избранном» у объявлений, потерявших актуальность.

1.2.2 Анализ мобильного приложения Drom

Сервис Drom (Дром) основан в 2002 году, но свою популярность получил относительно недавно, хотя в 2010 году вошел в топ15 самых быстрорастущих проектов российского Интернета. Главный экран мобильного приложения Drom представлен на рисунке 2.

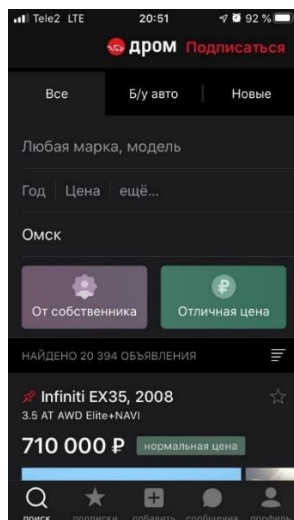


Рисунок 2 – Главный экран мобильного приложения Drom

Сервис имеет свою целевую аудиторию, которая полюбила его за большое многообразие параметров поиска объявлений и отсутствие явных недостатков технической части. Но при всем при этом многие пользователи выражают свое недовольство при использовании. Наибольший акцент уделяется на невозможность конкурировать бесплатным объявлениям по сравнению с платными, которые, в свою очередь, не гарантируют сто процентной продажи автомобиля. Также негатив вызывает и отсутствие сервиса поддержки. Многие объявления после оплаты не попадают на всеобщий обзор, но деньги за это не возвращают и не дают по этому поводу никаких комментариев.

1.2.3 Анализ мобильного приложения Avito

Это популярная площадка для покупки разнообразных товаров и услуг, в которой существует отдельный раздел по продаже автомобилей. Avito (Авито) был создан 15 октября 2007 года. Основали компанию Йонас Нордландер и Филип Энгельберт [8]. В 2011 году была запущена мобильная версия. Стартовая страница раздела транспорт мобильного приложения Avito показана на рисунке 3.

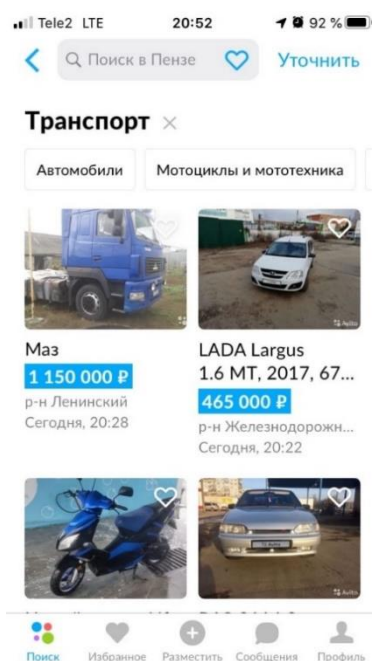


Рисунок 3 – Стартовая страница раздела транспорт
мобильного приложения Avito

Авито является одним из самых популярных сайтов объявлений о продаже и покупке чего бы то ни было, раздел «авто» неплохо конкурирует на рынке сервисов продажи вторичного транспорта. Приложение выполнено в стиле, удобном как для простых, так и для опытных пользователей сети.

Несмотря на свою популярность, сервис Авито, как и Дром, существует за счет платных объявлений. За немалые деньги объявление можно продвинуть вверх, цвет и шрифт заголовка будут выделенными, что, безусловно, влияет на выбор. Иногда продавец не указывает реальную стоимость транспортного средства, что затрудняет поиск по ценовой категории. Ко всему прочему объявление появляется только после модерации, через 30 минут после размещения.

Таким образом, в результате анализа мобильных приложений аналогичного характера было принято решение о создании собственного мобильного приложения для российского рынка автомобилей, не имеющего выявленных недостатков.

1.3 Анализ целевой аудитории

Важно проанализировать какими платформами пользуются люди на мобильных устройствах. Согласно Statcounter.com, среди смартфонов самая популярная операционная система в России – это Android, с долей рынка в 80,33%, на втором месте операционная система iOS, у которой сосредоточены 19,3% пользователей. У оставшихся конкурентов процент потребителей меньше одного [10]. Статистика распределения платформ к 2020 году представлена на рисунке 4.

Исходя из статистики, можно сделать вывод, что явным лидером среди мобильных операционных систем является Android, соответственно, разработку приложения следует проводить для данной платформы, с целью охвата большего числа пользователей.

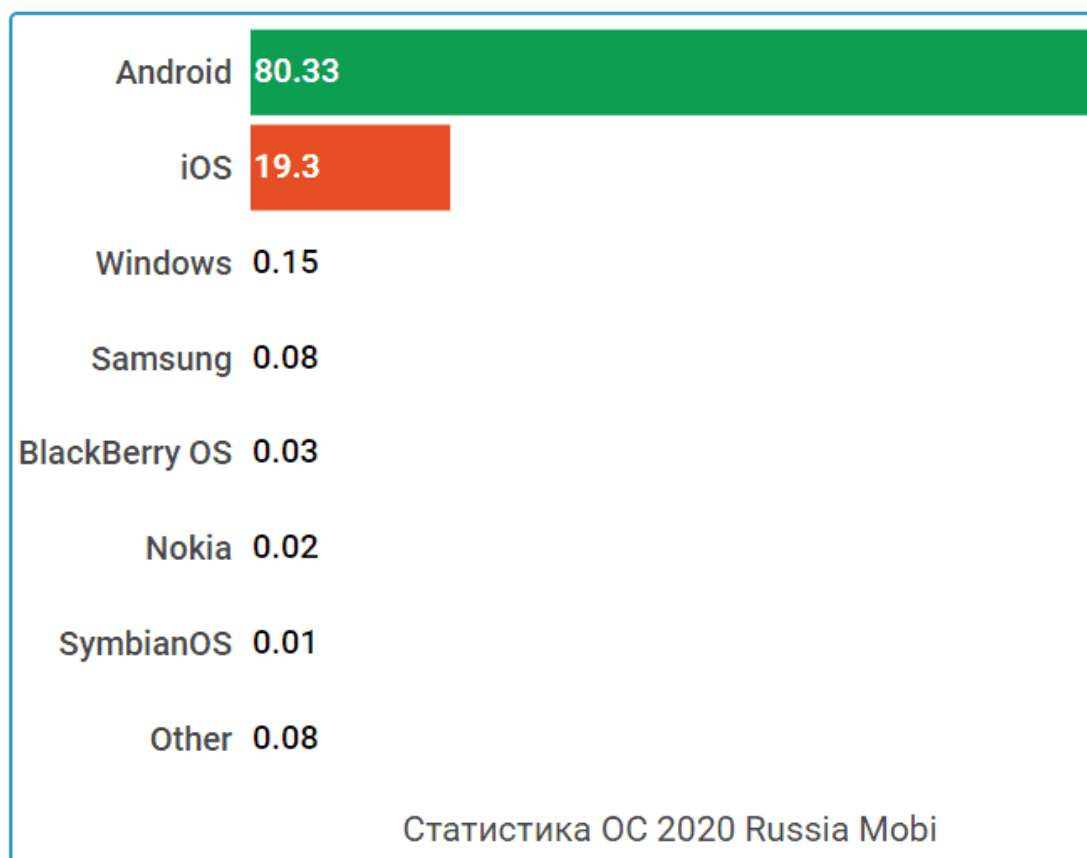


Рисунок 4 – Статистика распределения ОС смартфонов в России на 2020 год

Необходимо также проанализировать какие версии операционной системы нужно поддерживать. На рисунке 5 представлена статистика используемых версий для Android с апреля 2019 года по апрель 2020 года [11].

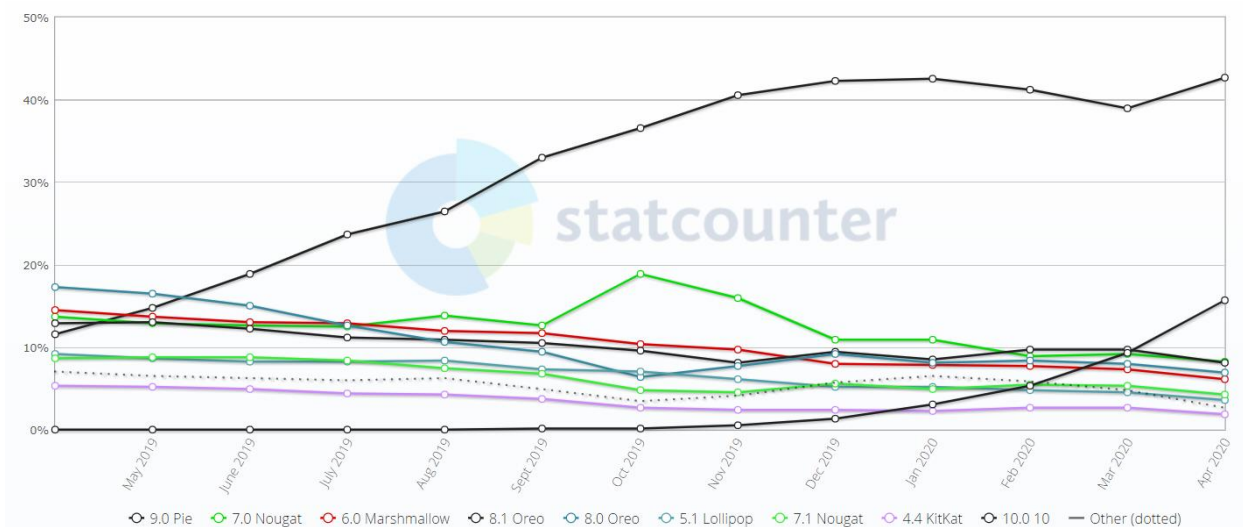


Рисунок 5 – Статистика используемых версий Android в России

Из рисунка видно, что наиболее используемой версией является 9.0 Pie, у которой 42,66% пользователей. На втором месте расположилась версия 10.0

с 15,75% пользователей. Учитывая данную статистику, можно сделать вывод, что разработка должна производиться для версий 9.0 и старше.

1.4 Анализ требований

Анализ требований – это процесс сбора требований к программному обеспечению (ПО), их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения [12].

Выделяют функциональные и нефункциональные требования. Функциональные требования – это перечень сервисов, которые должна выполнять система, причём должно быть указано, как система реагирует на те или иные входные данные, как она ведёт себя в определённых ситуациях и т.д. В некоторых случаях указывается, что система не должна делать. Нефункциональные требования описывают характеристики системы и её окружения, а не поведение системы. Также может быть приведён перечень ограничений, накладываемых на действия и функции, выполняемые системой. Они включают временные ограничения, ограничения на процесс разработки системы, стандарты и т.д [13].

1.5 Анализ функциональных требований

В результате анализа предметной области и продуктов-аналогов были выделены следующие функциональные требования к системе:

- пройти аутентификацию;
- пройти авторизацию;
- добавить объявление;
- изменить объявление;
- удалить объявление;
- добавить объявления в избранное;
- просмотреть список объявлений.

Диаграмма вариантов использования мобильного приложения представлена на рисунке 6.

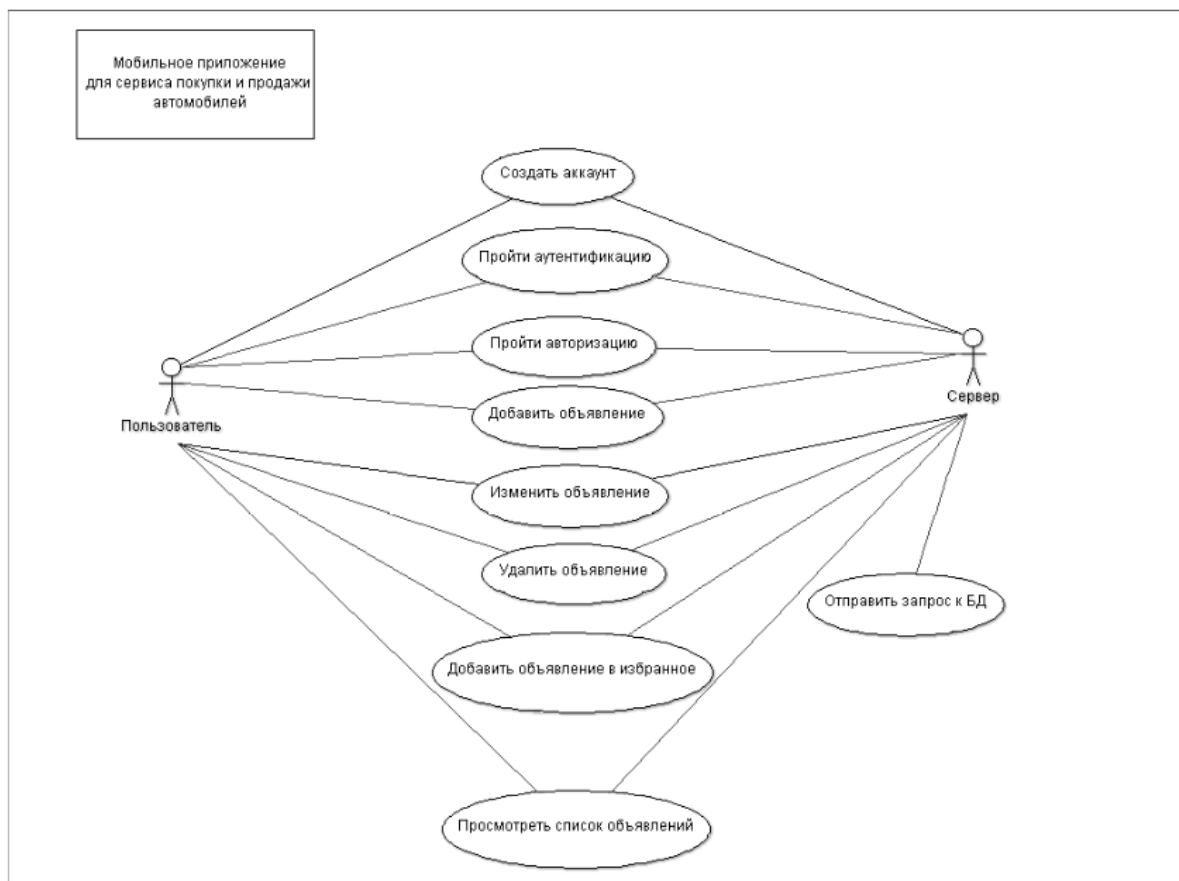


Рисунок 6 – Диаграмма вариантов использования мобильного приложения

Диаграмма отражает отношения между вариантами использования и актерами и описывает систему на концептуальном уровне [14]. Было выделено два типа актеров: пользователь и сервер, который предоставляет данные из БД. На диаграмме видно, какие функции выполняет каждый актер.

Для подробного рассмотрения предполагаемого поведения системы приведем подробную спецификацию некоторых вариантов использования. На рисунке 7 изображена спецификация прецедента «Создать аккаунт».

Спецификация данного прецедента показывает, как проходит процесс создания аккаунта в системе, и какие актеры принимают участие. Из основного потока видно, какие функции выполняет каждый актер.

Прецедент: Создать аккаунт
ID: 1
Краткое описание: Пользователь заполняет данные о себе в предложенные поля для создания аккаунта
Главные актёры: Пользователь, Сервер
Второстепенные актёры: Нет
Предусловия: Клиент открыл приложение
Основной поток: <ol style="list-style-type: none"> 1. Пользователь вводит логин в поле ввода 2. Пользователь вводит пароль в поле ввода 3. Пользователь вводит имя в поле ввода 4. Пользователь вводит номер телефона в поле ввода 5. Пользователь нажимает кнопку «Войти» 6. Отправляется сообщение серверу 7. Сервер отправляет запрос к БД 8. Сервер отправляет информацию от БД к Пользователю
Постусловия: Пользователь попадает на главную страницу
Альтернативные потоки: Пользователь ввел некорректные данные

Рисунок 7 – Спецификация варианта использования «Создать аккаунт»

На рисунке 8 изображена спецификация варианта использования «Добавить объявление».

Прецедент: Добавить объявление
ID: 2
Краткое описание: Пользователь заполняет данные об автомобиле, и новое объявление добавляется в общий список объявлений
Главные актёры: Пользователь, Сервер
Второстепенные актёры: Нет
Предусловия: Клиент открыл вкладку создания объявлений
Основной поток: <ol style="list-style-type: none"> 1. Пользователь выбирает тип объявления 2. Пользователь заполняет информацию об автомобиле 3. Пользователь нажимает кнопку «Создать» 4. Отправляется запрос к серверу 5. Сервер записывает полученные данные в БД 6. Сервер отправляет результат Пользователю
Постусловия: Пользователь получает сообщение об успешном создании объявления
Альтернативные потоки: Пользователь ввел некорректные данные

Рисунок 8 – Спецификация варианта использования «Добавить объявление»

Вариант использования «Добавить объявление» предназначен для пользователя. Использование данного прецедента позволяет разместить объявление о продаже или о покупке автомобиля, в зависимости от типа объявления. Объявление содержит в себе данные об автомобиле, а также имя и номер телефона пользователя, который создал объявление.

На рисунке 9 изображена спецификация варианта использования «Отправить запрос к БД».

Прецедент: Отправить запрос к БД
ID: 3
Краткое описание: Сервер формирует запрос к БД
Главные актёры: Сервер
Второстепенные актёры: Нет
Предусловия: Сервер получил данные от Пользователя
Основной поток: <ol style="list-style-type: none"> 1. Сервер выбирает тип запроса 2. Сервер создает запрос 3. Сервер дает команду БД на выполнение запроса
Постусловия: Сервер считывает данные из БД
Альтернативные потоки: Нет

Рисунок 9 – Спецификация варианта использования «Отправить запрос к БД»

Данный вариант использования предназначен для того, чтобы сервер формировал запросы к базе данных и посылал сигнал к ней об их выполнении.

Описание прецедентов позволяет подробно рассмотреть работу некоторых вариантов использования.

Для более наглядного рассмотрения требований к этим прецедентам, были построены диаграммы деятельности для них. На рисунке 10 изображена диаграмма для варианта использования «Создать аккаунт» .

