

**МИНОБРНАУКИ РОССИИ**  
**федеральное государственное бюджетное**  
**образовательное учреждение высшего образования**  
**«Череповецкий государственный университет»**

Институт (факультет)	<u>Информационных технологий</u>
Направление	<u>09.03.04 Программная инженерия</u>
подготовки	
(специальность)	
Выпускающая кафедра	<u>Математическое и программное обеспечение электронно-вычислительных машин</u>

## ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Название работы	<u>Разработка программного обеспечения для мобильных устройств «Помощь на дороге»</u>

Студента	<u>Кравинского Олега Алексеевича</u>
	Ф.И.О.

### ДОПУСТИТЬ К ЗАЩИТЕ

Директор института	<u>Е.В. Ершов</u>
Заведующий выпускающей кафедрой	<u>Е.В. Ершов</u>
Руководитель выпускной квалификационной работы	<u>С.В. Гордеев</u>
Нормоконтролер	<u>Л.Н. Виноградова</u>
Выпускник	<u>О.А. Кравинский</u>

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	8
1 Сравнительный анализ отечественных и зарубежных аналогов проектируемой системы.....	10
2 Выбор технологии, среды и языка программирования .....	13
2.1 Выбор модели жизненного цикла программного обеспечения .....	13
2.2 Выбор подхода к разработке.....	14
2.3 Выбор инструментальных средств .....	15
2.3.1 Выбор инструментальных средств клиентской части приложения .....	15
2.3.2 Выбор инструментальных средств серверной части приложения .....	17
2.4 Выбор CASE-средств.....	17
3 Анализ процесса обработки информации, выбор структур данных для ее хранения, выбор методов и алгоритмов решения задачи .....	18
3.1 Анализ процесса обработки информации и уточнение требований к ПО..	18
3.2 Выбор методов и разработка основных алгоритмов решения задачи.....	19
3.2.1 Определение расстояния между географическими координатами .....	20
3.2.2 Алгоритм получения заявок о помощи .....	20
4 Разработка спецификаций проектируемой системы .....	22
4.1 Построение диаграммы вариантов использования.....	22
4.1.1 Вариант использования «Редактировать данные о пользователе».....	23
4.1.2 Вариант использования «Просмотреть заявки» .....	24
4.1.3 Вариант использования «Получить уведомления» .....	28
4.1.4 Вариант использования «Добавить заявку».....	29
4.1.5 Вариант использования «Просмотреть заявки пользователя».....	32
4.1.6 Вариант использования «Регистрация в приложении» .....	34

4.1.7 Вариант использования «Авторизация в приложении» .....	35
4.2 Построение контекстной диаграммы классов.....	36
4.3 Построение диаграмм последовательностей системы .....	37
4.4 Проектирование структур данных и построение диаграмм отношений компонентов данных .....	51
4.4.1 Логическое проектирование базы данных серверной части .....	51
4.4.2 Логическое проектирование базы данных клиентской части .....	53
5 Проектирование системы .....	55
5.1 Проектирование структуры системы и построение диаграмм пакетов .....	55
5.2 Проектирование классов пакета «Объекты модели» .....	58
5.2.1 Построение исходной диаграммы классов.....	58
5.2.2 Построение диаграмм последовательностей действий.....	60
5.2.3 Построение уточненной диаграммы классов.....	64
5.2.4 Детальное проектирование классов .....	64
5.3 Проектирование классов пакета «Управление местоположением» .....	66
5.3.1 Построение исходной диаграммы классов.....	66
5.3.2 Построение диаграмм последовательностей действий.....	67
5.3.3 Построение уточненной диаграммы классов.....	71
5.3.4 Детальное проектирование классов .....	71
5.4 Проектирование классов пакета «Интерфейс к базе данных SQLite».....	72
5.4.1 Построение исходной диаграммы классов.....	72
5.4.2 Построение диаграмм последовательностей действий.....	74
5.4.3 Построение уточненной диаграммы классов.....	78
5.4.4 Детальное проектирование классов .....	78
5.5 Проектирование классов пакета «JSON-парсеры» .....	81

5.5.1 Построение исходной диаграммы классов.....	81
5.5.2 Построение диаграмм последовательностей действий.....	82
5.5.3 Построение уточненной диаграммы классов.....	86
5.5.4 Детальное проектирование классов .....	86
5.6 Проектирование классов пакета «Управление объектами модели» .....	87
5.6.1 Построение исходной диаграммы классов.....	87
5.6.2 Построение диаграмм последовательностей действий.....	88
5.6.3 Построение уточненной диаграммы классов.....	92
5.6.4 Детальное проектирование классов .....	92
5.7 Проектирование классов пакета «Управление добавлением заявок» .....	94
5.7.1 Построение исходной диаграммы классов.....	94
5.7.2 Построение диаграмм последовательностей действий.....	95
5.7.3 Построение уточненной диаграммы классов.....	99
5.7.4 Детальное проектирование классов .....	99
5.8 Проектирование классов пакета «Управление пользователем» .....	102
5.8.1 Построение исходной диаграммы классов.....	102
5.8.2 Построение диаграмм последовательностей действий.....	104
5.8.3 Детальное проектирование классов .....	104
5.9 Проектирование классов пакета «Управление списком заявок» .....	109
5.9.1 Построение исходной диаграммы классов.....	109
5.9.2 Построение диаграмм последовательностей действий.....	111
5.9.3 Детальное проектирование классов .....	114
5.10 Проектирование классов пакета «Интерфейс к базе данных MySQL» ...	116
5.10.1 Построение исходной диаграммы классов.....	116
5.10.2 Построение диаграмм последовательностей действий.....	117

5.10.3 Детальное проектирование классов .....	118
5.11 Проектирование пакета «База данных MySQL».....	119
5.12 Проектирование пакета «База данных SQLite» .....	121
5.13 Построение диаграмм компонентов.....	125
5.14 Построение диаграмм размещения .....	129
6 Проектирование интерфейса пользователя .....	130
6.1 Построение графа диалога .....	130
6.2 Разработка форм ввода-вывода информации.....	132
7 Выбор стратегии тестирования, разработка тестов, программа и методика испытаний .....	137
7.1 Объект испытаний.....	137
7.2 Цель испытаний.....	137
7.3 Требования к информационному, аппаратно-программному обеспечению и документации.....	137
7.3.1 Требования к функциональным характеристикам .....	137
7.3.2 Требования к надежности .....	138
7.3.3 Требования к аппаратному и информационному обеспечению .....	139
7.3.4 Требования к программной документации .....	139
7.4 Состав, порядок и методы испытаний .....	139
7.4.1 Состав испытаний .....	139
7.4.2 Методы испытаний .....	140
7.5 Результаты проведения испытаний .....	141
ЗАКЛЮЧЕНИЕ .....	142
СПИСОК ЛИТЕРАТУРЫ.....	144
ПРИЛОЖЕНИЕ 1. Техническое задание.....	147

ПРИЛОЖЕНИЕ 2. Текст программы .....	158
ПРИЛОЖЕНИЕ 3. Спецификация .....	246
ПРИЛОЖЕНИЕ 4. Спецификации на модули .....	247
ПРИЛОЖЕНИЕ 5. Руководство пользователя.....	282
ПРИЛОЖЕНИЕ 6. Наборы тестовых данных и результатов тестирования .....	315

## ВВЕДЕНИЕ

В настоящее время у водителей часто возникает необходимость в быстром устранении внезапно возникшей в процессе дорожного движения неисправности или поломки транспортного средства.