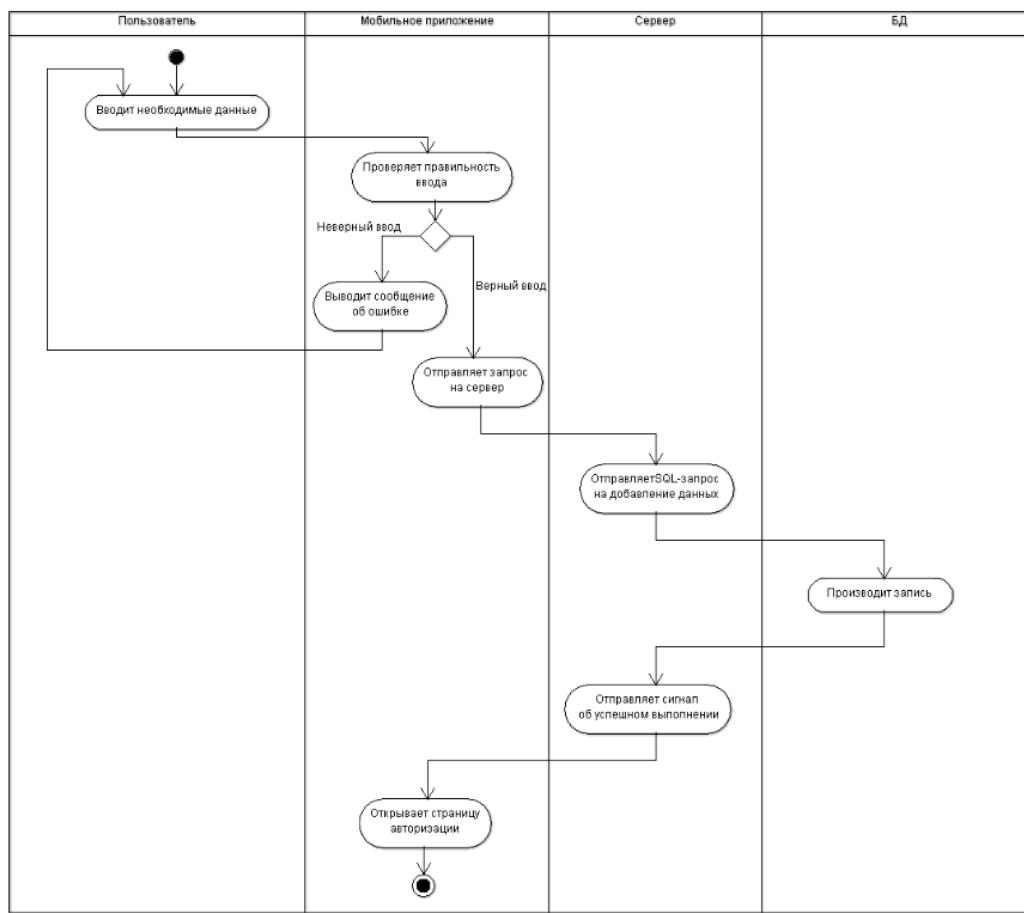


Рисунок 10 – Диаграмма деятельности для варианта использования «Создать аккаунт»

Диаграмма показывает, как происходит выполнение варианта использования «Создать аккаунт», для которого ранее была приведена спецификация. Пользователь работает с полями ввода мобильного приложения и вводит в них свое имя, номер сотового телефона, а также выбирает логин и пароль. Мобильным приложением происходит проверка введенных данных, если присутствует ошибка при заполнении полей, то тогда пользователь получит соответствующее сообщение. Сервер выполняет функцию чтения данных пользователя с последующей их записью в базу данных.

Теперь рассмотрим диаграмму деятельности для варианта использования «Добавить объявление». Она изображена на рисунке 11.

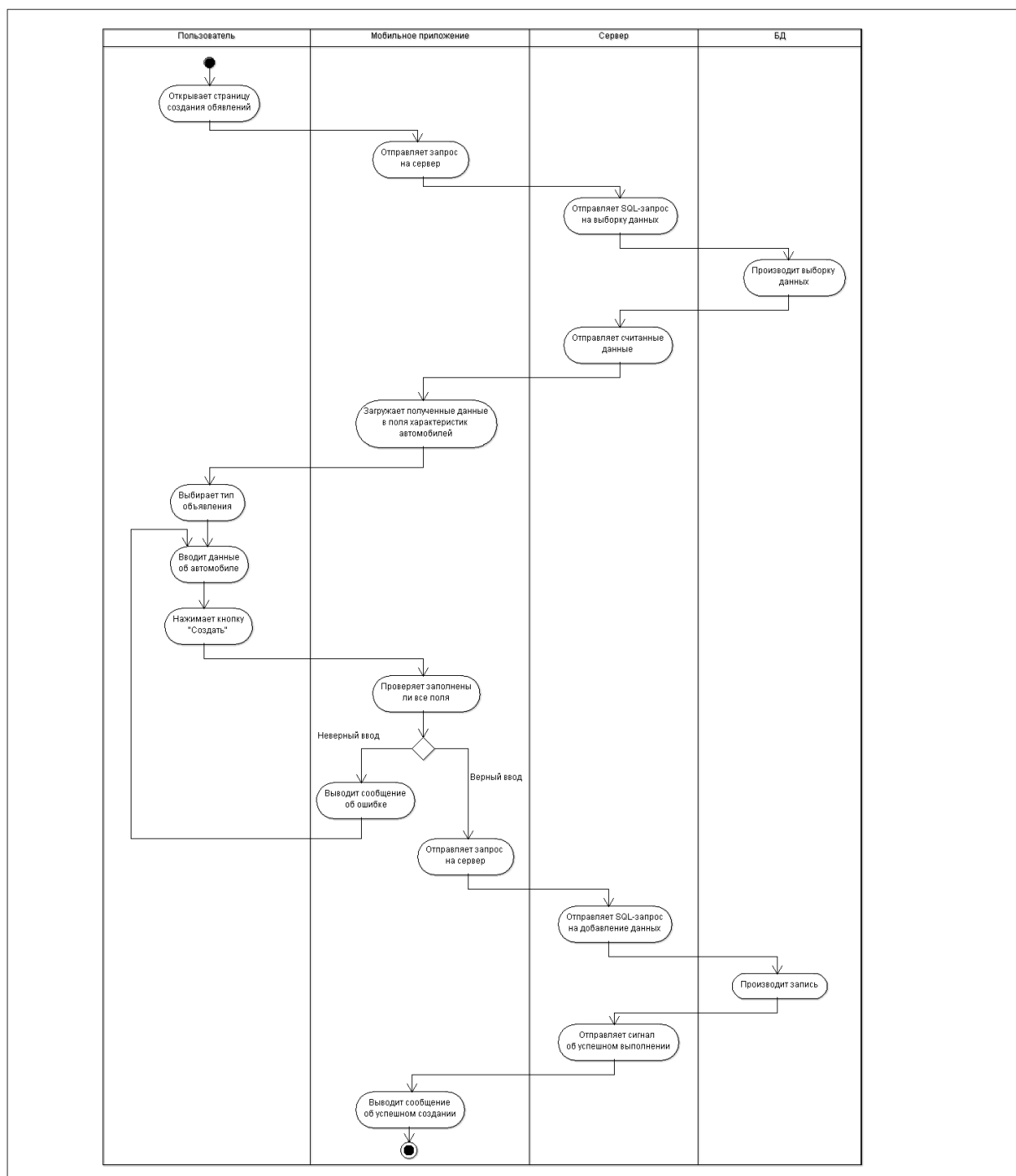


Рисунок 11 – Диаграмма деятельности для варианта использования «Добавить объявление»

На диаграмме видно, как проходит процесс создания объявления. После открытия страницы из базы данных берутся данные для заполнения полей выбора характеристик автомобиля. Выбирая, необходимые параметры, пользователь создает объявление, которое сервером посылается в базу данных и становится доступным для других пользователей.

Для более детального рассмотрения логики вариантов использования были разработаны диаграммы последовательности. На рисунке 12 изображена диаграмма последовательности для варианта использования «Создать аккаунт».

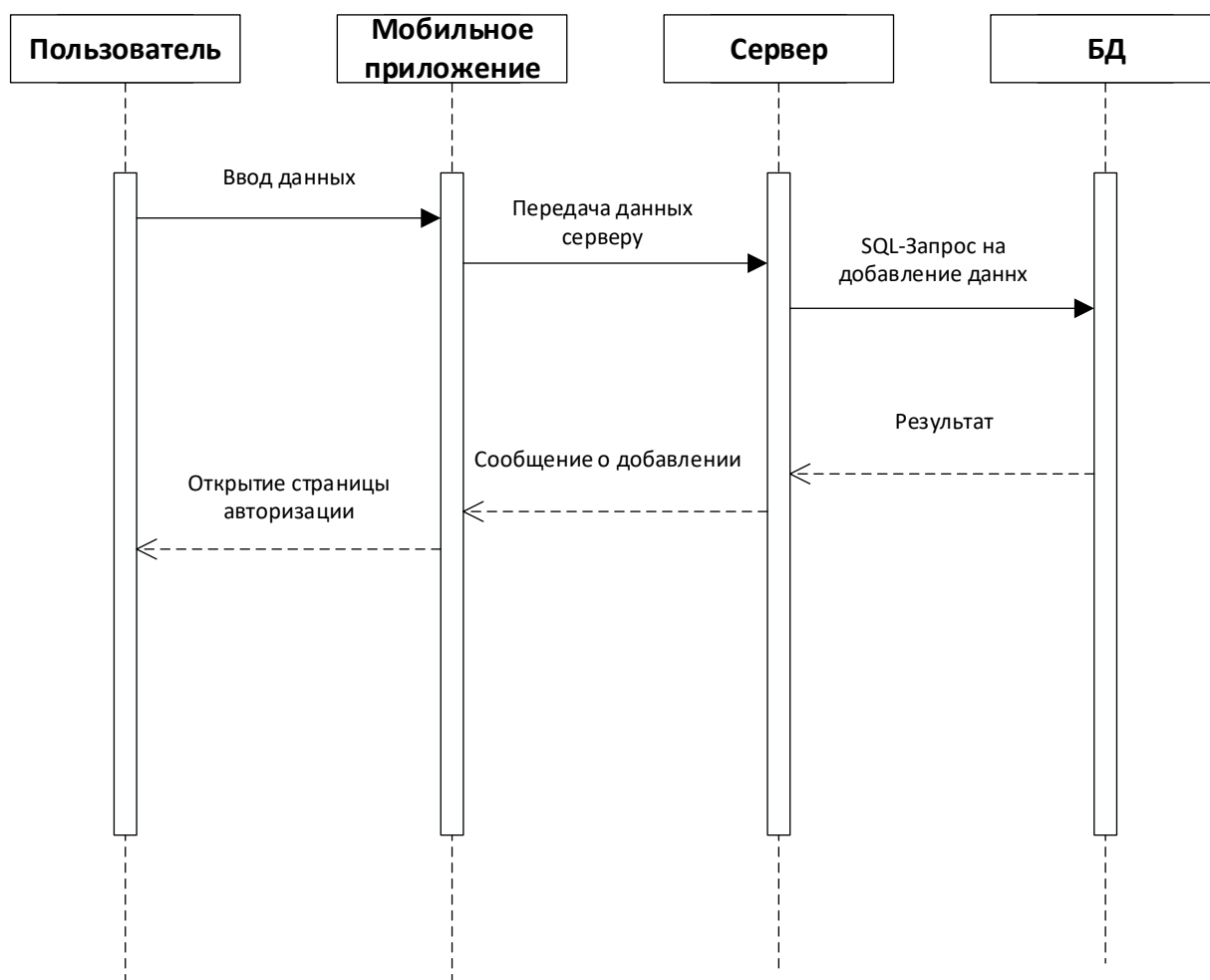


Рисунок 12 – Диаграмма последовательности для варианта использования «Создать аккаунт»

Из диаграммы видно в какой последовательности выполняются действия для варианта использования «Создать аккаунт». Выполнение начинается пользователем с ввода данных в мобильное приложение, затем эти данные передаются на сервер, а сервер, в свою очередь в дальнейшем записывает их в базу данных. Процесс получения ответа пользователем происходит в обратном порядке. Сначала сервер получает данные из БД, затем передает их на мобильное приложение, которое в случае успеха открывает пользователю страницу авторизации.

В аналогичном порядке происходит выполнение прецедента «Добавить объявление». Последовательность действий начинается с открытия страницы пользователем. Мобильное приложение посылает сигнал серверу о начале работы, сервер после этого считывает данные с БД и отправляет их мобильному приложению, которое выводит информацию пользователю для выбора характеристик автомобиля. После выбора всех необходимых параметров, данные попадают на сервер от мобильного приложения и записываются в БД. Затем сервер посылает ответ через мобильное приложение об успешном добавлении. Диаграмма для варианта использования «Добавить объявление» представлена на рисунке 13.

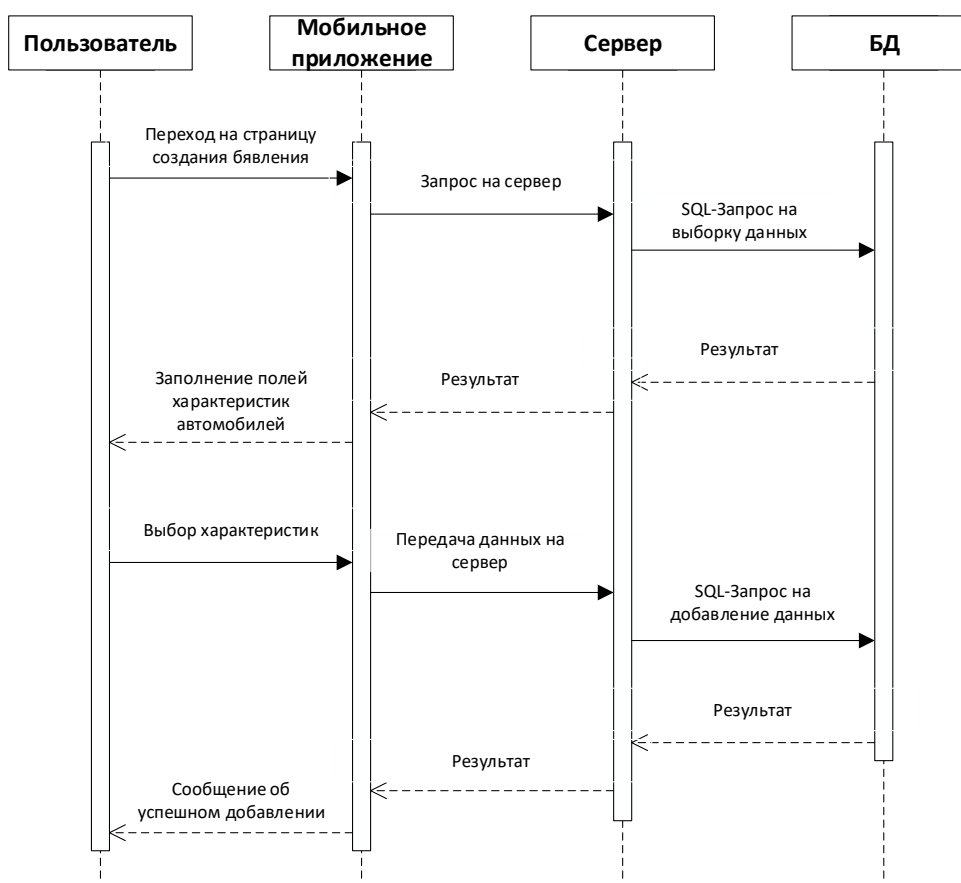


Рисунок 13 – Диаграмма последовательности для варианта использования «Добавить объявление»

В результате анализа функциональных требований была разработана диаграмма вариантов использования, приведена спецификация некоторых прецедентов. Для уточнения вариантов использования были представлены диаграммы деятельности и последовательности.

1.6 Анализ нефункциональных требований

Разработку разделим на 3 составляющих: разработка клиентской части, разработка серверной части, проектирование базы данных. Соответственно, составим требования для каждой из них отдельно.

Приведем требования к клиентской части:

- приложение должно обладать интерфейсом, удобным и понятным для пользователя;
- в приложении не должно возникать исключительных ситуаций, приводящих к аварийному завершению работы;
- приложение должно быть расширяемым в случае появления новых функциональных требований;
- разработка приложения должна осуществляться в среде Qt-creator при помощи фреймворка Qt на языке C++;
- приложение должно функционировать на смартфонах с операционной системой Android 9.0 и выше.

Из требований к серверной части выделим следующее:

- обмен должен осуществляться по протоколу TCP;
- разработку серверной части осуществлять в среде Qt-creator 4.11.0 при помощи фреймворка Qt версии 5.12 на языке C++.

Выделим требования к базе данных:

- в качестве целевой СУБД использовать Microsoft SQL Server Managmen Studio 17;
- запросы к базе данных должны осуществляться на языке SQL.

В результате анализа были составлены нефункциональные требования к клиентской части, серверной части и базе данных.

2 Проектирование мобильного приложения

2.1 Планирование разработки и оценка бюджета

Для определения трудоемкости и стоимости проекта необходимо провести его грамотное планирование. Для этого были определены список задач, решение которых необходимо для создания приложения.

В таблице 2 представлены задачи, представляющие из себя этапы создания проекта.

Таблица 2 – Список задач проекта

№	Название задачи	Длительность	Предшественники
1	Начало проекта	-	-
2	Анализ требований	8 дней	-
3	Обзор аналогов продукта	3	1
4	Изучение рынка	5	1
5	Анализ функциональных требований	2 дня	3;4
6	Составление технического задания	1 день	5
7	Анализ требований завершен	-	6
8	Выбор средств реализации проекта	8 дней	-
9	Выбор платформы	2 дня	7
10	Анализ возможных программных средств	4 дня	9
11	Выбор программного средства	2 дня	10
12	Выбор средств реализации завершен	-	11
13	Реализация проекта	37 дней	-
14	Выбор архитектуры ПО	2 дня	12
15	Проектирование пользовательского интерфейса	5 дней	14
16	Создание клиентской части	13 дней	-
17	Экран поиска автомобилей	7 дней	15
18	Экран добавления объявления	4 дня	17
19	Экран избранного	2 дня	18
20	Проектирование базы данных	8 дней	-
21	Концептуальное проектирование БД	3 дня	14

Продолжение таблицы 2

№	Название задачи	Длительность	Предшественники
22	Логическое проектирование БД	2 дня	21
23	Физическое проектирование БД	2 дня	22
24	Создание триггеров и хранимых процедур	1 день	23
25	Создание серверной части	27 дней	-
26	Программирование взаимодействия с БД	6 дней	24
27	Программирование функций пользователя	14 дней	26;19
28	Программирование сервиса регистрации	3 дня	27
29	Реализация проекта завершена	-	28
30	Отладка	12 дней	-
31	Тестирование проекта	5 дней	29
32	Исправление ошибок	7 дней	31
33	Отладка завершена	-	32
34	Внедрение приложения	5 дней	-
35	Отправка мобильного приложения в GooglePlay	1 день	33
36	Исправление ошибок при внедрении	4 дня	35
37	Внедрение завершено	-	36
38	Конец проекта	-	37

Всего в таблице содержатся 38 задач, которые необходимо решить для реализации проекта. В таблице 3 представлен список затрачиваемых ресурсов.

Для каждого ресурса приведены следующие показатели: тип, стандартная ставка и ставка сверхурочных. Для ресурсов с типом «Затраты» стандартная ставка и ставка сверхурочных не определены. Ресурсы с типом «Материальный» также не имеют показателя ставки сверхурочных.