Большинство поломок транспортного средства автовладелец не может выявить и устранить самостоятельно на месте. В подобной ситуации обращение в автосервис чаще всего затруднительно и/или необоснованно дорого (например, если поломка автомобиля произошла на малооживлённой трассе, вдали от ближайших населенные пунктов), поэтому в этой и других подобных ситуациях водителю приходится просить помочь у попутчиков на трассе [16].

Возникает проблема своевременного и информативного оповещения других участников дорожного движения, когда водителю, у которого произошла поломка требуется техпомощь. Проблема заключается в том, что в определенных ситуациях водителю не всегда удается достаточно быстро оповестить других автовладельцев о том, что ему требуется их помощь, размещая знак аварийной остановки рядом с транспортным средством.

Основными средствами передачи сообщений о техпомощи в подобной ситуации выступают (в виду их распространенности на современном этапе развития информационных технологий), имеющиеся у большинства участников дорожного движения мобильные устройства (смартфоны и планшеты).

Однако на данный момент не существует программного обеспечения для водителей г. Череповца и Череповецкого района для мобильных устройств, которое в полной мере позволило бы своевременно рассылать информативные сообщения водителей, которые нуждаются в техпомощи, другим участникам дорожного движения, которые готовы оказать помощь.

Таким образом, разработка мобильного приложения для водителей г. Череповца и Череповецкого района, которое бы позволило бы водителю, которому требуется помочь на дороге, отправить информативные сообщения об

этом другим участникам дорожного движения, которые находятся поблизости от автовладельца, является актуальной задачей.

Целью работы является предоставление сервиса взаимопомощи для водителей, которым при внезапной поломке или неисправности дорожно-транспортного средства требуется помочь со стороны других участников дорожного движения.

В работе предполагается решение следующих задач:

- проанализировать предметную область;
- провести сравнительный анализ имеющихся решений поставленной задачи и обосновать актуальность разработки;
- разработать требования к программному обеспечению для мобильных устройств;
- осуществить выбор технологии, среды и языка программирования;
- разработать спецификации для разрабатываемого программного обеспечения;
- осуществить проектирование разрабатываемого программного обеспечения;
- осуществить реализацию базы данных приложения и серверной части приложения;
- осуществить реализацию основных модулей клиентской части приложения – мобильного приложения для платформы Android;
- выполнить тестирование реализованных модулей;
- подготовить программную документацию.

## 1 Сравнительный анализ отечественных и зарубежных аналогов проектируемой системы

В качестве аналогов разрабатываемого программного обеспечения (ПО) для мобильных устройств можно выделить:

- группу «ДТП Череповца» в социальной сети «ВКонтакте»;
- мобильное приложение RoadHelper («Сервис взаимопомощи на дороге»).

Группа «ДТП Череповца» в социальной сети «ВКонтакте» (рис. 1.1): её задача заключается в поиске очевидцев дорожно-транспортных происшествий (ДТП) для связи с государственной инспекцией безопасности дорожного движения (ГИБДД), для улучшения дорожной ситуации и помощи автомобилистам и пешеходам [14].

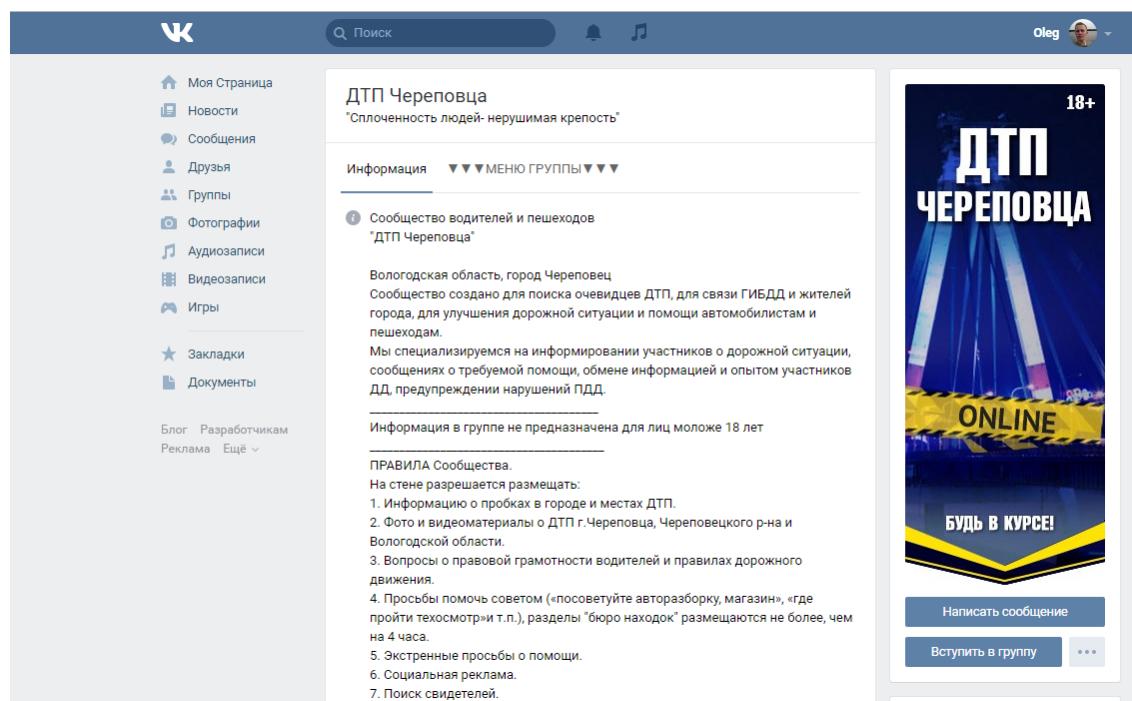


Рис. 1.1. Группа «ДТП Череповца» в социальной сети «ВКонтакте»

Сервис позволяет пользователям оставить сообщение в ленте (списке) сообщений сообщества. Пользователи имеют возможность просматривать список сообщений, оставлять комментарии к сообщению. Добавлять новые

сообщения могут пользователи, которые зарегистрированы (имеют учетную запись) в социальной сети.

Указанный сервис включает разнообразную специфику размещаемых сообщений: вопросы в сфере правовой помощи, фото и видеоматериалы о ДТП г. Череповца и Череповецкого района, социальная реклама, сообщения об экстренной помощи и другая информация.

К недостаткам группы «ДТП Череповца» в качестве сервиса оповещения о необходимости техпомощи водителю со стороны других участников дорожного движения можно отнести:

- для просмотра сообщений в сообществе необходимо войти под своим логином и паролем в социальную сеть «ВКонтакте» и перейти в указанную группу, что во многих случаях затруднительно в процессе дорожного движения;
- несвоевременное оповещение других участников группы о новой записи в сообществе;
- во многих случаях участники группы, которые прочитали сообщение о необходимости оказания помощи, находятся на значительном расстоянии от водителя, которому эта помощь необходима.

Другим аналогом разрабатываемого программного обеспечения служит мобильное приложение Road Helper («Сервис взаимопомощи на дороге») (рис. 1.2) для мобильных устройств на платформе Android. Приложение позволяет водителям попросить техпомощь на дороге, создав метку на карте и описав свою поломку [7].