Таблица 3 – Список ресурсов

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Программист 1	Трудовой	150 Р\ч	175 Р\ч
Программист 2	Трудовой	200 Р\ч	220 Р\ч
Менеджер проекта	Трудовой	225 Р\ч	250 Р\ч
Тестировщик	Трудовой	180 Р\ч	200 Р\ч
Компьютер	Материальный	1000 Р	-
Android-устройство	Материальный	600 Р	-
Интернет	Затраты	-	-

Для каждой задачи представим ресурсы, необходимые для ее решения.

Распределение ресурсов представлено в таблице 4.

Таблица 4 – Распределение ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало проекта		
2	Анализ требований		
3	Обзор аналогов продукта	Менеджер проекта Компьютер Интернет	50% 1 шт
4	Изучение рынка	Менеджер проекта Компьютер Интернет	50% 1 шт
5	Анализ функциональных требований	Менеджер проекта Программист 1 Программист 2 Компьютер	50% 25% 25% 3 шт
6	Составление технического задания	Менеджер проекта Компьютер	100% 1 шт
7	Анализ требований завершен		
8	Выбор средств реализации проекта		
9	Выбор платформы	Менеджер проекта Компьютер	100% 1 шт
10	Анализ возможных программных средств	Менеджер проекта Программист 1 Компьютер Интернет	50% 50% 2 шт
11	Выбор программного средства	Менеджер проекта	100%

Продолжение таблицы 4

№	Название задачи	Ресурс	Единицы
12	Выбор средств реализации завершен		
13	Реализация проекта		
14	Выбор архитектуры ПО	Менеджер проекта Компьютер	100% 1 шт
15	Проектирование пользовательского интерфейса	Менеджер проекта Программист 1 Компьютер Android-устройство	50% 25% 1 шт 1 шт
16	Создание клиентской части		
17	Экран поиска автомобилей	Программист 1 Компьютер Android-устройство	50% 1 шт 1 шт
18	Экран добавления объявления	Программист 1 Компьютер Android-устройство	50% 1 шт 1 шт
19	Экран избранного	Программист 1 Компьютер Android-устройство	50% 1 шт 1 шт
20	Проектирование базы данных		
21	Концептуальное проектирование БД	Программист 1 Компьютер	50% 1 шт
22	Логическое проектирование БД	Программист 1 Компьютер	50% 1 шт
23	Физическое проектирование БД	Программист 1 Компьютер	50% 1 шт
24	Создание триггеров и хранимых процедур	Программист 1 Компьютер	50% 1 шт
25	Создание серверной части		
26	Программирование взаимодействия с БД	Программист 2 Компьютер Интернет	100% 1 шт
27	Программирование функций пользователя	Программист 2 Компьютер Интернет	100% 1 шт
28	Программирование сервиса регистрации	Программист 2 Компьютер	100% 1 шт
29	Реализация проекта завершена		
30	Отладка		
31	Тестирование проекта	Тестировщик Android-устройство Интернет	100% 1 шт
32	Исправление ошибок	Тестировщик	100%

Окончание таблицы 4

№	Название задачи	Ресурс	Единицы
33	Отладка завершена		
34	Внедрение приложения		
35	Отправка мобильного приложения в GooglePlay	Менеджер проекта Компьютер Интернет	100% 1 шт
36	Исправление ошибок при внедрении	Тестировщик Компьютер Android-устройство	100% 1 шт 1 шт
37	Внедрение завершено		
38	Конец проекта		

Для решения задач будет необходимо наличие двух программистов, менеджера проекта, тестировщика, компьютера и Android-устройства с выходом в интернет.

На рисунке 14 изображено введенное в Project содержание таблицы 2 с вычисленными датами начала и завершения каждой задачи. За начало проекта была принята дата 10.02.2019.

№	Название задачи	Длительность	Начало	Окончание
1	Начало проекта	0 дней	Пн 10.02.20	Пн 10.02.20
2	Анализ требований	8 дней	Пн 10.02.20	Ср 19.02.20
3	Обзор аналогов продукта	3 дней	Пн 10.02.20	Ср 12.02.20
4	Изучение рынка	5 дней	Пн 10.02.20	Пт 14.02.20
5	Анализ функциональных требований	2 дней	Пн 17.02.20	Вт 18.02.20
6	Составление технического задания	1 день	Ср 19.02.20	Ср 19.02.20
7	Анализ требований завершен	0 дней	Ср 19.02.20	Ср 19.02.20
8	Выбор средств реализации проекта	8 дней	Чт 20.02.20	Пн 02.03.20
9	Выбор платформы	2 дней	Чт 20.02.20	Пт 21.02.20
10	Анализ возможных программных средств	4 дней	Пн 24.02.20	Чт 27.02.20
11	Выбор программного средства	2 дней	Пт 28.02.20	Пн 02.03.20
12	Выбор средств реализации завершен	0 дней	Пн 02.03.20	Пн 02.03.20
13	Реализация проекта	37 дней	Вт 03.03.20	Ср 22.04.20
14	Выбор архитектуры ПО	2 дней	Вт 03.03.20	Ср 04.03.20
15	Проектирование пользовательского интерфейса	5 дней	Чт 05.03.20	Ср 11.03.20
16	Создание клиентской части	13 дней	Чт 12.03.20	Пн 30.03.20
17	Экран поиска автомобилей	7 дней	Чт 12.03.20	Пт 20.03.20
18	Экран добавления объявления	4 дней	Пн 23.03.20	Чт 26.03.20
19	Экран избранного	2 дней	Пт 27.03.20	Пн 30.03.20
20	Проектирование базы данных	9 дней	Чт 05.03.20	Вт 17.03.20
21	Концептуальное проектирование БД	4 дней	Чт 05.03.20	Вт 10.03.20
22	Логическое проектирование БД	2 дней	Ср 11.03.20	Чт 12.03.20
23	Физическое проектирование БД	2 дней	Пт 13.03.20	Пн 16.03.20
24	Создание триггеров и хранимых процедур	1 день	Вт 17.03.20	Вт 17.03.20
25	Создание серверной части	26 дней	Ср 18.03.20	Ср 22.04.20
26	Программирование взаимодействия с БД	6 дней	Ср 18.03.20	Ср 25.03.20
27	Программирование функций пользователя	14 дней	Вт 31.03.20	Пт 17.04.20
28	Программирование сервиса регистрации	3 дней	Пн 20.04.20	Ср 22.04.20
29	Реализация проекта завершена	0 дней	Ср 22.04.20	Ср 22.04.20
30	Отладка	12 дней	Чт 23.04.20	Пт 08.05.20
31	Тестирование проекта	5 дней	Чт 23.04.20	Ср 29.04.20
32	Исправление ошибок	7 дней	Чт 30.04.20	Пт 08.05.20
33	Отладка завершена	0 дней	Пт 08.05.20	Пт 08.05.20
34	Внедрение приложения	5 дней	Пн 11.05.20	Пт 15.05.20
35	Отправка мобильного приложения в GooglePlay	1 день	Пн 11.05.20	Пн 11.05.20
36	Исправление ошибок при внедрении	4 дней	Вт 12.05.20	Пт 15.05.20
37	Внедрение завершено	0 дней	Пт 15.05.20	Пт 15.05.20
38	Конец проекта	0 дней	Пт 15.05.20	Пт 15.05.20

Рисунок 14 – Задачи проекта, введенные в среду Microsoft Project

На рисунке 15 представлена построенная программой диаграмма Ганта.

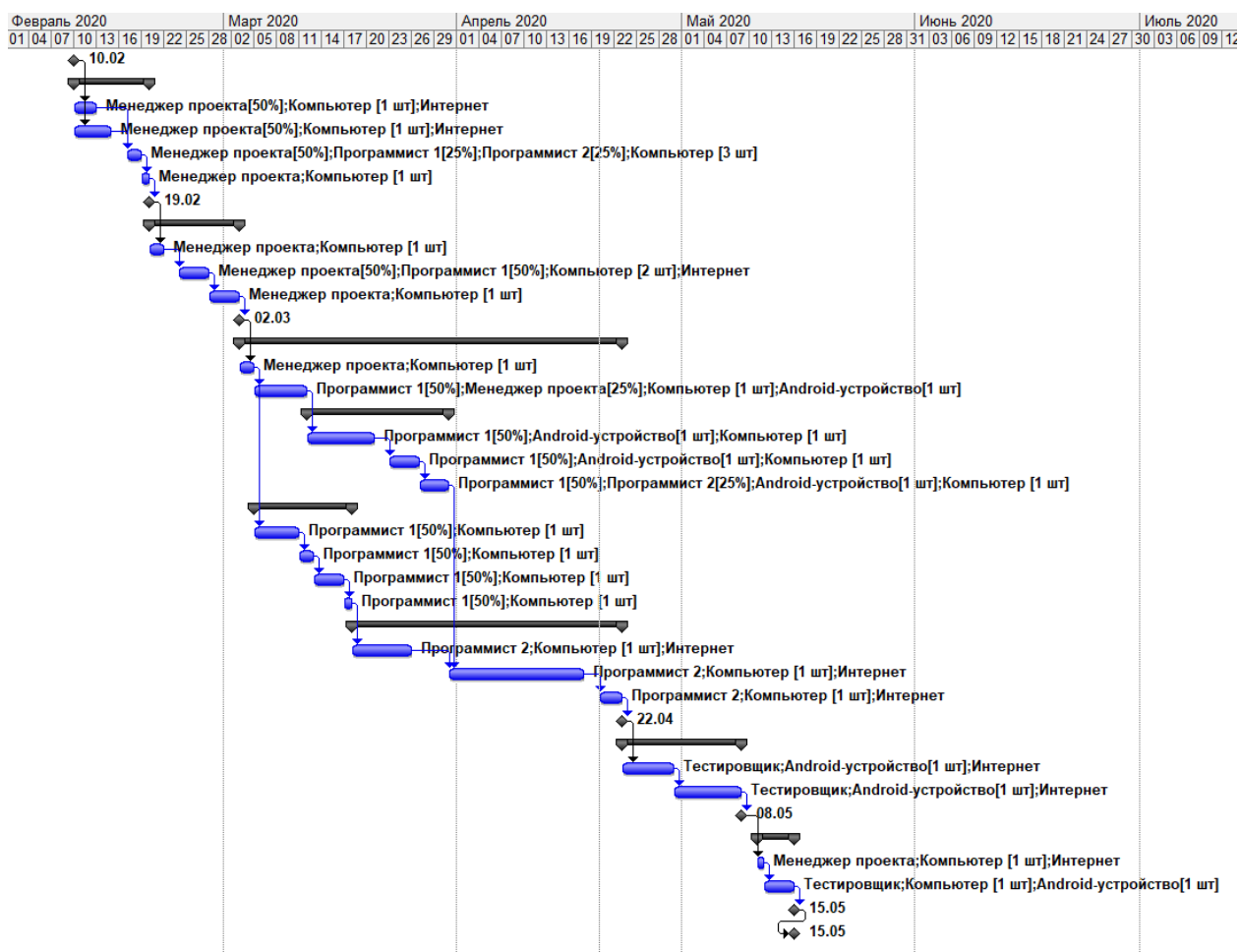


Рисунок 15 – Диаграмма Ганта

На рисунке 16 изображен обзор проекта.

	Начало	Окончание
Текущее	Пн 10.02.20	Пт 15.05.20
Базовое	НД	НД
Фактическое	НД	НД
Отклонение	0д	0д

	Длительность	Трудозатраты	Затраты
Текущие	70д	577,2ч	137 930,00 Р
Базовые	0д?	0ч	0,00 Р
Фактические	0д	0ч	0,00 Р
Оставшиеся	70д	577,2ч	137 930,00 Р

Процент завершения

Длительность: 0%Трудозатраты: 0%

Заккрыть

Рисунок 16 – Обзор проекта

Из рисунка видно подсчитанную в Microsoft Project длительность реализации проекта – 70 дней (что совпадает с вычисленной ранее вручную), и предполагаемые затраты – 137 930 руб.

Также приведем графики ресурсов. На рисунках 17 – 20 представлены графики для трудовых ресурсов.

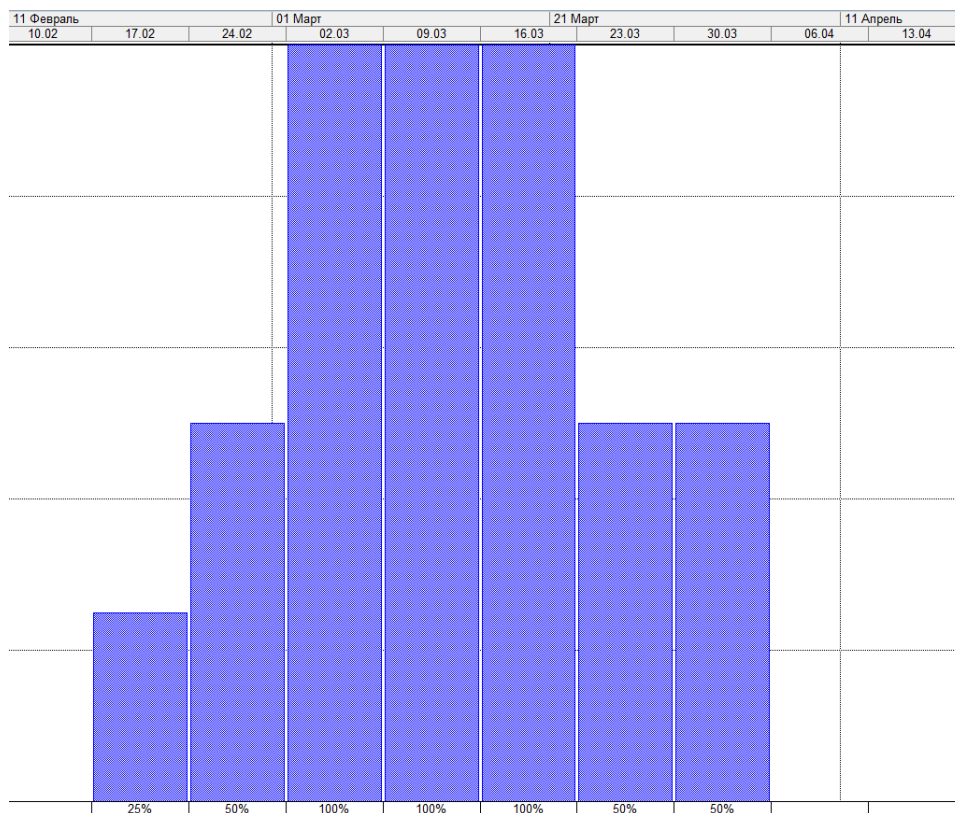


Рисунок 17 – График ресурсов для ресурса «Программист 1»

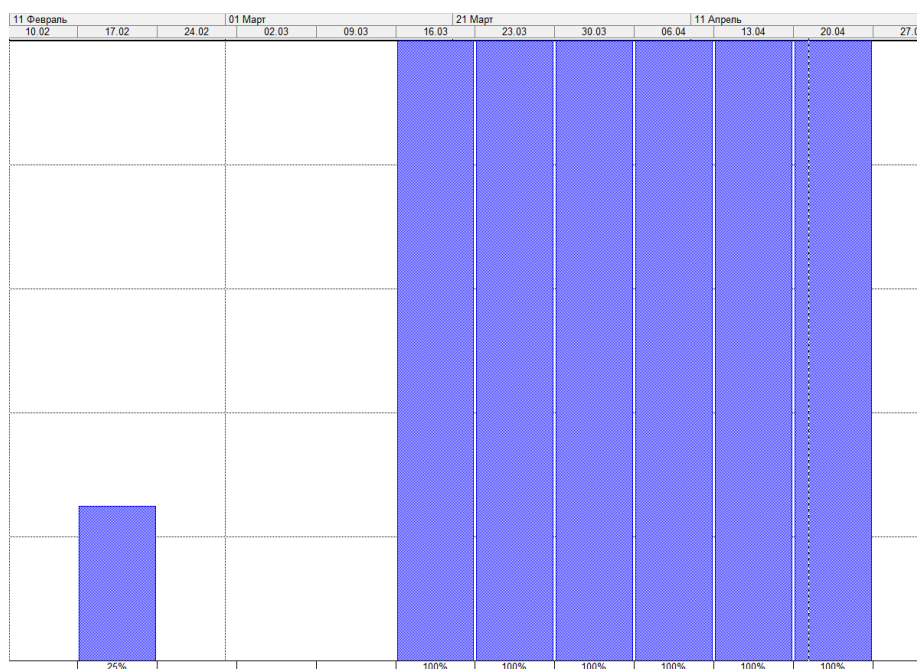


Рисунок 18 – График ресурсов для ресурса «Программист 2»

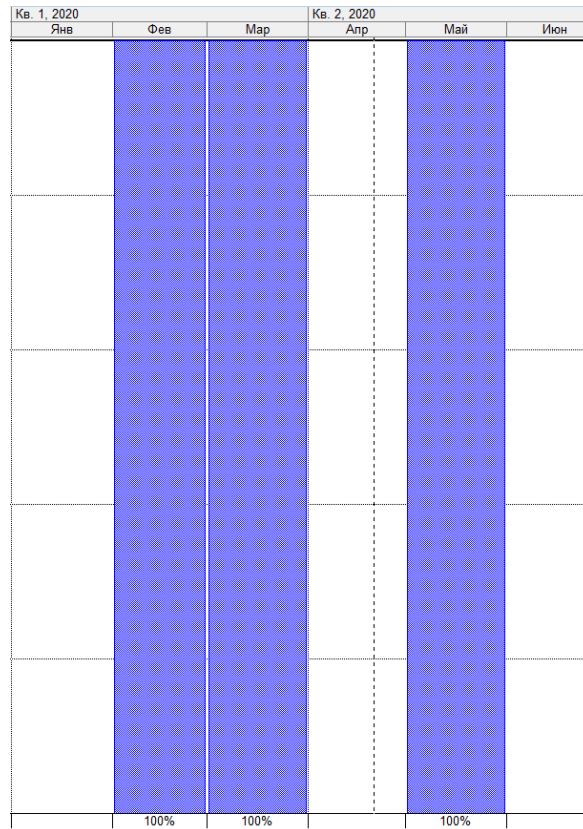


Рисунок 19 – График ресурсов для ресурса «Менеджер проекта»

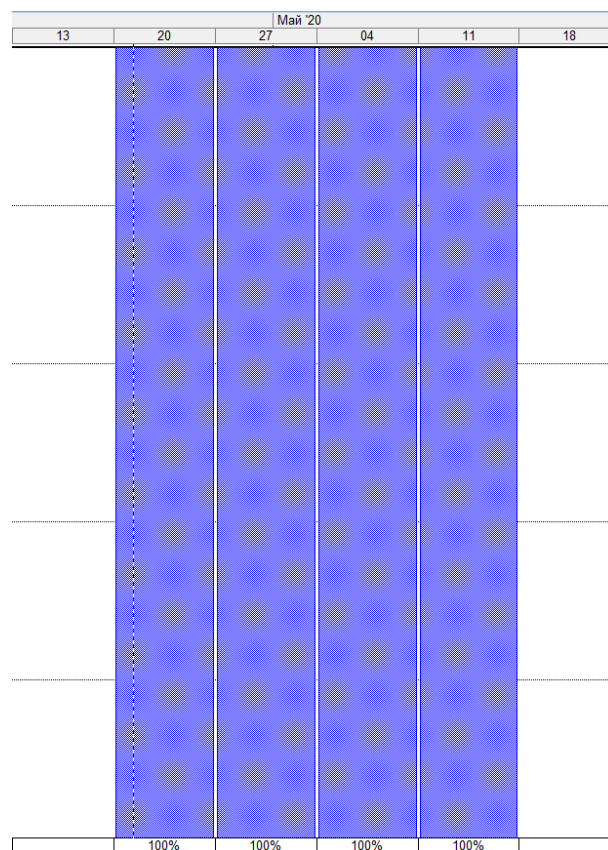


Рисунок 20 – График ресурсов для ресурса «Тестирующий»

На рисунках 21 – 22 изображены графики для материальных ресурсов.

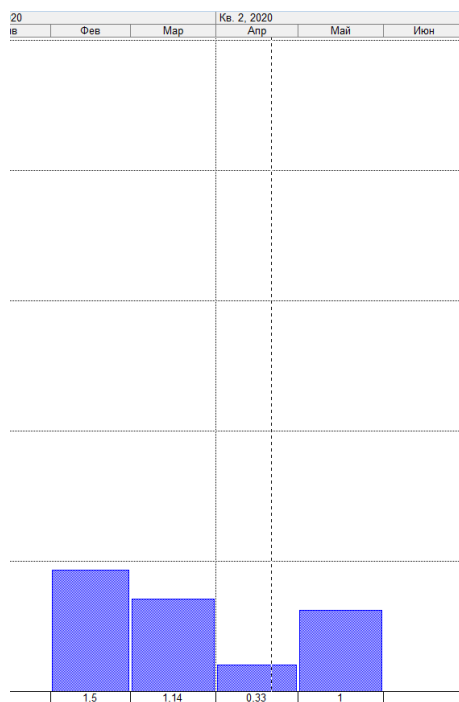


Рисунок 21 – График ресурсов для ресурса «Компьютер»

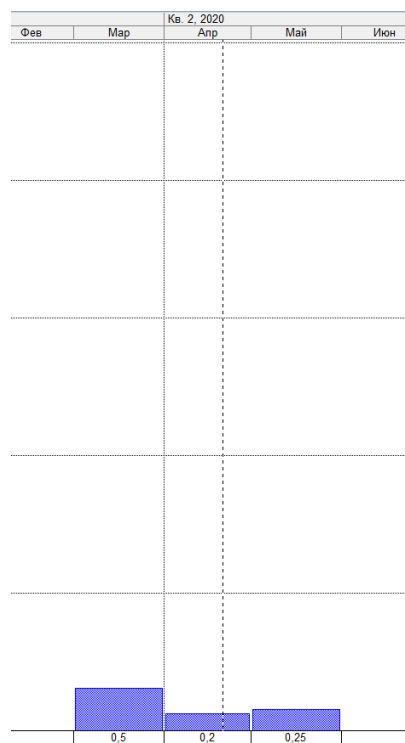


Рисунок 22 – График ресурсов для ресурса «Android-устройство»

Из графиков видно, что ни один ресурс не используется сверх нормы.

2.2 Проектирование пользовательского интерфейса

На основе анализа предметной области и требований к приложению был спроектирован интерфейс пользователя, который включает в себя следующие страницы:

- страница создания аккаунта;
- страница прохождения авторизации;
- страница создания объявления;
- страница профиля;
- страница поиска объявлений.

Рассмотрим подробнее каждую страницу в отдельности. На рисунке 23 изображен интерфейс страницы создания аккаунта. Пользователю необходимо ввести имя, номер телефона, логин и пароль в предложенные поля ввода для создания своего аккаунта.

The image shows a registration form titled "Регистрация" (Registration) on a dark background. The form contains five input fields and a button. Red lines with numbers 1 through 6 point to the following elements:

- 1: Имя (Name) input field
- 2: Номер телефона (Phone number) input field
- 3: Логин (Login) input field
- 4: Пароль (Password) input field
- 5: Повторите пароль (Repeat password) input field
- 6: Создать (Create) button

Рисунок 23 – Страница создания аккаунта

Рассмотрим основные части интерфейса страницы создания аккаунта:

- 1 – поле ввода имени;
- 2 – поле ввода номера телефона;
- 3 – поле ввода логина;
- 4 – поле ввода пароля;
- 5 – поле подтверждения пароля;
- 6 – кнопка создания аккаунта.

На рисунке 24 изображен интерфейс страницы прохождения авторизации, которая предоставляет возможность авторизоваться в системе.



Рисунок 24 – Страница прохождения авторизации

Рассмотрим основные части интерфейса страницы прохождения авторизации:

- 1 – поле ввода логина;
- 2 – поле ввода пароля;
- 3 – кнопка прохождения авторизации;

4 – кнопка перехода к странице создания аккаунта.

На рисунке 25 изображена страница создания объявления. Страница содержит поля выбора характеристик автомобиля и панель типа объявления. Также присутствует возможность выбора фотографии транспортного средства.

Рисунок 25 – Страница создания объявления

Рассмотрим основные элементы интерфейса этой страницы:

- 1 – панель выбора типа подаваемого объявления;
- 2 – поле выбора марки автомобиля;
- 3 – поле выбора модели автомобиля;
- 4 – поле выбора года выпуска автомобиля;
- 5 – поле выбора пробега автомобиля;
- 6 – поле выбора типа кузова автомобиля;
- 7 – поле выбора привода автомобиля;
- 8 – поле выбора объема двигателя автомобиля;

- 9 – поле выбора стоимости автомобиля;
- 10 – кнопка загрузки фотографии автомобиля;
- 11 – кнопка создания объявления.

На рисунке 26 изображена страница профиля пользователя приложения, на ней содержится информация о данных владельца пользователя: имя, номер телефона, избранные и созданные объявления.

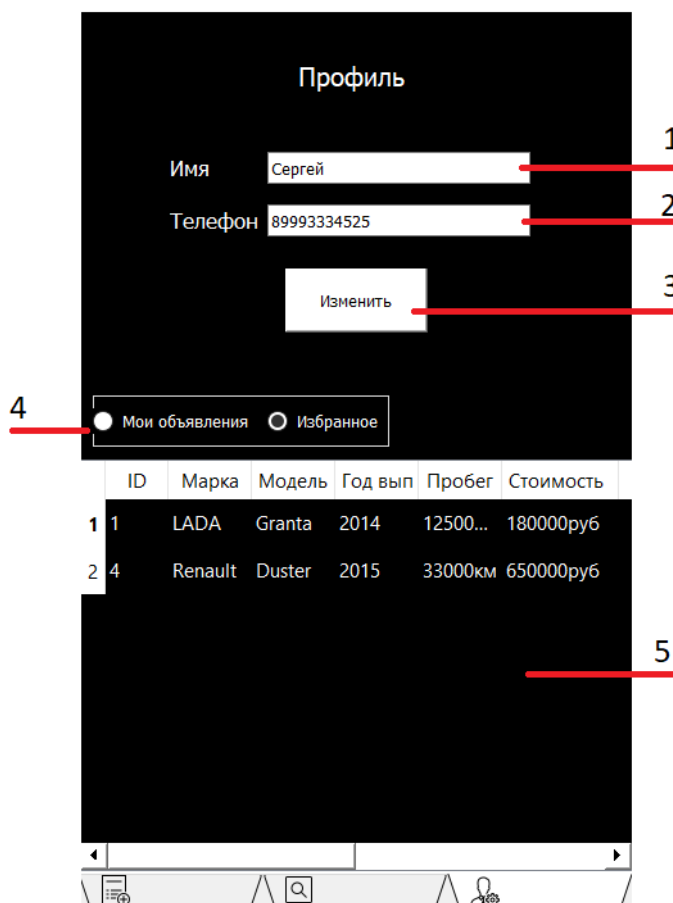


Рисунок 26 – Страница профиля

Рассмотрим основные части интерфейса страницы профиля пользователя:

- 1 – поле, содержащее имя пользователя;
- 2 – поле, содержащее номер телефона пользователя;
- 3 – кнопка для изменения имени и номера телефона;
- 4 – панель выбора типа объявлений;
- 5 – список объявлений.

На рисунке 27 изображена страница поиска объявлений других пользователей.

	ID	Год выпуска	Пробег	Цена
1	5	2015	33000	645000
2	7	2016	108000	680000
3	11	2015	230000	500000
4	21	2018	29000	750000

Рисунок 27 – Страница поиска объявлений

Рассмотрим элементы интерфейса страницы поиска объявлений:

- 1 – панель выбора типа объявления;
- 2 – панель выбора автомобиля;
- 3 – поле выбора типа кузова;
- 4 – поле выбора привода автомобиля;
- 5 – поле выбора объема двигателя автомобиля;
- 6 – диапазон выбора пробега автомобиля;
- 7 – диапазон выбора года выпуска автомобиля;
- 8 – диапазон – выбора стоимости автомобиля;
- 9 – кнопка поиска объявлений по выбранным критериям;
- 10 – список найденных объявлений.

Таким образом, были рассмотрены основные части интерфейса всех страниц приложения.

2.3 Проектирование архитектуры приложения

Мобильное приложение представлено клиент-серверной архитектурой (рисунок 28). Такая архитектура определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций и клиенты (потребители этих функций) [16].

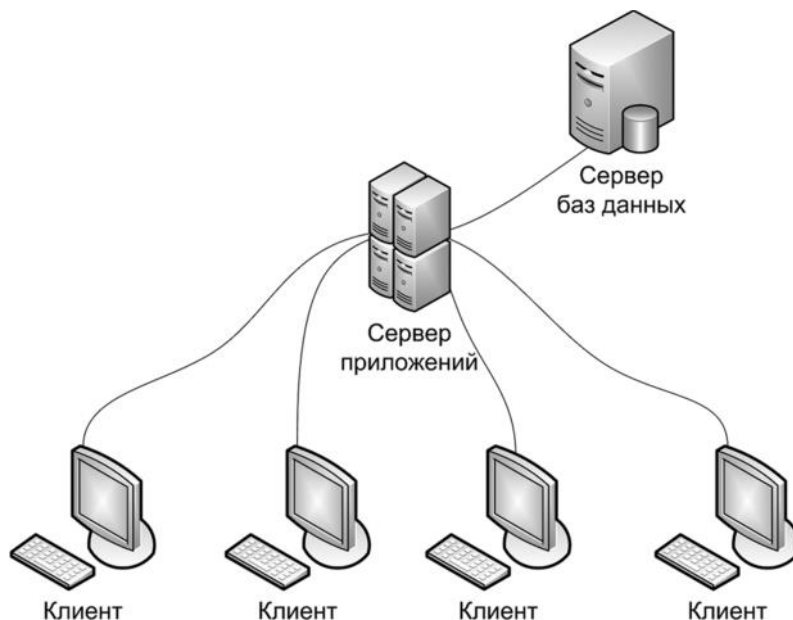


Рисунок 28 – Клиент-серверная архитектура

Клиентская часть использует данные с сервера и предоставляет их пользователю через графический интерфейс. Одновременно несколько клиентов могут взаимодействовать с сервером, клиент представлен мобильным приложением, но в будущем планируется использование еще и веб-сайта.

Взаимодействие осуществляется при помощи протокола TCP, который обеспечивает надежную доставку [17]. Клиент формирует запрос серверу и отправляет его. Сервер на основании полученного запроса генерирует SQL-запрос к серверу базы данных. После его выполнения сервер получает результат выполнения и передает полученные данные клиенту. Клиент, в свою очередь, предоставляет пользователю информацию в удобном виде.

2.4 Логическое проектирование базы данных

На основе анализа предметной области и функциональных требований была построена исходная концептуальная модель. Были выделены типы сущностей, их свойства и типы связей между сущностями.

В таблице 5 представлено описание типов сущностей.

Таблица 5 – Типы сущностей

№	Наименование	Описание	Категория
1	Марка	Список марок автомобилей	Справочник
2	Модель	Список моделей автомобилей для каждой марки	Стержневая
3	Пользователь	Список пользователей	Стержневая
4	Объявление	Список поданных объявлений	Стержневая
5	Избранное	Список объявлений, добавленных в избранное	Стержневая

В таблице 6 представлено описание видов связей между типами сущностей предметной области.

Таблица 6 – Виды связей между типами сущностей

№	Тип сущности 1	Тип сущности 2	Описание	Вид связи	Обязательность
1	Марка	Модель	Модели соответствуют марка	1 : M	Обязательно
2	Пользователь	Объявление	Объявлению соответствует пользователь	1 : M	Обязательно
3	Модель	Объявление	Объявлению соответствует модель	1 : M	Обязательно
4	Пользователь	Избранное	Избранному соответствует пользователь	1 : M	Обязательно
5	Объявление	Избранное	Избранному соответствует Объявление	1 : M	Обязательно

Концептуальная модель предметной области представлена в нотации IDEF1 на рисунке 29.

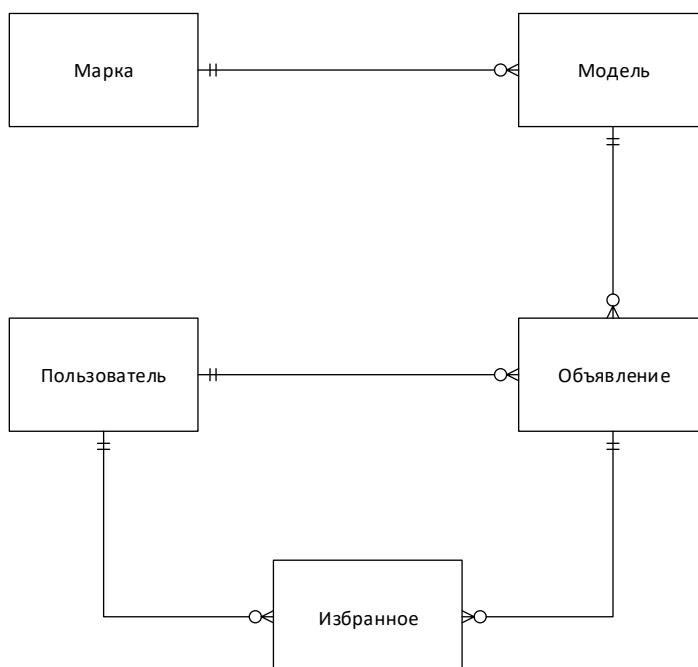


Рисунок 29 – Исходная концептуальная модель предметной области

На рисунке изображены взаимосвязи между сущностями системы. Стандарт IDEF1 специально разработан для удобного построения концептуальной схемы предметной области [19].

Логическая модель базы данных представлена на рисунке 30.

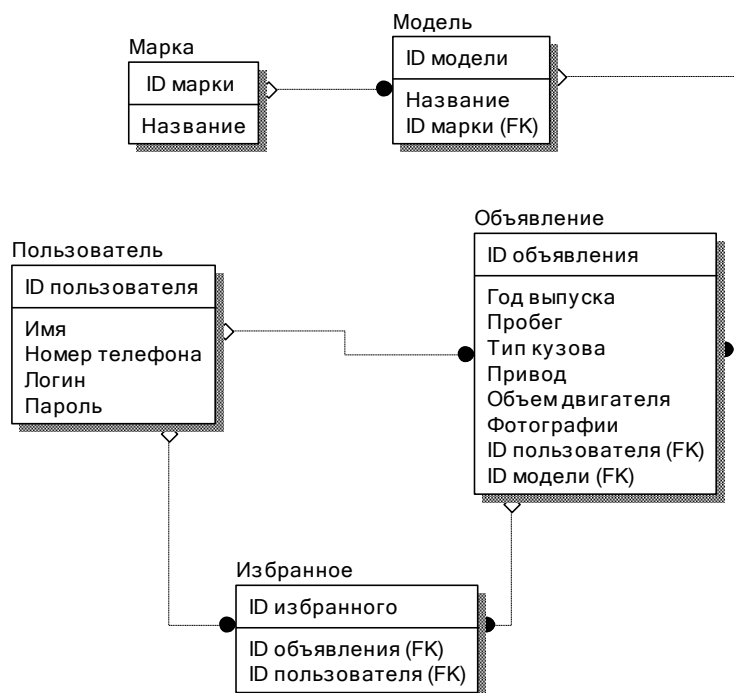


Рисунок 30 – Логическая модель базы данных

Логическое проектирование позволяет отобразить концептуальную модель предметной области в объекты используемой модели данных.

2.5 Проектирование классов клиентской части приложения

В соответствии с моделью предметной области были спроектированы классы, представленные на диаграмме классов (рисунок 31).