Процедура оформления заявки следующая: водитель оставляет свои контактные данные, описывает проблему, прикрепляет фото и указывает адрес. Уведомление о его заявке рассыпается всем пользователям, находящимся в заданном радиусе. Другие пользователи приложения просматривают список добавленных заявок водителей, которые находятся на определенном расстоянии (радиусе) от водителя, могут просмотреть и написать сообщение к выбранной заявке из списка.

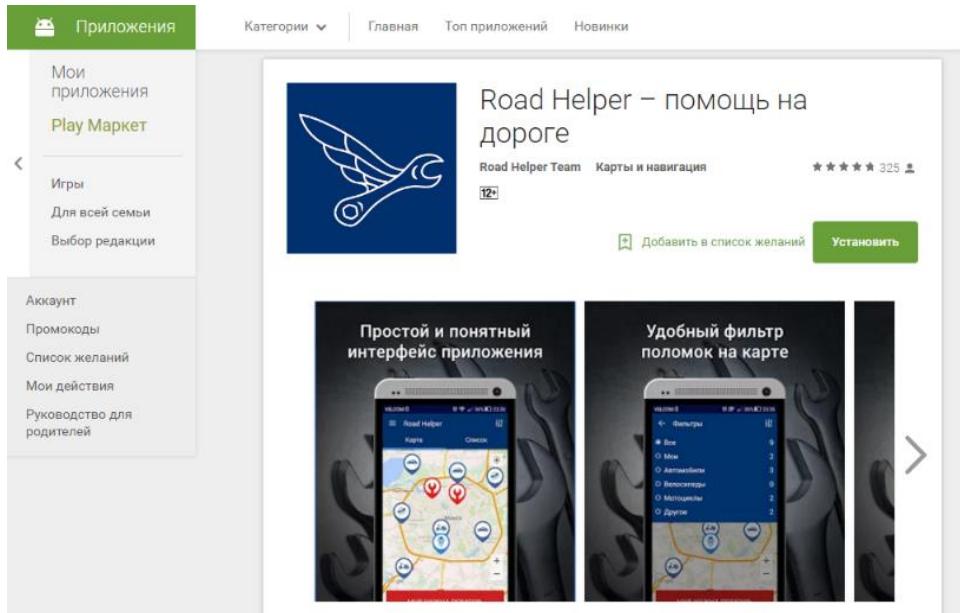


Рис. 1.2. Мобильное приложение Road Helper («Сервис взаимопомощи на дороге»)

К недостаткам приложения можно отнести то, что:

- приложение на настоящий момент не получило распространения на территории г. Череповца и Череповецкого района;
- отсутствует возможность удаления и редактирования заявок пользователем, который добавил их в систему (например, данная функция необходима при ошибочном добавлении заявки или в случае, когда водитель уже получил требуемую помощь);
- отсутствует поддержка фильтрации заявок по статусу выполнения заявки («требуется помощь», «помощь оказывается», «помощь оказана»).

На основании приведенного исследования существующих программных продуктов для водителей г. Череповца и Череповецкого района предлагается разработать мобильное приложение «Помощь на дороге», которое позволит водителям создавать заявки на помощь со стороны других участников дорожного движения с возможностью рассылки уведомлений другим участникам, находящимся на определенном расстоянии от водителя, оставившего заявку в приложении.

## 2 Выбор технологии, среды и языка программирования

### 2.1 Выбор модели жизненного цикла программного обеспечения

Жизненный цикл программного обеспечения (ЖЦ ПО) – это период от момента возникновения идеи создания программного обеспечения до момента завершения поддержки программного обеспечения фирмой-разработчиком или фирмой, выполняющей сопровождение [21].

Под моделью ЖЦ понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ [5].

Для разработки мобильного приложения «Помощь на дороге» проведен анализ моделей ЖЦ и выбрана каскадная модель ЖЦ с промежуточным контролем (рис. 2.1), которая предусматривает последовательное выполнение этапов ЖЦ с возможностью возврата на предыдущие этапы [1, 11].



Рис. 2.1. Каскадная модель с промежуточным контролем

Каскадная модель ЖЦ с промежуточным контролем имеет следующие преимущества для разрабатываемого программного обеспечения:

- данная модель ЖЦ позволяет сформулировать достаточно полно сроки и этапы разработки на ранних этапах, в отличие от спиральной и инкрементной моделей, что является немаловажным для проектируемой системы, поскольку разработка будет вестись в ограниченные сроки [18];
- явным преимуществом каскадной модели ЖЦ с промежуточным контролем является то, что модель подходит для систем, у которых требования удается выявить на ранних этапах проектирования. Это

справедливо для ПО для мобильных устройств «Помощь на дороге», поскольку его разработка преследует решение конкретной проблемы взаимопомощи водителей на дороге, что позволяет достаточно полно сформулировать требования к системе ещё на этапе анализа;

- недостатки указанной модели связаны прежде всего с разработкой систем, для которых требования не удается выявить на ранних этапах проектирования, что представляется маловероятным для мобильного приложения «Помощь на дороге».

## 2.2 Выбор подхода к разработке

Для разработки программного обеспечения в настоящее время применяют два подхода [4, 5]:

- структурный подход;
- объектный подход.

В основе структурного подхода лежит декомпозиция (разбиение на части) сложных систем с целью последующей реализации в виде отдельных небольших подпрограмм.

В основе объектного подхода лежит объектная декомпозиция, т. е. представление разрабатываемого программного обеспечения в виде совокупности объектов, в процессе взаимодействия которых через передачу сообщений происходит выполнение требуемых функций.

Достоинства объектного подхода по сравнению со структурным [5]:

- более естественная декомпозиция предметной области, которая существенно облегчает разработку;
- независимая разработка отдельных частей (объектов) программы;
- существенное увеличение степени повторного использования кода за счет способов организации программ, основанных на механизмах наследования, полиморфизма, композиции, наполнения;

– наличие стандартного средства описания разработки программных продуктов с использованием объектного подхода - языка UML (unified modeling language – унифицированный язык моделирования). Спецификация разрабатываемого программного обеспечения при использовании UML объединяет несколько моделей: использования, логическую, реализации, процессов, развертывания - каждая из которых характеризует определенный аспект моделируемой системы, а вместе они представляют собой полную модель разрабатываемого ПО [17, 19].

Поскольку для предметной области мобильного приложения «Помощь на дороге» естественной декомпозицией является объектная декомпозиция и на основании приведенных преимуществ объектного подхода для разрабатываемого ПО выбран объектный подход к разработке.

## 2.3 Выбор инструментальных средств

Согласно сформированным требованиям к мобильному приложению в техническом задании (см. прил. 1), разрабатываемое приложение имеет клиент-серверную архитектуру:

- клиентская часть представляет приложение для мобильных устройств под управлением операционной системы Android;
- серверная часть предоставляет набор методов API (application programming interface - интерфейс прикладного программирования) для работы с базой данных приложения, размещенной на удаленном сервере, запросы к которой будут выполнять клиентские приложения.

### 2.3.1 Выбор инструментальных средств клиентской части приложения

Android - операционная система (ОС) для мобильных устройств: смартфонов, планшетных компьютеров. Приложения под ОС Android являются программами в нестандартном байт-коде для виртуальной машины Dalvik. Для работы над приложениями доступно множество библиотек [6, 20].