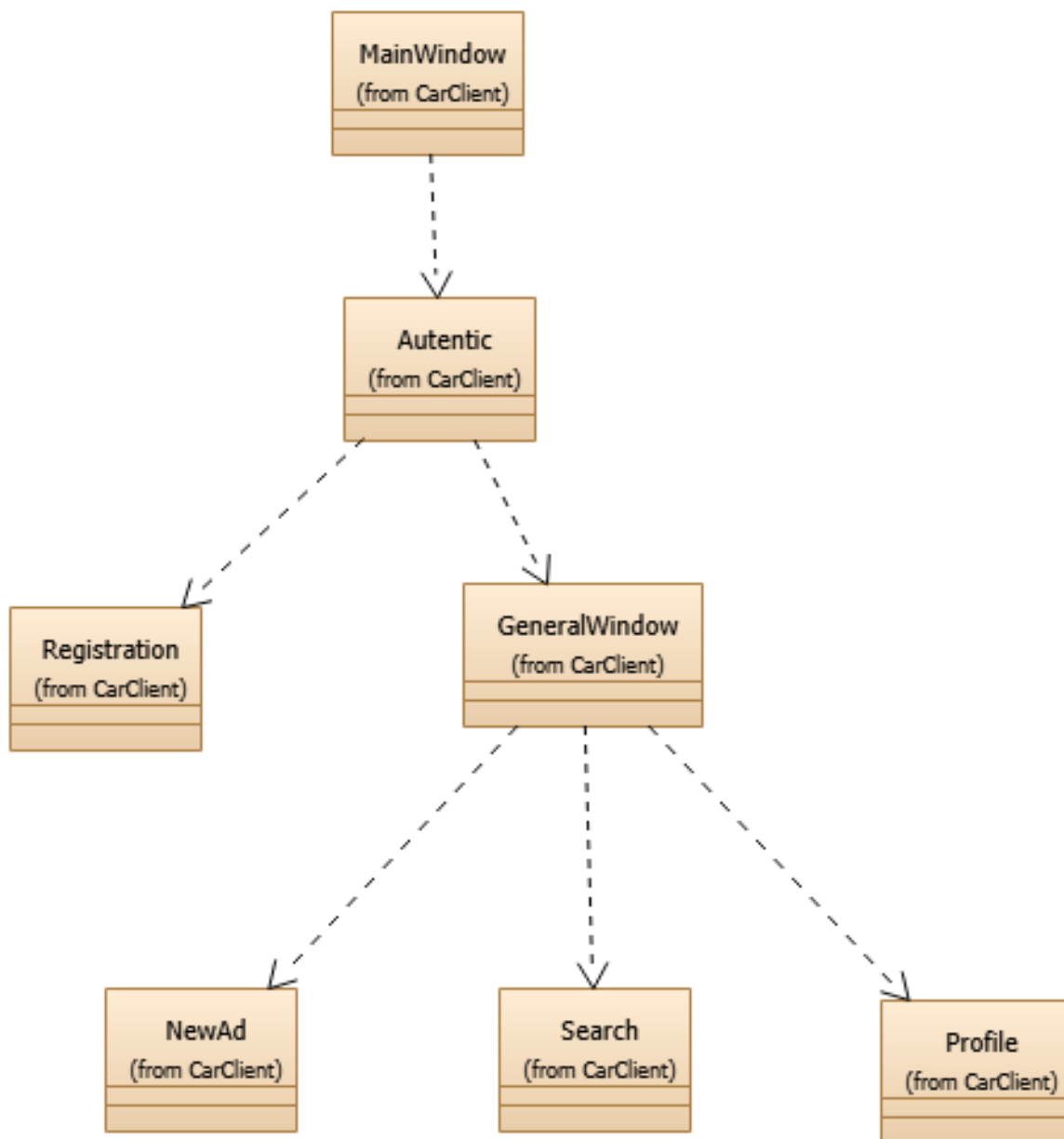


Рисунок 31 – Диаграмма классов клиентского приложения

На диаграмме классов представлено 7 классов. Краткое описание каждого из классов приведено ниже:

- MainWindow – класс, который отвечает за старт приложения;
- Autentic – класс, который отвечает за прохождение авторизации пользователем;

- Registration – класс, который отвечает за создание аккаунта пользователя;
- GeneralWindow – класс, который выводит главный экран приложения;
- NewAd – класс, который отвечает за создание объявлений;
- Search – класс, который отвечает за поиск объявлений;
- Profile – класс, который отвечает за просмотр профиля пользователя.

Листинг некоторых классов представлен в приложении Б.

2.6 Проектирование классов серверной части приложения

Серверная часть приложения представлена одним классом MainWindow, спецификация которого представлена на рисунке 32.

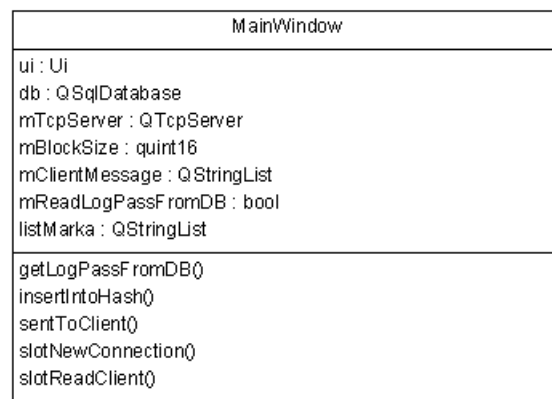


Рисунок 32 – Спецификация класса MainWindow

Этот класс выполняет прием сообщений от клиента и отправляет запросы к базе данных, после выполнения которых отправляет ответ клиенту. Для обеспечения этих возможностей в классе присутствуют:

- слот «slotReadClient()», выполняющий чтение данных от клиентского приложения и создание SQL-запроса к базе данных;
- виртуальный метод «sentToClient()», который отправляет данные клиентскому приложению.

Листинг класса приведен в приложении Б.

3 Реализация мобильного приложения

3.1 Реализация базы данных

Реализация базы данных проводилась при помощи ERwin Data Modeler на основе спроектированной логической модели базы данных. Полученная физическая модель представлена на рисунке 33.

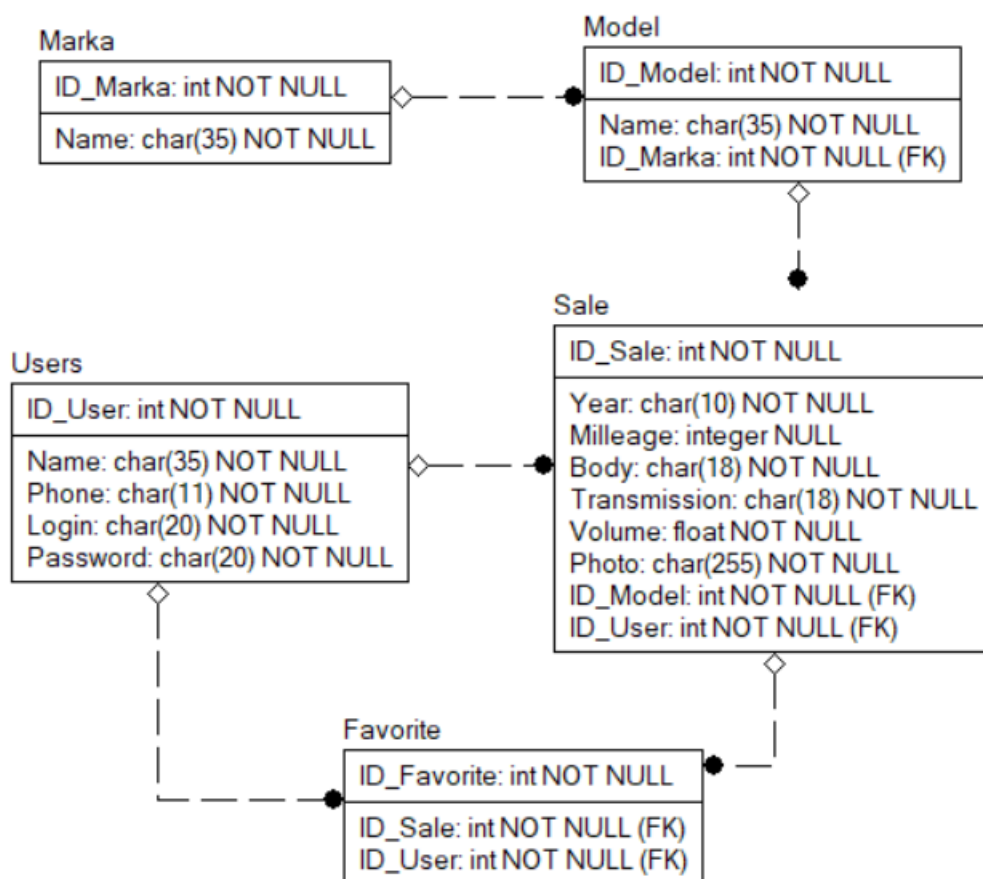


Рисунок 33 – Физическая модель базы данных

Исходный код создания базы данных приведен в приложении В.

В результате было выполнено развертывание базы данных путем запуска SQL-скрипта в СУБД MS SQL Server.

3.2 Реализация клиентской и серверной частей приложения

Разработка мобильного приложения осуществлялась в среде Qt-creator при использовании фреймворка Qt на языке C++.

На рисунке 34 приведена диаграмма компонентов приложения, отражающая общую структуру программы.

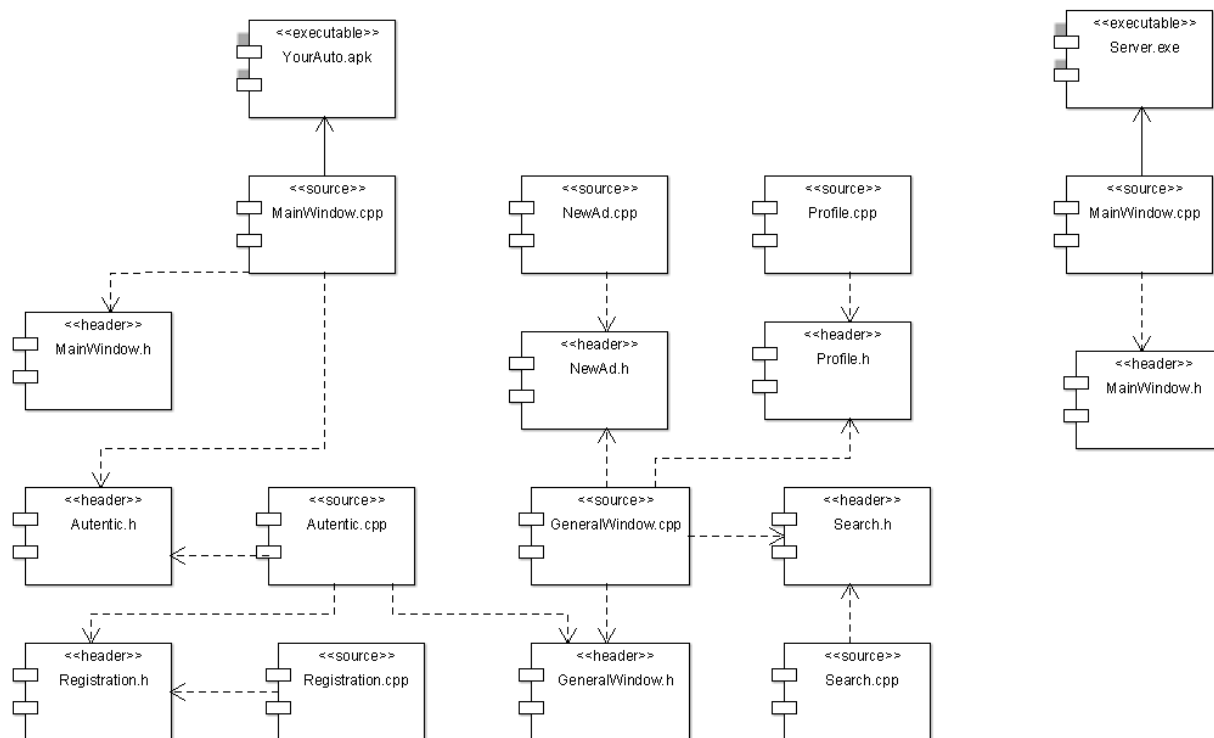


Рисунок 34 – Диаграмма компонентов

Для отображения взаимодействия составных частей системы приведем диаграмму развертывания (рисунок 35).

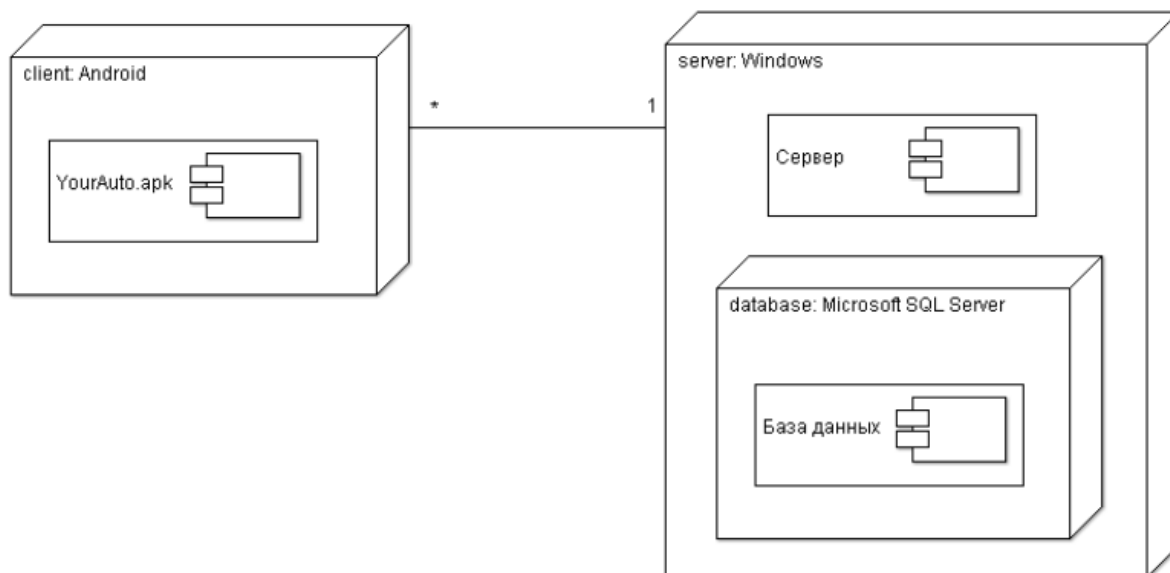


Рисунок 35 – Диаграмма развертывания

Согласно рисунку, клиентская часть приложения развернута на мобильном устройстве с операционной системой Android, а серверная часть и база данных располагаются на отдельном компьютере, который функционирует под управлением операционной системы Windows 10.

4 Контроль качества программного продукта

Важной частью жизненного цикла любого программного обеспечения является его тестирование, главная цель которого не только проверка написанного на более ранних этапах производства кода на наличие ошибок, но и проверка соответствия созданного продукта предъявленным к нему требованиям. Поэтому представленный в этой работе программный продукт тоже был подвергнут этому процессу. Для примера было приведено тестирование одной из лабораторных установок – математического маятника.

4.1 Функциональное тестирование

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и ее ожидаемым поведением на конечном наборе тестов, выбранных определенным образом.

4.1.1 Метод эквивалентного разбиения

В таблице 7 приведены классы эквивалентности для поля mLogin – логин учетной записи пользователя.

Таблица 7 – Классы эквивалентности

Входные данные	Класс эквивалентности	Полученный результат
Пустое значение	Недопустимая длина	Сообщение об ошибке (рисунок Г.1)
Значение длиной менее 5 символов	Недопустимая длина	Сообщение об ошибке (рисунок Г.2)
Значение длиной более 20 символов	Недопустимая длина	Сообщение об ошибке (рисунок Г.3)
Значение длиной менее 20 символов и более 5 символов	Допустимая длина	Авторизация пройдена, открылось окно поиска автомобилей (рисунок Г.4)
Кириллица	Некорректный ввод	Сообщение об ошибке (рисунок Г.5)
Символы, не принадлежащие латинскому алфавиту		
Символы латинского алфавита	Корректный ввод	Авторизация пройдена (рисунок Г.6)

Исходя из данной таблицы, можно сделать вывод, что для поля ввода логина существуют допустимые и недопустимые классы эквивалентности. Для успешного ввода необходимо сочетание всех допустимых вариантов.

По итогам тестирования полученные результаты соответствуют ожидаемым. Результаты тестирования приведены в приложении Г.

4.1.2 Метод границ

Рассмотрим данный метод тестирования на примере поля ввода «Стоимость» при создании объявления. Диапазон значений приведен на рисунке 36.



Рисунок 36 – Диапазон значений поля ввода «Цена»

Зеленым цветом обозначена область корректных, а красным – области некорректных значений. Следовательно, из рисунка видно, что область допустимых значений находится в промежутке от 0 до 99999999 включительно. В таблице 8 приведены типы границ для данного поля.

Таблица 8 – Границы поля ввода «Цена»

Тип границ	Входные данные	Выходные данные	Полученный результат
Произвольные (значение поля в диапазоне [0..99999999])	-1	Недопустимый ввод	Не удалось ввести значение (рисунок Г.7)
	0	Поле приняло значение 0	Ввод данных (рисунок Г.8)
	99999999	Поле приняло значение 99999999	Ввод данных (рисунок Г.9)

Продолжение таблицы 8

Тип границ	Входные данные	Выходные данные	Полученный результат
Произвольные (значение поля в диапазоне [0..9999999])	10000000	Недопустимый ввод	Не удалось ввести значение (рисунок Г.10)
Физические (количество знаков от 1 до 7)	-	Ошибка	Сообщение об ошибке (рисунок Г.11)
	8	Недопустимый ввод	Не удалось ввести значение (рисунок Г.12)
Технологические (выход за пределы int)	-2147483648	Недопустимый ввод	Не удалось ввести значение (рисунок Г.13)
	2147483647	Недопустимый ввод	Не удалось ввести значение (рисунок Г.14)

Приведенные границы можно разделить на 3 группы: произвольные, физические и технологические. Для всех видов границ приложение должно проверять выход за их пределы и выдавать сообщение об ошибке в случае некорректного ввода.

4.2 Метрики кода

Приведенная в листинге 1 функция предназначена для определения корректности заполненных пользователем полей ввода логина и пароля при авторизации. Она проверяет минимальную и максимальную длину полей. При ошибочных данных выдается сообщение, описывающее конкретную ошибку. Если данные введены верно, значения полей передаются серверу.

Листинг 1 – определения корректности логина и пароля

```

void Autentic::slotCheckLogPas()
{
    int minLen = 5;
    int maxLen = 20;
    int logCurrentLen = ui->lineEditLogin->text().count();
    int pasCurrentLen = ui->lineEditPassword->text().count();
    QStringList messageList;
    messageList << "Ошибка!" << "Логин" << "Пароль" << " не может быть
пустым!"
                                << " не должен быть короче 5 символов! " << " не может
превышать 20 символов!";
    bool checkLog;
    bool checkPass;

    if(logCurrentLen == 0 || logCurrentLen < minLen || logCurrentLen >
maxLen)
    {
        if(logCurrentLen == 0)
            QMessageBox::warning(this,messageList[0],messageList[1] +
messageList[3],QMessageBox::Ok);
        if(logCurrentLen < minLen)
            QMessageBox::warning(this,messageList[0],messageList[1] +
messageList[4],QMessageBox::Ok);
        if(logCurrentLen > maxLen)
            QMessageBox::warning(this,messageList[0],messageList[1] +
messageList[5],QMessageBox::Ok);
        checkLog = false;
    }
    if(pasCurrentLen == 0 || pasCurrentLen < minLen || pasCurrentLen >
maxLen)
    {
        if(pasCurrentLen == 0)
            QMessageBox::warning(this,messageList[0],messageList[2] +
messageList[3],QMessageBox::Ok);
        if(pasCurrentLen < minLen)
            QMessageBox::warning(this,messageList[0],messageList[2] +
messageList[4],QMessageBox::Ok);
        if(pasCurrentLen > maxLen)
            QMessageBox::warning(this,messageList[0],messageList[2] +
messageList[5],QMessageBox::Ok);
        checkPass = false;
    }
    else
        checkPass = true;

    if(checkLog && checkPass)
        emit sigCorrectInput();
}

```

Данная функция проверяет логин и пароль на выход за диапазон минимальных и максимальных значений.

4.2.1 Метрика Холстеда

Входными данными для метрик Холстеда являются уникальные операторы и операнды из листинга 1, представленные в таблицах 9 и 10.