Для разрабатываемого мобильного приложения выбрана интегрированная среда разработки от компании JetBrains - Android Studio, официальное средство разработки Android приложений, рекомендуемое компанией Google [12].

В Android Studio реализованы [6, 20]:

- поддержка сборки приложения, основанной на системе автоматической сборки приложений Gradle;
- оптимизация написанного кода программы и быстрое исправление дефектов;
- инструменты для поиска проблем с производительностью, с совместимостью версий;
- окно предварительного просмотра, показывающее запущенное приложение сразу на нескольких устройствах и в реальном времени.

Для написания приложений в среде Android Studio используется язык программирования Java.

Java — объектно-ориентированный язык программирования, разработанный компанией Sun Microsystem.

Его особенности [12]:

- программы на Java транслируются в байт-код, выполняемый на виртуальной Java-машине (JVM);
- Java сочетает в себе достоинства объектно-ориентированной разработки приложений;
- Java предоставляет программисту богатый набор классов объектов.

Для хранения данных на мобильном устройстве пользователя выбрана база данных SQLite, являющаяся компактной встраиваемой реляционная базой данных. SQLite не использует парадигму клиент-сервер, как другие системы управления базами данных (СУБД), такие как Microsoft SQLServer, MySQL, а предоставляет библиотеку, с которой клиентская часть приложения компонуется. В качестве протокола обмена данными приложения и базы данных используются вызовы функций библиотеки SQLite [12].

### 2.3.2 Выбор инструментальных средств серверной части приложения

База данных приложения будет располагаться на удаленном сервере под управлением web-сервера Apache, который получает и обрабатывает HTTP-запросы от клиентской части приложения [8].

В качестве СУБД для управления базой данных приложения на сервере выбрана система управления реляционными базами данных MySQL.

MySQL – это одна из самых популярных и самых распространенных СУБД в интернете. Она не предназначена для работы с большими объемами информации, но ее применение идеально для сайтов, клиент-серверных мобильных приложений. MySQL отличается хорошей скоростью работы, надежностью, гибкостью. Немаловажным фактором является ее бесплатность: MySQL распространяется на условиях общей лицензии [12].

Для написания серверной части выбран язык PHP, скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений [9].

Для написания скриптов на языке PHP используется кроссплатформенный редактор исходного кода Sublime Text [12].

### 2.4 Выбор CASE-средств

На современном этапе развития разработки программного обеспечения применяются автоматизированные технологии его разработки и сопровождения – CASE-технологии (computer-aided software engineering – разработка программного обеспечения с использованием компьютерной поддержки) [5].

Для разработки ПО выбрано CASE-средство Software Ideas Modeler, программа для моделирования диаграмм UML. Средство поддерживает генерацию кода при выполнении модели UML на многие языки программирования, в том числе Java и PHP. Кроме UML-модели средство предоставляет 14 различных моделей среди которых: ER-модель, DFD-модель. Программа имеет удобный графический интерфейс и позволяет экспортить построенные диаграммы в различные форматы файлов [10].

### 3 Анализ процесса обработки информации, выбор структур данных для ее хранения, выбор методов и алгоритмов решения задачи

#### 3.1 Анализ процесса обработки информации и уточнение требований к ПО

Процесс обработки информации, который осуществляется мобильном приложении «Помощь на дороге», включает следующие пункты:

1) добавление заявки на помощь пользователем мобильного приложения.

Входная информация (данные заявки) представляет собой информацию о:

- пользователе: имя, фотография пользователя;
- заявке на помощь: описание и фото неисправности, местоположение пользователя (широта и долгота), выбранное на карте Google или автоматически полученное модулем для определения местоположения устройства, дата и время добавления заявки.

Процесс обработки полученной информации представляется следующим образом:

- информация проверяется на правильность (корректность) заполнения. В случае, если введенные данные имеют ошибочный или неполный формат, пользователю выдается сообщение о необходимости проверки и корректировки данных;
- если введенные данные прошли проверку на правильность заполнения, то происходит их сохранение в базе данных, размещённой на удаленном сервере.

2) просмотр списка заявок пользователем мобильного приложения.

Мобильным устройством делается запрос к базе данных на сервере, и выполняется загрузка заявок о помощи на устройство пользователя в специальном текстовом формате представления данных JSON [12].

Каждая полученная заявка включает следующую информацию:

- текст сообщения заявки;
- имя, фото пользователя, который её добавил;
- время добавления заявки;

- данные о местоположении пользователя;
- список сообщений о готовности оказать помощь.

Полученные данные сохраняются в базе данных SQLite мобильного приложения и отображаются пользователю на экране устройства.

Описание алгоритма получения актуальных заявок представлено в п. 3.2.

### 3) добавление предложения помощи к заявке.

Если в процессе просмотра заявок на помощь пользователь желает предложить помощь, то он выбирает заявку из списка заявок и добавляет сообщение (предложение помощи), которое включает:

- имя и фото пользователя;
- текст сообщения;
- местоположение (долгота и широта), выбранное на карте Google или автоматически полученное модулем для определения местоположения устройства;
- дату и время добавления сообщения.

Затем происходит проверка введенных данных предложения помощи и их сохранение в базе данных на сервере.

## 3.2 Выбор методов и разработка основных алгоритмов решения задачи

Ключевым элементом разрабатываемого программного обеспечения для мобильных устройств «Помощь на дороге» является база данных, размещаемая на удаленном сервере, которая содержит сведения о заявках на помощь пользователей приложения.

Требуется разработать алгоритм получения списка актуальных заявок, которые добавлены в течение суток на момент их получения из базы данных пользователем устройства.

Алгоритм включает определение расстояния между двумя географическими координатами на земной поверхности.

### 3.2.1 Определение расстояния между географическими координатами

Географические координаты однозначно определяют положение точки на земной поверхности. Каждая координата — это набор двух чисел: широта и долгота, измеренных в радианах, которые однозначно определяют положение предмета на земной поверхности [13].

Для расчета расстояния  $s$  в метрах между двумя точками используется «сферическая теорема косинусов» [13]:

$$s = \arccos(\sin\varphi_1 \sin\varphi_2 + \sin\varphi_1 \cos\varphi_2 \cos\Delta\lambda) * R, \quad (3.1)$$

где:

$\varphi_1, \lambda_1$  - широта и долгота первой точки в радианах;

$\varphi_2, \lambda_2$  - широта и долгота второй точки в радианах;

$\Delta\lambda = \lambda_1 - \lambda_2$  – разница координат по долготе;

$R=6372795$  метров – радиус Земли.

При небольших расстояниях формула (3.1) может приводить к ошибке вычисления расстояний порядка 0.5%. Для более точных расчетов расстояния  $s$  в метрах используется «формула гаверсинусов» (3.2) [13].

$$s = 2\arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_1 - \varphi_2}{2}\right) + \cos\varphi_1 \cos\varphi_2 \sin^2\left(\frac{\Delta\lambda}{2}\right)}\right) * R \quad (3.2)$$

### 3.2.2 Алгоритм получения заявок о помощи

Алгоритм получения актуальных заявок из базы данных основан на определении местоположения между двумя географическими координатами:

- $m_{польз}$  – текущее местоположение пользователя, которое характеризуется набором двух чисел: широтой -  $lat_{польз}$  и долготой -  $long_{польз}$ ;

- $m_{заявки}$  – местоположение заявки, которое хранится в базе данных. Местоположение  $m_{заявки}$  характеризуется набором двух чисел: широтой -  $lat_{польз}$  и долготой -  $long_{польз}$ .

Для определения местоположения между двумя координатами  $m_{польз}$  и  $m_{заявки}$  используется формула (3.2).

Алгоритм получения заявок обобщенно можно представить следующим образом:

1) пользователем мобильного приложения (клиентской частью) выполняется HTTP-запрос к серверной части приложения на получение списка заявок на помощь.

2) серверная часть принимает данные от клиентской части и делает запрос на выборку заявок на помощь из базы данных MySQL, удовлетворяющих условию: разница времени добавления заявки и текущего времени не больше 24 часов ( $t_{заявки} - t_{текущее} \leq 24$  часа).

3) данные заявок преобразуются в формат JSON и передаются клиентской части.

4) клиентская часть, получив данные заявок, сохраняет их в базе данных приложения SQLite (локальной базе данных).

5) клиентская часть извлекает из локальной базы данных список заявок. Вычисляется расстояние  $r_{заявки}$  от заявки до пользователя устройства по формуле (3.2). Выполняется сравнение  $r_{заявки}$  и радиуса получения заявок пользователя –  $r_{пользователя}$ , даты добавления –  $t_{заявки}$  и текущего значения даты –  $t_{текущее}$ .

В список добавляются те заявки, расстояние до которых меньше или равно радиусу получения заявок (т.е.  $r_{заявки} \leq r_{пользователя}$ ) и разница времени добавления и текущего времени не превосходит 24 часов ( $t_{заявки} - t_{текущее} \leq 24$  час.).

Примечание: если значение радиуса получения заявок пользователя –  $r_{пользователя}$  принимает значение «получать все заявки», то в 5 пункте алгоритма выбираются все заявки, для которых разница времени добавления и текущего времени не превосходит 24 часов ( $t_{заявки} - t_{текущее} \leq 24$  час.).

## 4 Разработка спецификаций проектируемой системы

### 4.1 Построение диаграммы вариантов использования

Диаграммы вариантов использования позволяют наглядно представить ожидаемое поведение системы. Основными понятиями диаграмм вариантов использования являются: действующее лицо, вариант использования, связь [5].

Действующее лицо - внешняя по отношению к разрабатываемому программному обеспечению сущность, которая взаимодействует с ним с целью получения или предоставления какой-либо информации (пользователь, другое программное обеспечение).

Вариант использования - некоторая очевидная для действующего лица процедура, решаяшая его конкретную задачу.

Связь - взаимодействие действующих лиц и соответствующих вариантов использования.

На основании требований к проектируемому мобильному приложению «Помощь на дороге» (см. прил. 1 «Техническое задание») разработана диаграмма вариантов использования (рис. 4.1).

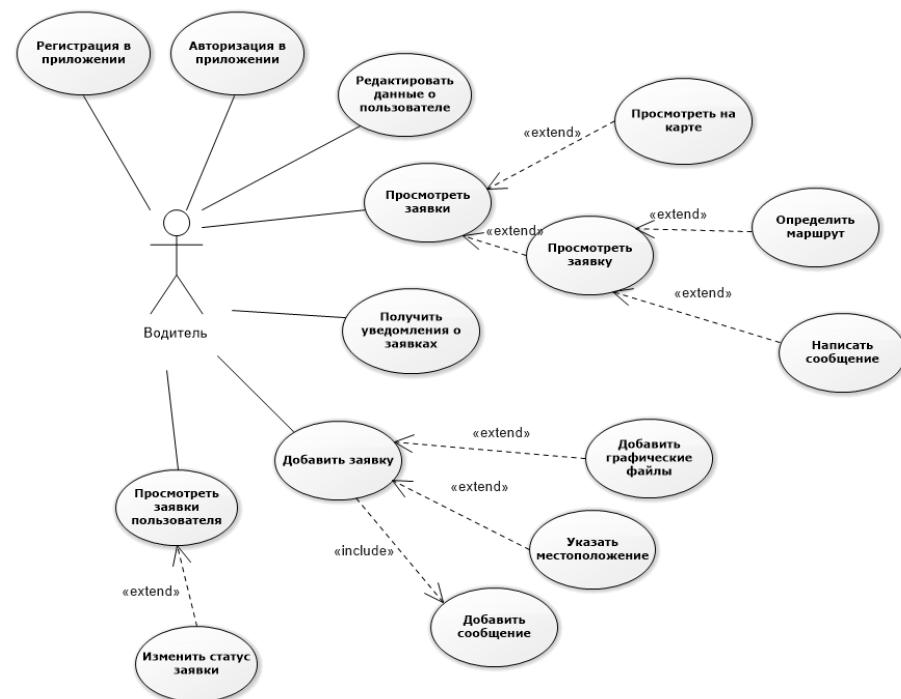


Рис. 4.1. Диаграмма вариантов использования ПО «Помощь на дороге»

Представлен список детальных описаний вариантов использования первого уровня, включающих: описание расширяющих и дополняющих вариантов, описание типичного хода событий (с указанием возможных альтернатив и дополнительной информации).

#### 4.1.1 Вариант использования «Редактировать данные о пользователе»

Краткое описание варианта использования представлено в табл. 4.1.

Таблица 4.1  
Вариант использования «Редактировать данные о пользователе»

Название варианта	Редактировать данные о пользователе
Цель	Просмотр, изменение данных пользователя: имя, фото
Действующие лица	Водитель
Краткое описание	Изменение данных предполагает загрузку и отображение имеющихся данных о пользователе устройства из базы, ввод новых данных пользователем и проверку правильности ввода, сохранение измененных данных в базе
Тип варианта	Основной
Расширяет варианты	-
Включает варианты	-

Типичный ход событий варианта использования «Редактировать данные о пользователе» представлен в табл. 4.2.

Таблица 4.2  
Типичный ход событий варианта использования «Редактировать данные о пользователе»

Действия исполнителя	Отклик системы
1. Пользователь инициирует редактирование данных о пользователе	2. Система делает запрос к базе данных, загружает текущие данные о пользователе (имя, фото) из базы, отображает их на экране и ожидает ввода данных пользователем
3. Пользователь вводит данные о пользователе	4. Система проверяет введенные данные, сохраняет их в базе данных и выводит сообщение пользователю об изменении данных 5. Система переходит в режим ожидания

**Альтернативы типичного хода событий варианта использования  
«Редактировать данные о пользователе»**

5. Если пользователь не ввел значение имени пользователя, то система отображает сообщение об ошибке и возвращается на шаг 3.

**Дополнительная информация**

1. Необходимо обеспечить возможность выхода из варианта на любом этапе.

**4.1.2 Вариант использования «Просмотреть заявки»**

Краткое описание варианта использования в табл. 4.3.

Таблица 4.3

**Вариант использования «Просмотреть заявки»**

Название варианта	Просмотреть заявки
Цель	Просмотр заявок в системе, в соответствии с заданным радиусом получения заявок и местоположением устройства
Действующие лица	Водитель
Краткое описание	Просмотр заявок предусматривает: определение местоположения (широта, долгота) пользователя, получение радиуса пользователя, получение данных из базы о заявках (идентификатор заявки, сообщение, имя и фото пользователя, местоположение (широта, долгота)), местоположение которых не превосходит заданный радиус, в некоторых случаях просмотр местоположения заявок на карте
Тип варианта	Основной
Расширяет варианты	Просмотреть на карте (см. табл. 4.4) Просмотреть заявку (см. табл. 4.5)
Включает варианты	-

Варианты использования, расширяющие вариант «Просмотреть заявки», представлены в табл. 4.4, 4.5.