Таблица 9 – Уникальные операторы

Номер	Оператор
1	=
2	.
3	->
4	<<
5	==
6	
7	<
8	>
9	&&
10	[]
11	::
12	if
13	else
14	Text()
15	Count()

Таблица 10 – Уникальные операнды

Номер	Операнд
1	minLen
2	maxLen
3	logCurrentLen
4	pasCurrentLen
5	messageList
6	checkLog
7	checkPass

Метрики Холстеда основаны на следующих показателях:

– n1 – число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);

- $n2$ – число уникальных операндов программы (словарь операндов);
- $N1$ – общее число операторов в программе;
- $N2$ – общее число операндов в программе;
- $n1'$ – теоретическое число уникальных операторов;
- $n2'$ – теоретическое число уникальных операндов.

Учитывая введенные обозначения, можно определить словарь программы по формуле:

$$n = n1 + n2.$$

Длина программы рассчитывается по формуле:

$$N = N1 + N2.$$

Теоретический словарь программы:

$$n' = n1' + n2'.$$

Теоретическая длина программы (для стилистически корректных программ отклонение N от N' не превышает 10%):

$$N' = n1 * \log_2 n1 + n2 * \log_2 n2.$$

Объем программы:

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

Уровень качества программирования, основанный лишь на параметрах реальной программы без учета теоретических параметров:

$$L' = \frac{2 * n2}{n1 * N2}.$$

Сложность понимания программы:

$$E_c = \frac{V}{L' * 2}.$$

Результаты расчета перечисленных метрик Холстеда представлены в таблице 11.

Таблица 11 – Расчет метрик Холстеда

Метрика	Значение
$n1$	15
$n2$	7

Продолжение таблицы 11

N1	90
N2	52
n	22
N	142
V	633.3
N'	78.1
L'	0.018
E _C	17591

Полученные результаты сложности показывают, что разработанный код имеет приемлемый уровень качества, что позволит легко поддерживать и изменять его при поддержке проекта.

4.2.2 Цикломатическая метрика Мак-Кейба

Самой распространенной оценкой, основанной на анализе графа, является цикломатическая сложность программы (цикломатическое число Мак-Кейба). Она определяется по формуле:

$$V(G) = e - n + 2p,$$

где e – это количество дуг, n – количество вершин, p – число компонент связности.

На рисунке 37 представлен управляющий граф (граф, описывающий программу в виде вершин-операторов и дуг-переходов) прецедента «Вход пользователя в приложение YourAuto». На его основе будем проводить расчет метрик Мак-Кейба. Граф используется для вычисления цикломатической сложности программы.

Расчет метрик Мак-Кейба на основании графа, приведенного на рисунке 37, представлен в таблице 12.

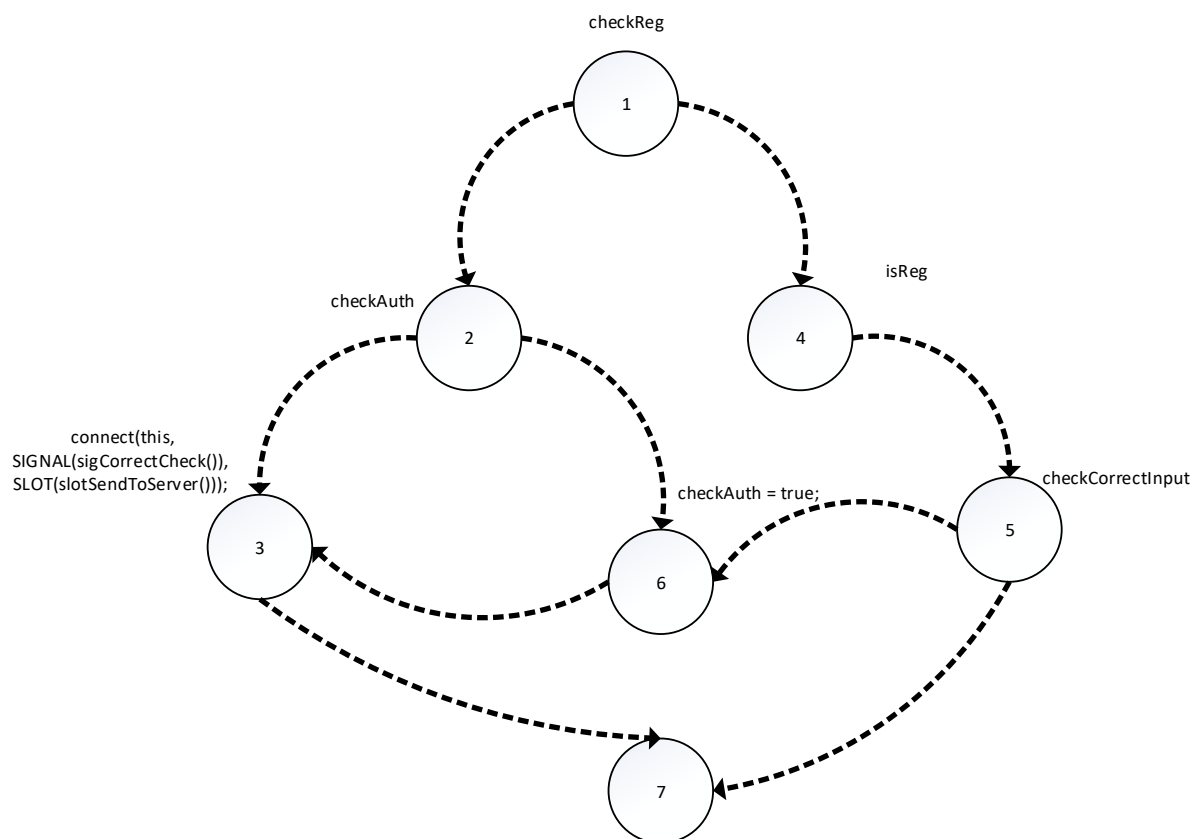


Рисунок 37 – Управляющий граф прецедента «Вход пользователя в приложение YourAvto»

Таблица 12 – Расчет метрик Мак-Кейба

Метрика	Значение
e	9
n	7
p	1
$V(G)$	4

Полученное значение цикломатической метрики меньше 10 и считается приемлемым.

4.2.3 Метрика Чепина

Суть метода состоит в оценке информационной прочности отдельно взятого программного модуля с помощью анализа характера использования переменных из списка ввода-вывода.

Все множество переменных, составляющих список ввода-вывода, разбивается на 4 функциональные группы:

- Р – вводимые переменные для расчетов и для обеспечения вывода;
- М – модифицируемые, или создаваемые внутри программы переменные;
- С – переменные, участвующие в управлении работой программного модуля (управляющие переменные);
- Т – не используемые в программе («паразитные») переменные.

Далее вводится значение метрики Чепина:

$$Q = a_1 * P + a_2 * M + a_3 * C + a_4 * T,$$

где a_1 , a_2 , a_3 , a_4 – весовые коэффициенты. Они использованы для отражения различного влияния на сложность программы каждой функциональной группы. По мнению автора метрики, наибольший вес, равный трем, имеет функциональная группа С, так как она влияет на поток управления программы. Весовые коэффициенты остальных групп распределяются следующим образом: $a_1=1$, $a_2=2$, $a_4=0.5$. Весовой коэффициент группы Т не равен 0, поскольку "паразитные" переменные не увеличивают сложность потока данных программы, но иногда затрудняют ее понимание.

С учетом весовых коэффициентов выражение принимает вид:

$$Q = P + 2M + 3C + 0.5T.$$

Рассчитаем выражение для программы прохождения регистрации и авторизации пользователя. $Q = 5 + 4 + 3 + 0 = 12$.

На основе полученных значений метрики Чепина уровень сложности данной программы считаем низким, как так количество переменных малое и не затрудняет понимание программы.

Заключение

В рамках выполнения выпускной квалификационной работы было разработано мобильное приложение для сервиса покупки и продажи автомобилей.

После рассмотрения предметной области и постановки задач был проведен сравнительный анализ аналогичных продуктов и анализ целевой аудитории. На их основе были составлены требования к разрабатываемой системе. Была составлена диаграмма вариантов использования, отображающая функциональные возможности мобильного приложения; была приведена подробная спецификация некоторых прецедентов. Для более детального рассмотрения логики вариантов использования были составлены диаграммы деятельности и последовательности. По результатам анализа функциональных требований были определены функции, которые должны быть реализованы в приложении.

На этапе проектирования было осуществлено календарное планирование разработки приложения и оценка бюджета, которое показало, что длительность реализации проекта составляет 70 дней и на его реализацию следует затратить 137 930 руб. Также на этом этапе были рассмотрены страницы пользовательского интерфейса, была приведена диаграмма классов для клиентской части и рассмотрена спецификация класса серверной части приложения.

Реализация мобильного приложения проводилась на языке программирования C++ с использованием библиотек Qt в среде программирования Qt Creator 4.11.0. Приложение функционирует на мобильных устройствах с операционной системой Android 9.0 и выше.

Тестирование клиентской части приложения проводилось методами функционального тестирования. Также были просчитаны метрики кода для оценки его качества. Все варианты тестирования были выполнены успешно.

В дальнейшем планируется доработка приложения. Для этого будут улучшены параметры клиентской части и будет создана Web-версия сервиса.

Из всего выше перечисленного, следует, что поставленная задача по реализации мобильного приложения для сервиса покупки и продажи автомобилей была выполнена.

Список использованных источников

1. 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: 01.05.2020).
2. 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://znedu.hostedu.ru/files/ykaz_7668.pdf (дата обращения: 1.05.2020).
3. Просветление. Рынок подержанных автомобилей обеляется и уходит в онлайн. // Forbes [Электронный ресурс]. – URL: <https://www.forbes.ru/biznes/365413-prosvetlenie-rynok-poderzhannyhavtomobiley-obelyaetsya-i-uhodit-v-onlayn> (дата обращения: 03.05.2020).
4. Шматко, А. Д. Анализ проблем автомобильного рынка / А. Д. Шматко, Н. Е. Грицай, Д. В. Мазунин, Д. В. Рахматуллина // Международный журнал гуманитарных и естественных наук. – 2018. – Т. 2. №5. – С. 222 – 225.
5. Продажи через интернет автомобилей с пробегом активно растут. // Ведомости [Электронный ресурс]. – URL: <https://www.vedomosti.ru/auto/articles/2018/05/08/768861-avtomobileiprobegom-internet> (дата обращения: 03.05.2020).
6. Российский рынок легковых автомобилей с пробегом в феврале 2020 года. // Автостат [Электронный ресурс]. – URL: <https://www.autostat.ru/press-releases/43190/> (дата обращения: 10.05.2020).
7. Auto.ru // Википедия [Электронный ресурс]. – URL: <https://ru.wikipedia.org/wiki/Auto.ru> (дата обращения: 12.05.2020).
8. Avito // Википедия [Электронный ресурс]. – URL: <https://ru.wikipedia.org/wiki/Авито> (дата обращения: 12.05.2020).

9. Мобильный маркетинг 2018 – 20 статистик, которые нужно знать // Selectivv [Электронный ресурс]. – URL: <https://selectivv.com/ru/мобильныймаркетинг-2018-20-статистик-ко/> (дата обращения: 15.05.2020).
10. Самые популярные операционные системы мира // marketer [Электронный ресурс]. – URL: <https://marketer.ua/stats-operating-system-2020/> (дата обращения: 15.05.2020).
11. Материалы с сайта «statcounter» [Электронный ресурс]. – URL: <https://gs.statcounter.com/android-version-market-share/mobile-tablet/russianfederation> (дата обращения: 15.05.2020).
12. Анализ требований // sites.google.ru [Электронный ресурс]. – URL: <https://sites.google.com/site/infoprobusinessanalysis/project-definition/processrazrabotki-programmnogo-obespecenia/analiz-trebovanij> (дата обращения: 16.05.2020).
13. Формирование требований и классификация требований// sites.google [Электронный ресурс]. – URL: <https://analytics.infozone.pro/formation-requirements-and-classificationrequirements/#i-10> (дата обращения: 16.05.2020).
14. Арлоу, Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2е издание / Д. Арлоу, И. Нейштадт – Спб.: Символ Плюс, 2007. – 624 с.
15. Богданов, В.В. Управление проектами в Microsoft Project 2007. Учебный курс (+CD) / В.В. Богданов – Спб.: Питер, 2008. – 592 с.
16. Компоненты сетевого приложения. Клиент-серверное взаимодействие и роли серверов. // 4stud.info [Электронный ресурс]. – URL: <http://www.4stud.info/networking/lecture5.html> (дата обращения: 20.05.2020).
17. Протокол TCP — назначение и функционал // ciscotips.ru [Электронный ресурс]. – URL: <http://ciscotips.ru/tcp> (дата обращения: 22.05.2020).

18. Модель «клиент-сервер» // qt-doc.ru [Электронный ресурс]. – URL: <http://qt-doc.ru/model-klient-server.html> (дата обращения: 22.05.2020).
19. Методология. Стандарт IDEF1 // iteranet.ru [Электронный ресурс]. – URL: <http://iteranet.ru/it-novosti/2013/10/16/metodologiya-standart-idef1/> (дата обращения: 22.05.2020).
20. Christian Cote. SQL Server 2017 Integration Services Cookbook / Christian Cote, Matija Lah, Dejan Sarka – Bir. : Packt Publishing, 2017. – 533 с.
21. Функциональное тестирование // qalight.com [Электронный ресурс]. – URL: <https://qalight.com.ua/baza-znaniy/funktsionalnoe-testirovanie/> (дата обращения: 22.05.2020).
22. Рязжков Е. Программный код и его метрики // Habr [Электронный ресурс]. – URL: <https://habr.com/company/intel/blog/106082/> (дата обращения: 24.05.2020)

Приложение А
ГЛОССАРИЙ
(справочное)

Android – операционная система для мобильных устройств, разработанная компанией Google.

UML (Unified Modeling Language) – язык графического описания, с помощью которого осуществляется объектное моделирование в области разработки программного обеспечения, проектирования систем и отображения организационных структур.

SDK (Software Development Kit) – набор средств разработки, позволяющий программистам разрабатывать приложения для определенной платформы.

Qt – фреймворк для разработки программного обеспечения на языке программирования C++.

Qt Creator – разработанная компанией Trolltech свободная среда программирования для разработки на языках C и C++.

Фреймворк – программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Приложение Б
ИСХОДНЫЙ КОД ПРИЛОЖЕНИЯ
(рекомендуемое)

Листинг Б.1 – Autentic.cpp

```

#include "autentic.h"
#include "ui_autentic.h"
#include "QMessageBox"
#include "QTime"
#include "qdebug.h"
#include "qmessagebox.h"

Autentic::Autentic(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Autentic)
{
    ui->setupUi(this);
    mClient = new ClientClass();
    //mGeneralWindow = new GeneralWindow();
    mRegistration = new Registration();
    mCheck = false;

    connect(ui->pushButtonOk, SIGNAL(clicked(bool)), this,
    SLOT(slotCheckLogPas()));
    connect(ui->pushButtonReg, SIGNAL(clicked(bool)), this,
    SLOT(slotToRegistration()));

    connect(mClient->mTcpSocket, SIGNAL(connected()), this,
    SLOT(slotConnected()));
    connect(mClient->mTcpSocket, SIGNAL(readyRead()), this,
    SLOT(slotReadyRead()));
    connect(mClient->mTcpSocket, SIGNAL(error(QAbstractSocket::SocketError)),
    this, SLOT(slotError(QAbstractSocket::SocketError)));

    connect(this, SIGNAL(sigCorrectInput()), this,
    SLOT(slotSendToServer()));
    connect(mRegistration, SIGNAL(sigCorrectInputReg()), this,
    SLOT(slotSendToServer()));
}

Autentic::~Autentic()
{
    delete ui;
}

void Autentic::slotToRegistration()
{
    mRegistration->show();
    mRegistration->slotStart();
}

void Autentic::slotCheckLogPas()
{
    int minLen = 5;
    int maxLen = 20;
    int logCurrentLen = ui->lineEditLogin->text().count();
    int pasCurrentLen = ui->lineEditPassword->text().count();
    QStringList messageList;
    messageList << "Ошибка!" << "Логин" << "Пароль" << " не может быть
пустым!"
    << " не должен быть короче 5 символов!" << " не может
превышать 20 символов!";
    bool checkLog;

```

```

    bool checkPass;

    if(logCurrentLen == 0 || logCurrentLen < minLen || logCurrentLen >
maxLen)
    {
        if(logCurrentLen == 0)
            QMessageBox::warning(this,messageList[0],messageList[1] +
messageList[3],QMessageBox::Ok);
        if(logCurrentLen < minLen && logCurrentLen != 0)
            QMessageBox::warning(this,messageList[0],messageList[1] +
messageList[4],QMessageBox::Ok);
        if(logCurrentLen > maxLen)
            QMessageBox::warning(this,messageList[0],messageList[1] +
messageList[5],QMessageBox::Ok);
        checkLog = false;
    }
    else
    {
        checkLog = true;
    }

    if(pasCurrentLen == 0 || pasCurrentLen < minLen || pasCurrentLen >
maxLen)
    {
        if(pasCurrentLen == 0)
            QMessageBox::warning(this,messageList[0],messageList[2] +
messageList[3],QMessageBox::Ok);
        if(pasCurrentLen < minLen && pasCurrentLen != 0)
            QMessageBox::warning(this,messageList[0],messageList[2] +
messageList[4],QMessageBox::Ok);
        if(pasCurrentLen > maxLen)
            QMessageBox::warning(this,messageList[0],messageList[2] +
messageList[5],QMessageBox::Ok);
        checkPass = false;
    }
    else
    {
        checkPass = true;
    }

    if(checkLog && checkPass)
    {
        mCheck = true;
        emit sigCorrectInput();
    }
}

void Autentic::slotSendToServer()
{
    QStringList outMessage;
    QString strType;
    if(mCheck)
    {
        strType = "autentic";
        outMessage << ui->lineEditLogin->text() << ui->lineEditPassword-
>text();
        mCheck = false;
    }
    if(mRegistration->check)
    {
        strType = "registr";
        outMessage << mRegistration->name << mRegistration->phone <<
mRegistration->login << mRegistration->password;
        mRegistration->check = false;
    }
}