Таблица 4.4

**Вариант использования «Просмотреть на карте»**

Название варианта	Просмотреть на карте
Цель	Просмотр карты с отметками местоположений заявок
Действующие лица	Водитель
Краткое описание	Отображение карты с отметками местоположений заявок
Тип варианта	Основной
Расширяет варианты	-
Включает варианты	-

Таблица 4.5

## Вариант использования «Просмотреть заявку»

Название варианта	Просмотреть заявку
Цель	Просмотр данных о заявке
Действующие лица	Водитель
Краткое описание	Просмотр заявки предусматривает: получение данных о заявке (текст сообщения, графические файлы, имя, фото пользователя, местоположение (долгота, широта), текст сообщений предложений помощи) из базы, отображение данных на экране устройства
Тип варианта	Основной
Расширяет варианты	Определить маршрут (табл. 4.6) Написать сообщение (табл. 4.7)
Включает варианты	Отправить сообщение

Варианты использования, расширяющие вариант «Просмотреть заявку», представлены в табл. 4.6, 4.7.

Таблица 4.6

## Вариант использования «Определить маршрут»

Название варианта	Определить маршрут
Цель	Определение маршрута между местоположением устройства пользователя и местоположением заявки
Действующие лица	Водитель
Краткое описание	Определение маршрута предусматривает: определение местоположения (широта, долгота) устройства, расчет и отображение маршрута на карте
Тип варианта	Основной
Расширяет варианты	-
Включает варианты	-

Таблица 4.7

## Вариант использования «Написать сообщение»

Название варианта	Написать сообщение
Цель	Сохранить сообщение предложения помощи к выбранной пользователем заявке
Действующие лица	Водитель
Краткое описание	Ввод и сохранение текста сообщения в базе данных
Тип варианта	Основной
Расширяет варианты	-
Включает варианты	-

Типичный ход событий варианта использования «Просмотреть заявки» с указанием возможных альтернатив представлен в табл. 4.8 - 4.12.

Таблица 4.8

## Типичный ход событий варианта использования «Просмотреть заявки».

Действия исполнителя	Отклик системы
1. Пользователь выбрал просмотр заявок на карте	2. Система определяет местоположение пользователя и радиус получения заявок, получает из базы данные о заявках (идентификатор, текст сообщения, фотографию заявки, местоположение (долгота, широта), имя, фото пользователя), расстояние до которых не превосходит заданного радиуса
3. Пользователь выбирает дополнительные варианты: а) Если выбран просмотр на карте, то см. раздел «Просмотреть на карте» (табл. 4.9); б) Если выбирает заявку для просмотра, то см. раздел «Просмотреть заявку» (табл. 4.10)	4. Система переходит в режим ожидания

## Альтернативы типичного хода событий варианта использования «Просмотреть заявки»

1. Если не удается определить местоположение пользователя, то система выдает сообщение о невозможности определения местоположения и переходит к шагу 4.
2. Если не удается получить заявки из базы, то система выдает сообщение о невозможности получения заявок из базы и переходит к шагу 4.
2. Если в базе данных нет заявок, удовлетворяющих условию отбора, то система выдает сообщение о том, что нет заявок, и переходит к шагу 4.
3. Если пользователь не выбрал дополнительные варианты, то система переходит к шагу 4.

## Дополнительная информация

1. Необходимо обеспечить возможность выхода из варианта на любом этапе.

Таблица 4.9

## Раздел «Просмотреть на карте»

Действия исполнителя	Отклик системы
1. Пользователь инициирует получение списка заявок	2. Система отображает карту с отметками местоположения заявок (широта, долгота)

### Альтернативы раздела «Просмотреть на карте»

1. Если не удается определить местоположение пользователя, то система выдает сообщение о невозможности определения местоположения.

Таблица 4.10

### Раздел «Просмотреть заявку»

Действия исполнителя	Отклик системы
1. Пользователь инициирует просмотр заявки	2. Система отображает данные заявки (текст сообщения, графические файлы, местоположение (длгота, широта), имя, фото пользователя, статус заявки)
3. Пользователь выбирает дополнительные варианты: а) Если выбрано определение маршрута, то см. раздел «Определить маршрут» (табл. 4.11); б) Если выбрано написание сообщения, то см. раздел «Написать сообщение» (табл. 4.12); г) Пользователь возвращается к просмотру заявок	4. Система переходит к шагу 3

### Альтернативы раздела «Просмотреть заявку»

3. Если пользователь не выбрал дополнительные варианты, то система переходит в режим ожидания.

3. Если пользователь выбрал возврат к просмотру заявок, то система возвращается к шагу 4 варианта «Просмотреть заявки» (табл. 4.8).

Таблица 4.11

### Раздел «Определить маршрут»

Действия исполнителя	Отклик системы
1. Пользователь инициирует просмотр заявки	2. Рассчитать и отобразить маршрут на карте

Таблица 4.12

### Раздел «Написать сообщение»

Действия исполнителя	Отклик системы
1. Пользователь инициирует написание сообщения и вводит текст сообщения	2. Система проверяет данные сообщения и сохраняет их в базе данных

### Альтернативы раздела «Написать сообщение»

2. Если пользователь не ввел текст сообщения, то система выдает сообщение о необходимости ввести текст сообщения и возвращается на предыдущий шаг.

#### Дополнительная информация

1. Необходимо обеспечить возможность выхода из варианта на любом этапе.

#### 4.1.3 Вариант использования «Получить уведомления»

Краткое описание варианта использования представлено в табл. 4.13.

Таблица 4.13

#### Вариант использования «Получить уведомления»

Название варианта	Получить уведомления
Цель	Получить уведомление на устройство при появлении новой заявки в приложении
Действующие лица	Водитель
Краткое описание	Приложение определяет местоположение устройства, определяет список заявок, формирует уведомления о новых заявках
Тип варианта	Основной
Расширения варианта использования	-
Включает варианты	-

Типичный ход событий варианта использования «Получить уведомления» с указанием возможных альтернатив представлен в табл. 4.14.

Таблица 4.14

#### Типичный ход событий варианта использования «Получить уведомления»

Действия исполнителя	Отклик системы
1. Пользователь выбрал получение уведомлений	2. Запрос местоположения устройства (широта, долгота). 3. Формирование данных уведомлений заявок (текст сообщения, имя водителя), которые еще не были отправлены на устройство. 4. Отправка уведомлений о новых заявках
5. Пользователь получает уведомления	6. Система переходит в режим ожидания

## Альтернативы типичного хода событий варианта использования «Получить уведомления»

2. Если устройство не подключено для получения уведомлений, то система переходит к шагу 6.

2. Если не удается определить местоположение устройства, то система переходит к шагу 6.

### 4.1.4 Вариант использования «Добавить заявку»

Краткое описание варианта использования представлено в табл. 4.15.

Таблица 4.15

#### Вариант использования «Добавить заявку»

Название варианта	Добавить заявку
Цель	Добавить заявку на помощь в приложение
Действующие лица	Водитель
Краткое описание	Добавление заявки предусматривает: добавление сообщения с возможностью указания местоположения, добавления фотографии заявки, проверки введенных данных, получения данных о пользователе (имя, фото), сохранение данных в базу данных
Тип варианта	Основной
Расширяет варианты	Добавить графические файлы (табл. 4.18). Указать местоположение (табл. 4.17)
Включает варианты	Добавить сообщение (табл. 4.16)

Вариант использования, дополняющий вариант «Добавить заявку», представлен в табл. 4.16.

Таблица 4.16

#### Вариант использования «Добавить сообщение»

Название варианта	Добавить сообщение
Цель	Ввод и проверка текста сообщения
Действующие лица	Водитель
Краткое описание	Добавление сообщения предусматривает ввод текста сообщения
Тип варианта	Основной
Расширяет варианты	-
Включает варианты	-

Варианты использования, расширяющие вариант «Добавить заявку», представлены в табл. 4.17 - 4.18.

Таблица 4.17

## Вариант использования «Указать местоположение»

Название варианта	Указать местоположение
Цель	Уточнение местоположения пользователя, который добавляет заявку
Действующие лица	Водитель
Краткое описание	Указание местоположения предусматривает: отображение карты с отметкой автоматически определенного местоположения устройства, изменение местоположения на карте, внесение изменения в отправляемые данные о местоположении
Тип варианта	Основной
Расширяет варианты	-
Включает варианты	-

Таблица 4.18

## Вариант использования «Добавить графические файлы»

Название варианта	Добавить графические файлы
Цель	Добавить к заявке фотографию
Действующие лица	Водитель
Краткое описание	Вариант использования включает: выбор фотографии в форматах: .jpg, .png - из галереи устройства, отправку и сохранение файла фотографии на сервере
Тип варианта	Основной
Расширяет варианты	-
Включает варианты	-

Типичный ход событий варианта использования «Добавить заявку» с указанием возможных альтернатив представлен в табл. 4.19 - 4.21.

Таблица 4.19

## Типичный ход событий варианта использования «Добавить заявку»

Действия исполнителя	Отклик системы
1	2
1. Пользователь инициирует добавление заявки	2. Система запрашивает ввод данных заявки и определяет местоположение, данные пользователя устройства
3. Пользователь вводит текст сообщения	4. Система проверяет правильность введенных данных

## Продолжение табл. 4.19

1	2
<p>5. Пользователь выбирает дополнительные варианты:</p> <p>а) Если выбрано определение местоположения, то см. раздел «Указать местоположение» (табл. 4.21);</p> <p>б) Если выбрано добавление графических файлов, то см. раздел «Добавить графические файлы» (табл. 4.20)</p>	<p>6. Система проверяет правильность введенных данных, сохраняет их в базе и выдает сообщение о добавлении данных.</p> <p>7. Система переходит в режим ожидания</p>

**Альтернативы типичного хода событий варианта использования «Добавить заявку»**

4. Если пользователем введены недопустимые данные, то система выдает сообщение об ошибке и возвращается на предыдущий шаг.

**Дополнительная информация**

1. Необходимо обеспечить возможность выхода из варианта на любом этапе.

Таблица 4.20

**Раздел «Добавить графические файлы»**

Действия исполнителя	Отклик системы
1. Пользователь инициирует добавление графических файлов	2. Система открывает проводник устройства для выбора графических файлов
3. Пользователь выбирает графические файлы	4. Система добавляет графические файлы к заявке

**Альтернативы раздела «Добавить графические файлы»**

3. Если пользователь не выбрал файлы, система переходит к шагу 5 варианта использования «Добавить заявку» (табл.4.19).

Таблица 4.21

**Раздел «Указать местоположение»**

Действия исполнителя	Отклик системы
1. Пользователь инициирует указание местоположения	2. Система отображает карту, где показано текущее положение пользователя
3. Пользователь выбирает новое местоположение	4. Система изменяет местоположение пользователя

#### 4.1.5 Вариант использования «Просмотреть заявки пользователя»

Краткое описание варианта представлено в табл. 4.22.

Таблица 4.22

##### Вариант использования «Просмотреть заявки пользователя»

Название варианта	Просмотреть заявки пользователя
Цель	Просмотр списка заявок, добавленных пользователем
Действующие лица	Водитель
Краткое описание	Предусматривает получение идентификатора пользователя устройства, получение данных заявок пользователя из базы (идентификатор заявки, сообщение, имя, фото пользователя, местоположение)
Тип варианта	Основной
Расширяет варианты	Изменить статус заявки (табл. 4.23).
Включает варианты	-

Вариант использования «Изменить статус заявки», расширяющий вариант «Просмотреть заявки пользователя», представлен в табл. 4.23.

Таблица 4.23

##### Вариант использования «Изменить статус заявки»

Название варианта	Изменить статус заявки
Цель	Предусматривает изменение статуса для выбранной пользователем из списка заявки на помочь
Действующие лица	Водитель
Краткое описание	Изменение статуса выбранной заявки. Статус заявки может принимать значения: «требуется помочь» - если водителю необходима помощь; «помощь оказывается» - если водителю помощь уже оказывается; «помощь оказана» - если водитель получил помощь и помощь ему больше не требуется
Тип варианта	Основной
Расширяет варианты	-
Включает варианты	-

Типичный ход событий варианта использования «Просмотреть заявки пользователя» представлен в табл. 4.24 – 4.25, с указанием возможных альтернатив и дополнительной информации о ходе выполнения сценария варианта использования.

Таблица 4.24

**Типичный ход событий варианта использования «Просмотреть заявки  
пользователя»**

Действия исполнителя	Отклик системы
1. Пользователь выбрал просмотр заявок на карте	2. Получение данных о заявках из базы (идентификатор пользователя, текст сообщения, графические файлы, местоположение (долгота, широта), имя, фото пользователя), которые добавлены пользователем устройства
3. Пользователь выбирает дополнительные варианты: а) Если выбрана заявка на изменение статуса, то смотреть раздел «Изменить статус заявки» (табл. 4.25); б) Если не выбран дополнительный вариант, то система переходит к шагу 4	4. Система переходит в режим ожидания

**Альтернативы типичного хода событий варианта использования  
«Просмотреть заявки пользователя»**

2. Если нет соединения с базой данных, то система выдает сообщение и переходит к шагу 5.
2. Если в базе данных нет заявок, удовлетворяющих условию отбора, то система выдает сообщение о том, что нет заявок, и переходит к шагу 5.
4. Если пользователь не выбрал дополнительные варианты, то система переходит в режим ожидания.

**Дополнительная информация**

1. Необходимо обеспечить возможность выхода из варианта на любом этапе.

Таблица 4.25

**Раздел «Изменить статус заявки»**

Действия исполнителя	Отклик системы
1. Пользователь инициирует изменение статуса выбранной заявки	2. Система определяет идентификатор заявки и выводит пользователю список значений для статуса заявки
3. Пользователь выбирает значение статуса из выпадающего списка	4. Система делает запрос в базу на изменение статуса заявки и выводит пользователю сообщение об изменении статуса заявки

#### 4.1.6 Вариант использования «Регистрация в приложении»

Краткое описание варианта представлено в табл. 4.26.