```

```

    QByteArray arrBlock;
    QDataStream out(&arrBlock, QIODevice::WriteOnly);
    out.setVersion(QDataStream::Qt_5_12);
    out << quint16(0) << QTime::currentTime() << strType << outMessage;
    out.device()->seek(0);
    out << quint16(arrBlock.size() - sizeof(quint16));

    mClient->mTcpSocket->write(arrBlock);
    mClient->mTcpSocket->flush();
}

void Autentic::slotReadyRead()
{
    //QTime time;
    //QString typeMessage;

    QDataStream in(mClient->mTcpSocket);
    in.setVersion(QDataStream::Qt_5_12);
    for (;;)
    {
        if (!mBlockSize)
        {
            if (mClient->mTcpSocket->bytesAvailable() < (int)sizeof(quint16))
            {
                break;
            }
            in >> mBlockSize;
        }

        if (mClient->mTcpSocket->bytesAvailable() < mBlockSize)
        {
            break;
        }

        QTime time;
        QString typeMessage, message;
        in >> time >> typeMessage >> message;
        mBlockSize = 0;

        if(typeMessage == "bad autentic")
            QMessageBox::warning(0, "Предупреждение!", "Неверный логин или
пароль", QMessageBox::Ok);

        if(typeMessage == "success autentic")
        {
            mGeneralWindow = new GeneralWindow();
            mGeneralWindow->userID = message.toInt();
            mGeneralWindow->show();
            this->close();
        }

        if(typeMessage == "bad registr")
        {
            QString messageReg;
            in >> messageReg;
            if(messageReg == "error insert")
                QMessageBox::critical(0, "Ошибка на сервере!", "Пожалуйста,
повторите попытку", QMessageBox::Ok);
            if(messageReg == "login is contains")
                QMessageBox::warning(0, "Предупреждение!", "Пользователь с
таким логином уже существует", QMessageBox::Ok);
        }

        if(typeMessage == "success registr")

```

```

        {
            mRegistration->close();
        }
    }
}

void Autentic::slotError(QAbstractSocket::SocketError err)
{
    QString strError =
        "Error: " + (err == QAbstractSocket::HostNotFoundError ?
                     "The host was not found." :
                     err == QAbstractSocket::RemoteHostClosedError ?
                     "The remote host is closed." :
                     err == QAbstractSocket::ConnectionRefusedError ?
                     "The connection was refused." :
                     QString(mClient->mTcpSocket->errorString())
                    );
    mInfo = strError;
    qDebug() << "error aut" << mInfo;
}

// -----
void Autentic::slotConnected()
{
    //mInfo = "Received the connected() signal";
    qDebug() << mInfo;
}

```

Листинг Б.2 – NewAd.cpp

```

#include "newad.h"
#include "ui_newad.h"
#include "QTime"
#include "qdebug.h"
#include "qmessagebox.h"
#include "QTimer"
#include "QFileDialog"

NewAd::NewAd(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::NewAd)
{
    ui->setupUi(this);
    mTcpSocket = new QTcpSocket(nullptr);
    mTcpSocket->connectToHost("DESKTOP-BL07E90", 2123);

    connect(mTcpSocket, SIGNAL(connected()), this, SLOT(slotConnected()));
    connect(mTcpSocket, SIGNAL(readyRead()), this, SLOT(slotReadyRead()));
    connect(mTcpSocket, SIGNAL(error(QAbstractSocket::SocketError)),
            this, SLOT(slotError(QAbstractSocket::SocketError)));
    connect(this, SIGNAL(sigCreate()), this, SLOT(slotSendToServer()));
    connect(ui->pushButtonCreate, SIGNAL(clicked(bool)), this,
            SLOT(slotPushButtonOkClick()));
    connect(ui->pushButtonPhoto, SIGNAL(clicked(bool)), this,
            SLOT(slotPushButtonPhotoClick()));
    connect(ui->comboBoxMarka, SIGNAL(currentIndexChanged(QString)), this,
            SLOT(slotComboBoxMarkaChange()));

    ui->radioButtonSale->setChecked(true);
}

```

```

ui->comboBoxMarka->setCurrentText("Марка");
ui->comboBoxModel->setCurrentText("Модель");
ui->comboBoxYear->setCurrentText("Год выпуска");
ui->comboBoxV->setCurrentText("Объем двигателя");
ui->comboBoxBody->setCurrentText("Тип кузова");
ui->comboBoxTrans->setCurrentText("Привод");

ui->comboBoxModel->setEnabled(false);

for(int i = 2020; i >= 1980; i--)
{
    QString year = QString::number(i);
    ui->comboBoxYear->addItem(year);
}
for(double i = 2.0; i >= 1.0; i = i - 0.2)
{
    QString volume = QString::number(i);
    ui->comboBoxV->addItem(volume);
}

QStringList listBody;
listBody << "Седан" << "Хэтчбек" << "Универсал" << "Лифтбек" << "Пикап"
<< "Кабриолет" << "Внедорожник";
ui->comboBoxBody->addItem(listBody);

QStringList listTrans;
listTrans << "Передний" << "Задний" << "4x4";
ui->comboBoxTrans->addItem(listTrans);
}

NewAd::~NewAd()
{
    delete ui;
}

void NewAd::showEvent(QShowEvent* /*event*/)
{
    slotCreate();
}

void NewAd::slotCreate()
{
    typeOut = false;
    emit sigCreate();
}

void NewAd::slotPushButtonOkClick()
{
    if(ui->comboBoxV->currentIndex() < 0 || ui->comboBoxBody->currentIndex()
< 0
        || ui->comboBoxYear->currentIndex() < 0 || ui->comboBoxMarka-
>currentIndex() < 0
        || ui->comboBoxModel->currentIndex() < 0 || ui->comboBoxTrans-
>currentIndex() < 0)
    {
        QMessageBox::warning(this, "Ошибка!", "Заполните пустые поля!");
    }
    else
    {
        typeOut = true;
        slotSendToServer();
    }
}

```

```

void NewAd::slotPushButtonPhotoClick()
{
    QString str = QFileDialog::getOpenFileName(0, "", "", "*.png");
    QPixmap pixmap;
    pixmap.load(str);
    listPhoto.append(pixmap);
}

void NewAd::slotComboBoxMarkaChange()
{
    ui->comboBoxModel->clear();
    QStringList list;
    list = modelHash.value(ui->comboBoxMarka->currentIndex()+1);
    ui->comboBoxModel->setEnabled(true);
    ui->comboBoxModel->addItem(list);
    qDebug() << modelHash;
}

void NewAd::slotSendToServer()
{
    QString strType;
    QByteArray arrBlock;
    QDataStream out(&arrBlock, QIODevice::WriteOnly);
    out.setVersion(QDataStream::Qt_5_12);

    if(!typeOut)
    {
        strType = "newadStart";
        out << quint16(0) << QTime::currentTime() << strType;
    }
    if(typeOut)
    {
        strType = "newadCreate";
        QStringList listParams;
        QString radio;
        if(ui->radioButtonSale->isChecked())
            radio = "Продажа";
        if(ui->radioButtonBuy->isChecked())
            radio = "Покупка";

        QString userIDint = QString::number(userID);
        listParams << ui->comboBoxModel->currentText() << userIDint
            << ui->comboBoxYear->currentText() << ui->spinBoxKm->
>text()
            << ui->comboBoxBody->currentText() << ui->comboBoxTrans-
>currentText()
            << ui->comboBoxV->currentText() << ui->spinBoxPrice->
>text() << radio;
        out << quint16(0) << QTime::currentTime() << strType << listParams <<
listPhoto;
        listPhoto.clear();
    }

    out.device()->seek(0);
    out << quint16(arrBlock.size() - sizeof(quint16));
    mTcpSocket->write(arrBlock);
    mTcpSocket->flush();
}

void NewAd::slotReadyRead()
{
    QTime time;
    QStringList message;
    QString typeMessage;

```

```

QDataStream in(mTcpSocket);
in.setVersion(QDataStream::Qt_5_12);

in >> mBlockSize;
in >> time >> typeMessage >> markaList >> modelHash;

if(typeMessage == "success newadRead")
{
    ui->comboBoxMarka->addItem(markaList);
}
if(typeMessage == "succes newadCreate")
    QMessageBox::information(this, "Успешно!", "Объявление создано");
}

void NewAd::slotError(QAbstractSocket::SocketError err)
{
    QString strError =
        "Error: " + (err == QAbstractSocket::HostNotFoundError ?
                     "The host was not found." :
                     err == QAbstractSocket::RemoteHostClosedError ?
                     "The remote host is closed." :
                     err == QAbstractSocket::ConnectionRefusedError ?
                     "The connection was refused." :
                     QString(mTcpSocket->errorString())
        );
}

void NewAd::slotConnected()
{
    QString mInfo = "Received the connected() signal";
    qDebug() << mInfo;
}

```

Листинг Б.3 – Registration.cpp

```

#include "registration.h"
#include "ui_registration.h"
#include <QWidget>
#include "QMessageBox"
#include "QTime"
#include "qdebug.h"
#include "qmessagebox.h"

Registration::Registration(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Registration)
{
    ui->setupUi(this);

    //mClient = new ClientClass();
    //mTcpSocket = new QTcpSocket(nullptr);
    // mTcpSocket->connectToHost("DESKTOP-BL07E90", 2123);
    check = false;

    connect(ui->pushButtonOk, SIGNAL(clicked(bool)), this,
        SLOT(slotCheckLogPasReg()));

    //connect(mClient->mTcpSocket, SIGNAL(connected()),
    SLOT(slotConnected()));
    //connect(mTcpSocket, SIGNAL(readyRead()), this, SLOT(slotReadyRead()));
    //connect(mTcpSocket, SIGNAL(error(QAbstractSocket::SocketError)),
    //    this, SLOT(slotError(QAbstractSocket::SocketError)));
}

```

```

        //connect(this, SIGNAL(sigCorrectInputReg()), this,
        SLOT(slotSendToServer()));
    }

    Registration::~Registration()
    {
        delete ui;
    }

    void Registration::slotCheckLogPasReg()
    {
        int minLen = 5;
        int maxLen = 20;
        int logCurrentLen = ui->lineEditLogin->text().count();
        int pasCurrentLen = ui->lineEditPassword->text().count();
        QString pass2 = ui->lineEditPasword2->text();
        QStringList messageList;
        messageList << "Ошибка!" << "Логин" << "Пароль" << " не может быть
        пустым!"
                << " не должен быть короче 5 символов! " << " не может
        превышать 20 символов!";
        bool checkLog;
        bool checkPass;
        bool checkPass2;

        if(logCurrentLen == 0 || logCurrentLen < minLen || logCurrentLen >
        maxLen)
        {
            if(logCurrentLen == 0)
                QMessageBox::warning(this,messageList[0],messageList[1] +
        messageList[3],QMessageBox::Ok);
            if(logCurrentLen < minLen && logCurrentLen != 0)
                QMessageBox::warning(this,messageList[0],messageList[1] +
        messageList[4],QMessageBox::Ok);
            if(logCurrentLen > maxLen)
                QMessageBox::warning(this,messageList[0],messageList[1] +
        messageList[5],QMessageBox::Ok);
            checkLog = false;
        }
        else
            checkLog = true;

        if(pasCurrentLen == 0 || pasCurrentLen < minLen || pasCurrentLen >
        maxLen)
        {
            if(pasCurrentLen == 0)
                QMessageBox::warning(this,messageList[0],messageList[2] +
        messageList[3],QMessageBox::Ok);
            if(pasCurrentLen < minLen && pasCurrentLen != 0)
                QMessageBox::warning(this,messageList[0],messageList[2] +
        messageList[4],QMessageBox::Ok);
            if(pasCurrentLen > maxLen)
                QMessageBox::warning(this,messageList[0],messageList[2] +
        messageList[5],QMessageBox::Ok);
            checkPass = false;
        }
        else
            checkPass = true;

        if(pass2 != ui->lineEditPassword->text())
        {
            QMessageBox::warning(this, messageList[0], "Пароли не совпадают",
        QMessageBox::Ok);
            checkPass2 = false;
        }
    }

```

```

else
    checkPass2 = true;

if(checkLog && checkPass && checkPass2)
{
    check = true;
    this->close();
    login = ui->lineEditLogin->text();
    password = ui->lineEditPassword->text();
    name = ui->lineEditName->text();
    phone = ui->lineEditPhone->text();
    emit sigCorrectInputReg();
}

//emit sigCorrectInputReg();
}

void Registration::slotStart()
{
}

void Registration::slotSendToServer()
{
    QStringList logPass;
    QString strLogPass = "registr";
    logPass << ui->lineEditLogin->text() << ui->lineEditPassword->text();
    QByteArray arrBlock;
    QDataStream out(&arrBlock, QIODevice::WriteOnly);
    out.setVersion(QDataStream::Qt_5_12);
    //out << quint16(0) << QTime::currentTime() << ui->lineEditLogin->text();
    //out << quint16(0) << QTime::currentTime() << ui->lineEditPassword-
>text();
    out << quint16(0) << QTime::currentTime() << strLogPass << logPass;
    out.device()->seek(0);
    out << quint16(arrBlock.size() - sizeof(quint16));

    mTcpSocket->write(arrBlock);
    mTcpSocket->flush();
}

void Registration::slotReadyRead()
{
    QDataStream in(mTcpSocket);
    in.setVersion(QDataStream::Qt_5_12);
    for (;;)
    {
        if (!mBlockSize)
        {
            if (mTcpSocket->bytesAvailable() < (int)sizeof(quint16))
            {
                break;
            }
            in >> mBlockSize;
        }

        if (mTcpSocket->bytesAvailable() < mBlockSize)
        {
            break;
        }
        QTime time;
        QStringList message;
        QString typeMessage;
        in >> time >> typeMessage >> message;
    }
}

```

```

        //mInfo = time.toString() + " " + typeMessage + message;
        // mBlockSize = 0;

        if(typeMessage == "success registr")
        {
            check = true;
            this->close();
        }
        mBlockSize = 0;
    }
}

void Registration::slotError(QAbstractSocket::SocketError err)
{
    QString strError =
        "Error: " + (err == QAbstractSocket::HostNotFoundError ?
                     "The host was not found." :
                     err == QAbstractSocket::RemoteHostClosedError ?
                     "The remote host is closed." :
                     err == QAbstractSocket::ConnectionRefusedError ?
                     "The connection was refused." :
                     QString(mTcpSocket->errorString())
                    );
    mInfo = strError;
}

```

Листинг Б.4 – MainWindow.cpp (клиентская часть)

```

#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    timer = new QTimer();
    timer->setInterval(3000);
    connect(timer, SIGNAL(timeout()), SLOT(slotToAutentic()));
    timer->start();
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::slotToAutentic()
{
    mAutentic = new Autentic();
    mAutentic->show();
    this->close();
    this->destroy(true);
}

```

Листинг Б.5 – MainWindow.cpp (серверная часть)

```

#include "mainwindow.h"
#include "ui_mainwindow.h"

```

```

#include "QDebug"
#include "QSqlError"
#include "QSqlRecord"
#include "qdebug.h"
#include "QMessageBox"
#include "QDir"

MainWindow::MainWindow(int nPort, QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow), mBlockSize(0)
{
    ui->setupUi(this);

    mTcpServer = new QTcpServer(this);
    if (!mTcpServer->listen(QHostAddress::Any, nPort))
    {
        QMessageBox::critical(0, "Server Error", "Unable to start the server:" +
mTcpServer->errorString());
        qDebug() << mTcpServer->serverAddress().toString();
        mTcpServer->close();
        return;
    }
    connect(mTcpServer,
SIGNAL(newConnection()), this, SLOT(slotNewConnection()));

    db = QSqlDatabase::addDatabase("QODBC");
    db.setDatabaseName("DRIVER={SQL Server};SERVER=DESKTOP-
BL07E90;DATABASE=cars;Trusted_Connection=yes;");

    if(db.open())
        qDebug() << "db open";
    else
        qDebug() << db.lastError().text();
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::slotNewConnection()
{
    QTcpSocket* pClientSocket = mTcpServer->nextPendingConnection();
    connect(pClientSocket, SIGNAL(disconnected()), pClientSocket,
SLOT(deleteLater()));
    connect(pClientSocket, SIGNAL(readyRead()), this, SLOT(slotReadClient()));

    QString str = "Connect: ", message = "server is connected!";
    sendToClient(pClientSocket, str, message);
}

// -----
void MainWindow::slotReadClient()
{
    QTcpSocket* pClientSocket = (QTcpSocket*)sender();
    QDataStream in(pClientSocket);
    in.setVersion(QDataStream::Qt_5_12);
    for (;;)
    {
        if (!mBlockSize)
        {
            if (pClientSocket->bytesAvailable() < (int)sizeof(quint16))
            {
                break;
            }
        }
    }
}

```

```

        in >> mBlockSize;
    }

    if (pClientSocket->bytesAvailable() < mBlockSize)
        break;

    QTime    time;
    QString  strType;
    in >> time >> strType >> mClientMessage;

    for(int i = 0; i < mClientMessage.size(); i++)
    {
        QString strMessage = time.toString() + " " + "Client has sent - "
+ mClientMessage[i];
        ui->textEdit->append(strMessage);
    }

    mBlockSize = 0;

    if(strType == "autentic")
    {
        getLogPassFromDB(mClientMessage[0], mClientMessage[1]);
        if(mReadLogPassFromDB)
        {
            //QStringList listParams;
            QString request = "select id from users where login = '" +
mClientMessage[0] + "'";
            QSqlQuery query = QSqlQuery(request);
            while (query.next())
            {
                sendToClient(pClientSocket, "success autentic",
query.value(0).toString());
            }
        }
        else
            sendToClient(pClientSocket, "bad autentic", "");
    }

    if(strType == "registr")
    {
        QSqlQuery query = QSqlQuery("select * from users", db);
        QStringList listLogin;
        QString message;

        while (query.next())
            listLogin << query.value(3).toString();

        if(listLogin.contains(mClientMessage[2]))
            sendToClient(pClientSocket, "bad registr", "login is
contains");
        else
        {
            QSqlQuery queryWrite;
            queryWrite.prepare("insert into users (name, phone, login,
password) "
                                "VALUES (?, ?, ?, ?)");
            queryWrite.addBindValue(mClientMessage[0]);
            queryWrite.addBindValue(mClientMessage[1]);
            queryWrite.addBindValue(mClientMessage[2]);
            queryWrite.addBindValue(mClientMessage[3]);
            queryWrite.exec();

            if(queryWrite.lastError().type() == QSqlError::NoError)
                sendToClient(pClientSocket, "success registr", "");
        }
    }

```

```

        else
            sendToClient(pClientSocket, "bad registr", "error
insert");
    }
}

if(strType == "newadStart")
{
    QList<int> listID;
    QSqlQuery queryModel = QSqlQuery("select * from model", db);
    QSqlQuery queryMarka = QSqlQuery("select * from marka", db);

    while (queryMarka.next())
    {
        listMarka << queryMarka.value(1).toString();
    }

    while(queryModel.next())
    {
        insertIntoHash(queryModel.value(2).toInt(),
queryModel.value(1).toString());
    }
    sendToClient(pClientSocket, "success newadRead", listMarka,
hashModel);
}

if(strType == "newadCreate")
{
    QList<QPixmap> listPhoto;
    in >> listPhoto;
    QString strRequest = "select id from model where name = '" +
mClientMessage[0] + "'";
    int idModel;
    QString photo;

    QSqlQuery queryModel = QSqlQuery(strRequest);
    while(queryModel.next())
    {
        idModel = queryModel.value(0).toInt();
    }

    if(!listPhoto.empty())
        QDir().mkdir(mClientMessage[1]);

    for(int i = 0; i < listPhoto.size(); i++)
    {

    }

}

}

void MainWindow::insertIntoHash(int key, QString value)
{
    QStringList newList;
    newList << value;
    if(!hashModel.empty())
    {
        for(int i = 0; i < hashModel.size(); i++)
        {
            if(hashModel.keys()[i] == key)
            {
                QStringList list = hashModel.value(i);
                list << value;
            }
        }
    }
}

```

```

        hashModel.insert(key, list);
    }
    else
        hashModel.insert(key, newList);
    }
}
else
    hashModel.insert(key, newList);
}

void MainWindow::getLogPassFromDB(QString login, QString password)
{
    QSqlQuery query = QSqlQuery("select * from users", db);
    QStringList listLogin, listPassword;
    bool checkLog;
    bool checkPass;
    while (query.next())
    {
        listLogin << query.value(3).toString();
        listPassword << query.value(4).toString();
    }
    if(listLogin.contains(login))
        checkLog = true;
    else
        checkLog = false;
    if(listPassword.contains(password))
        checkPass = true;
    else
        checkPass = false;
    if(checkLog && checkPass)
        mReadLogPassFromDB = true;
    else
        mReadLogPassFromDB = false;
}

// -----
void MainWindow::sendToClient(QTcpSocket* pSocket, QString strType, QString
message)
{
    QByteArray arrBlock;
    QDataStream out(&arrBlock, QIODevice::WriteOnly);
    out.setVersion(QDataStream::Qt_5_12);
    out << quint16(0) << QTime::currentTime() << strType << message;

    out.device()->seek(0);
    out << quint16(arrBlock.size() - sizeof(quint16));

    pSocket->write(arrBlock);
    pSocket->flush();
}

void MainWindow::sendToClient(QTcpSocket* pSocket, QString strType,
QStringList message)
{
    QByteArray arrBlock;
    QDataStream out(&arrBlock, QIODevice::WriteOnly);
    out.setVersion(QDataStream::Qt_5_12);
    out << quint16(0) << QTime::currentTime() << strType << message;

    out.device()->seek(0);
    out << quint16(arrBlock.size() - sizeof(quint16));

    pSocket->write(arrBlock);
    pSocket->flush();
}

```

```

void MainWindow::sendToClient(QTcpSocket *pSocket, QString strType,
                             QStringList marka, QHash<int, QStringList>
model)
{
    QByteArray arrBlock;
    QDataStream out(&arrBlock, QIODevice::WriteOnly);
    out.setVersion(QDataStream::Qt_5_12);
    out << quint16(0) << QTime::currentTime() << strType << marka << model;

    out.device()->seek(0);
    out << quint16(arrBlock.size() - sizeof(quint16));

    pSocket->write(arrBlock);
    pSocket->flush();
}

```

Приложение В
ИСХОДНЫЙ КОД СОЗДАНИЯ БАЗЫ ДАННЫХ
(рекомендуемое)

Листинг В.1 – Исходный код создания базы данных

```

create table [marka] (
    ID integer IDENTITY(1,1) NOT NULL PRIMARY KEY CHECK(ID>=1 AND ID<500),
    name varchar (255) NOT NULL,
)

create table [model] (
    ID integer IDENTITY(1,1) NOT NULL PRIMARY KEY CHECK(ID>=1 AND ID<5000),
    name varchar (255) NOT NULL,
    ID_marka integer NOT NULL,
    FOREIGN KEY (ID_marka) REFERENCES marka (ID)
)

create table [sale] (
    ID integer IDENTITY(1,1) NOT NULL PRIMARY KEY CHECK(ID>=1 AND
ID<1000000),
    ID_model integer NOT NULL,
    year varchar (255) NOT NULL,
    mileage integer NOT NULL,
    body varchar (255) NOT NULL,
    transmission varchar (255) NOT NULL,
    volume float NOT NULL,
    photo varchar (255) NOT NULL,
    date smalldatetime NOT NULL,
    FOREIGN KEY (ID_model) REFERENCES model (ID)
)

create table [user] (
    ID integer IDENTITY(1,1) NOT NULL PRIMARY KEY CHECK(ID>=1 AND
ID<100000),
    name varchar (255) NOT NULL,
    surname varchar (255),
    favorite varchar (255),
    ID_sale integer,
    FOREIGN KEY (ID_sale) REFERENCES sale (ID)
)

```

Приложение Г
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ
(рекомендуемое)

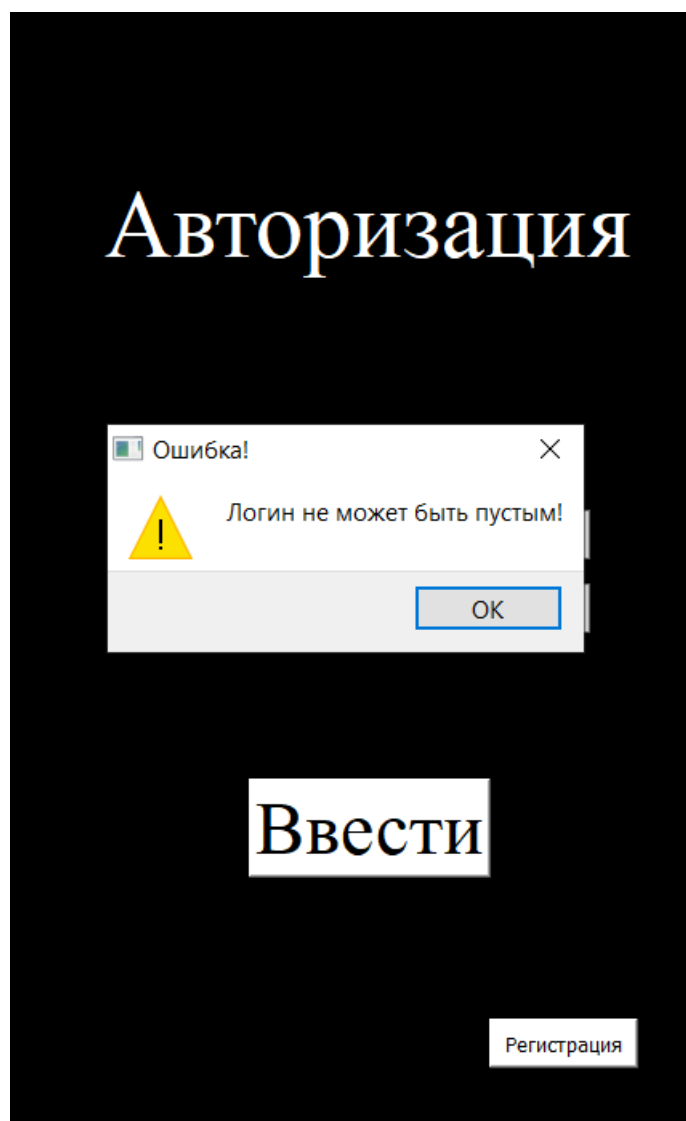


Рисунок Г.1 – Ввод пустого значения в поле логина

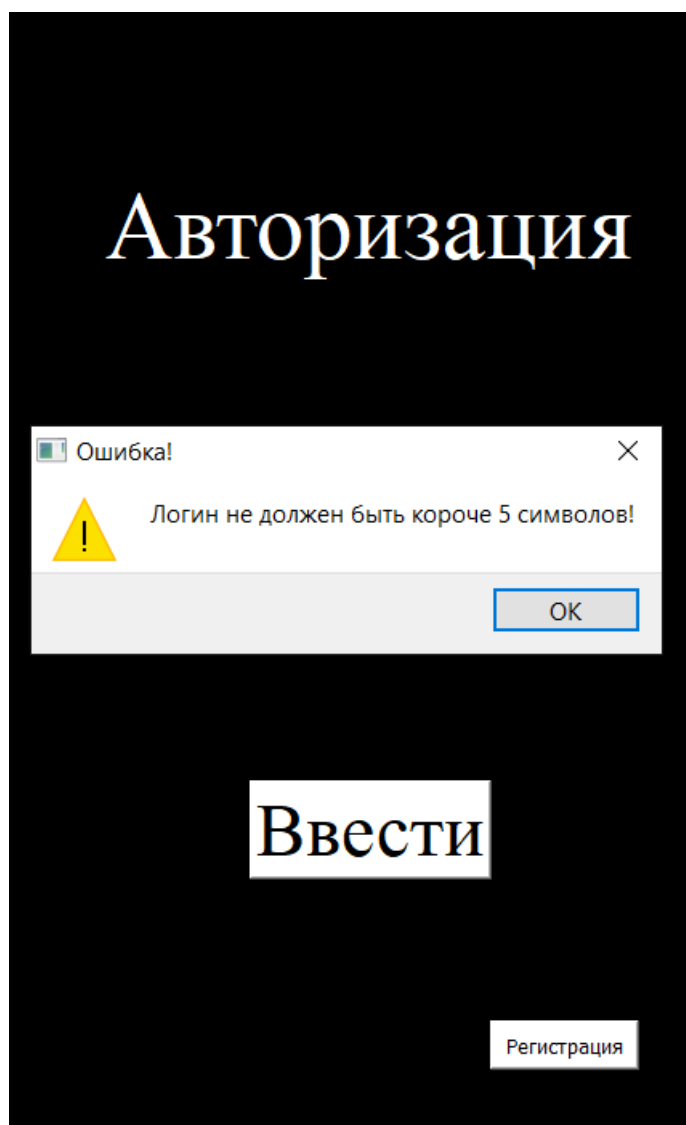


Рисунок Г.2 – Ввод логина длиной менее 5 символов

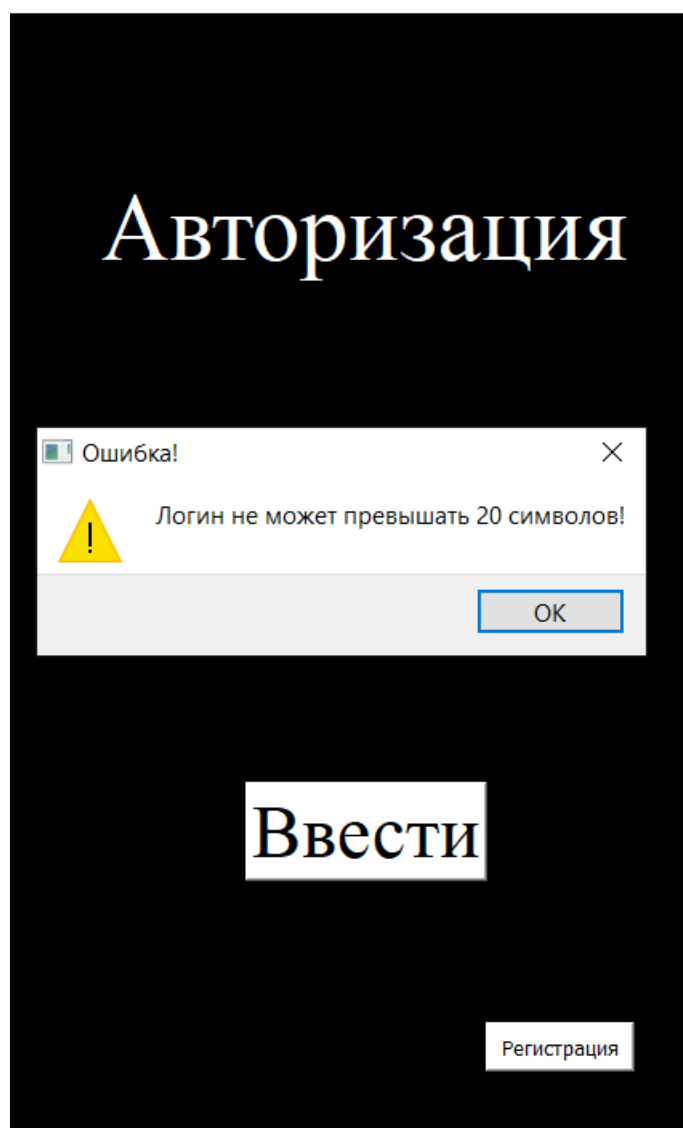


Рисунок Г.3 – Ввод логина длиной более 20 символов

Поиск объявлений

Критерии поиска

☒ Покупка
 ☐ Продажа

Renault

Logan

Пробег

-

Кузов

Хетчбэк

Год выпуска

-

Привод

4x4

Цена

-

Объем двигателя

2.0

Искать

	ID	Год выпуска	Пробег	
1	5	2015	33000	64
2	7	2016	108000	68
3	11	2015	230000	50
4	21	2018	29000	75

Рисунок Г.4 – Ввод логина корректной длины

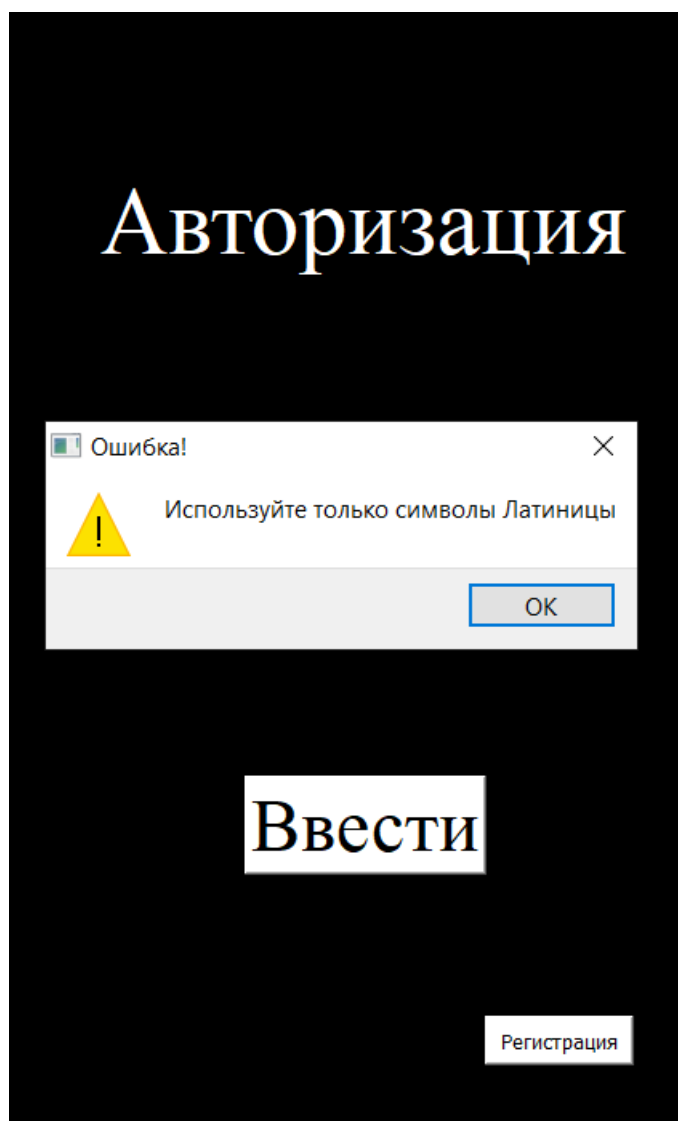


Рисунок Г.5 – Использование символов, не принадлежащих латинице

Поиск объявлений

Критерии поиска

☒ Покупка
 ☐ Продажа

Renault ▾

Logan ▾

Пробег

-

Кузов

Хетчбэк ▾

Год выпуска

-

Привод

4x4 ▾

Цена

-

Объем двигателя

2.0 ▾

Искать

	ID	Год выпуска	Пробег	
1	5	2015	33000	64
2	7	2016	108000	68
3	11	2015	230000	50
4	21	2018	29000	75

Рисунок Г.6 – Ввод логина латинскими символами

Создание объявления

☐ Продажа ☒ Покупка

Марка	<input type="text" value="Ford"/>
Модель	<input type="text" value="Fusion"/>
Год выпуска	<input type="text" value="2014"/>
Пробег	<input type="text" value="100000 км"/>
Кузов	<input type="text" value="Седан"/>
Привод	<input type="text" value="Передний"/>
Объем двигателя	<input type="text" value="1.6"/>
Цена	<input type="text" value="руб"/>

Выбрать фото

Создать объявление

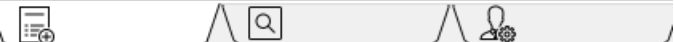


Рисунок Г.7 – попытка ввода значения -1 в поле «Цена»

Создание объявления

☐ Продажа ☒ Покупка

Марка

Модель

Год выпуска

Пробег

Кузов

Привод

Объем двигателя

Цена

Успешно!

 Объявление создано

Рисунок Г.8 – Ввод значения 0 в поле «Цена»

Создание объявления

☐ Продажа ☒ Покупка

Марка

Модель

Год выпуска

Пробег

Кузов

Привод

Объем двигателя

Цена

Успешно!

Объявление создано

OK

Рисунок Г.9 – Ввод значения 999999 в поле «Цена»

Создание объявления

☐ Продажа ☒ Покупка

Марка	<input type="text" value="Honda"/>
Модель	<input type="text" value="Accord"/>
Год выпуска	<input type="text" value="2018"/>
Пробег	<input type="text" value="8000 км"/>
Кузов	<input type="text" value="Седан"/>
Привод	<input type="text" value="Передний"/>
Объем двигателя	<input type="text" value="1.4"/>
Цена	<input type="text" value="1000000 руб"/>

Выбрать фото

Создать объявление



Рисунок Г.10 – Попытка ввода значения 10000000 в поле «Цена»

Создание объявления

☐ Продажа ☒ Покупка

Марка

Модель

Год выпуска

Пробег

Кузов

Привод

Объем двигателя

Цена

Ошибка!

Заполните пустые поля!

OK

Рисунок Г.11 – Ввод пустого значения в поле «Цена»

Создание объявления

☐ Продажа ☒ Покупка

Марка

Модель

Год выпуска

Пробег

Кузов

Привод

Объем двигателя

Цена

Рисунок Г.12 – Попытка ввода значения из 8 знаков в поле «Цена»

Создание объявления

☐ Продажа ☒ Покупка

Марка

Модель

Год выпуска

Пробег

Кузов

Привод

Объем двигателя

Цена

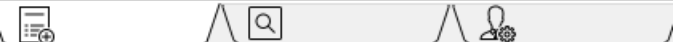


Рисунок Г.13 – Попытка ввода значения меньше минимального значения int в поле «Цена»

Создание объявления

☐ Продажа ☒ Покупка

Марка	<input type="text" value="Honda"/>
Модель	<input type="text" value="Accord"/>
Год выпуска	<input type="text" value="2018"/>
Пробег	<input type="text" value="8000 км"/>
Кузов	<input type="text" value="Седан"/>
Привод	<input type="text" value="Передний"/>
Объем двигателя	<input type="text" value="1.6"/>
Цена	<input type="text" value="2147483 руб"/>

Выбрать фото

Создать объявление

Рисунок Г.14 – Попытка ввода значения больше максимального значения int в поле «Цена»