Таблица 4.26

##### Вариант использования «Регистрация в приложении»

Название варианта	Регистрация в приложении
Цель	Регистрация пользователя в приложении
Действующие лица	Водитель
Краткое описание	Пользователь вводит логин и пароль, система осуществляет проверку введенных данных и регистрирует пользователя в системе
Тип варианта	Основной
Расширяет варианты	-
Включает варианты	-

Типичный ход событий варианта использования «Регистрация в приложении» представлен в табл. 4.27.

Таблица 4.27

##### Типичный ход событий варианта использования «Регистрация в приложении»

Действия исполнителя	Отклик системы
1. Пользователь выбрал регистрацию в приложении	2. Отображение окна для ввода регистрационных данных
3. Пользователь вводит логин и пароль	4. Система проверяет и отправляет данные в базу, выводит сообщение пользователю об успешности регистрации в приложении. 5. Система переходит в режим ожидания

##### Альтернативы типичного хода событий варианта использования «Регистрация в приложении»

4. Если нет соединения с базой данных, то система выдает сообщение об ошибке и переходит к шагу 5.

##### Дополнительная информация

1. Необходимо обеспечить возможность выхода из варианта на любом этапе.

#### 4.1.7 Вариант использования «Авторизация в приложении»

Краткое описание варианта представлено в табл. 4.28.

Таблица 4.28

#### Вариант использования «Авторизация в приложении»

Название варианта	Авторизация в приложении
Цель	Авторизация пользователя в приложении
Действующие лица	Водитель
Краткое описание	Пользователь вводит логин и пароль, система осуществляет проверку введенных данных и выводит пользователю сообщение об авторизации в приложении
Тип варианта	Основной
Расширяет варианты	-
Включает варианты	-

Типичный ход событий варианта использования «Авторизация в приложении» представлен в табл. 4.29.

Таблица 4.29

#### Типичный ход событий варианта использования «Авторизация в приложении»

Действия исполнителя	Отклик системы
1. Пользователь выбрал авторизацию в системе	2. Отображение формы для ввода логина и пароля
3. Пользователь вводит логин и пароль	5. Система проверяет и отправляет данные в базу, выводит сообщение пользователю об успешной авторизации в приложении. 6. Система переходит в режим ожидания

#### Альтернативы типичного хода событий варианта использования «Авторизация в приложении»

5. Если нет соединения с базой данных, то система выдает сообщение об ошибке и переходит к шагу 6.

5. Если введенные данные не удовлетворяют условию полноты, то система выдает сообщение об ошибке и переходит к шагу 6.

#### Дополнительная информация

1. Необходимо обеспечить возможность выхода из варианта на любом этапе.

## 4.2 Построение контекстной диаграммы классов

Контекстные диаграммы классов оперируют понятиями предметной области, атрибутами этих понятий и отношениями между ними [5].

Контекстная диаграмма классов программного обеспечения для мобильных устройств «Помощь на дороге» включает следующие классы-понятия и связи между ними (рис. 4.2): основным классом является класс «Заявка». Полное описание «Заявки» включает «Сообщение», «Графический файл», «Местоположение» устройства. С классом «Заявка» связан класс «Предложение», которое добавляется к заявке на помощь. «Заявки» и «Предложения» хранятся в «Базе данных», их может добавлять «Водитель» (пользователь мобильного устройства). Также «Водитель» может просматривать «Список заявок», состоящий из «Заявок». «Список заявок» можно просмотреть в виде отметок на «Карте», которые может просматривать «Водитель» на своем мобильном устройстве. «Водителю» приходят «Уведомления» о новых заявках из «Базы данных».

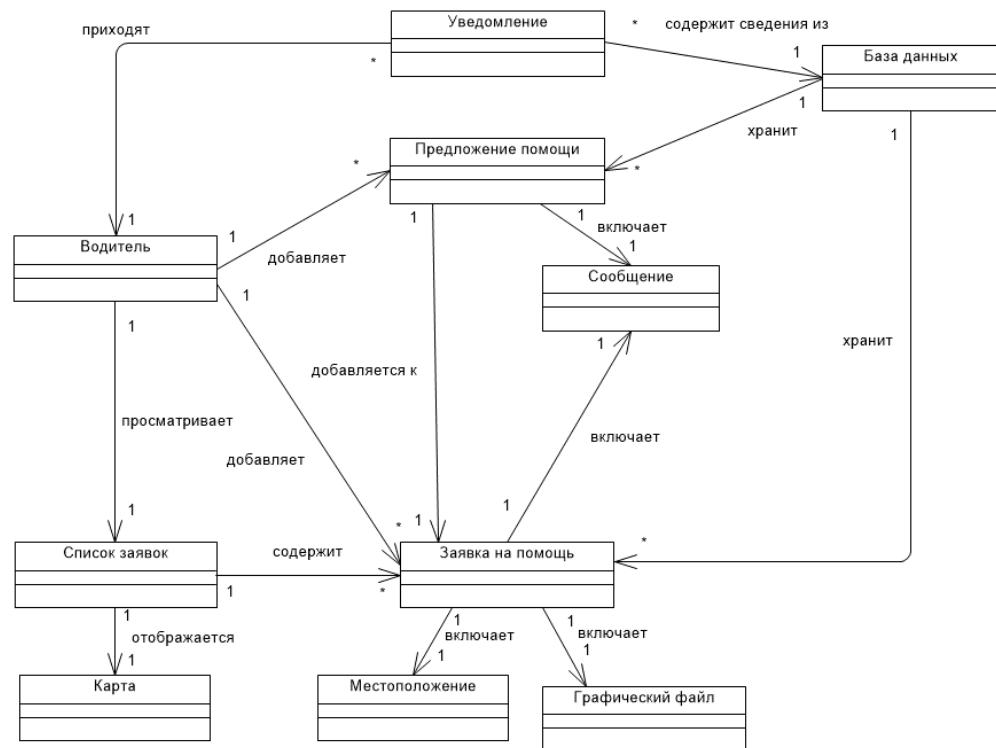


Рис. 4.2. Контекстная диаграмма классов ПО «Помощь на дороге»

### 4.3 Построение диаграмм последовательностей системы

Диаграмма последовательностей системы – графическая модель, которая для определенного сценария варианта использования показывает генерируемые действующими лицами события и их порядок [5, 18].

Диаграмма последовательностей системы для варианта использования «Редактировать данные о пользователе» представлена на рис. 4.3.

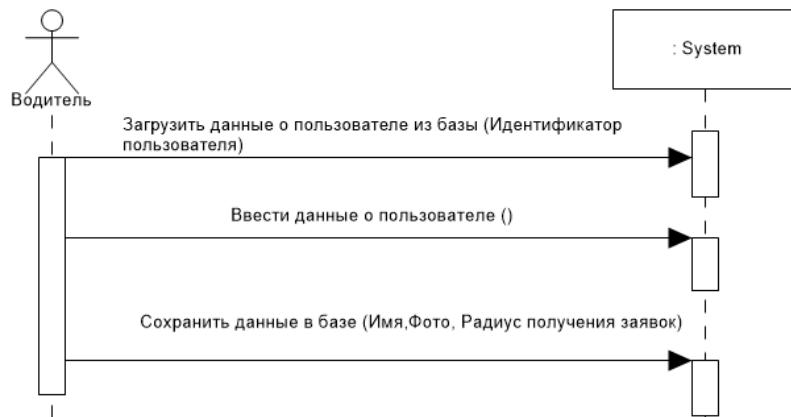


Рис. 4.3. Диаграмма последовательностей системы для варианта использования «Редактировать данные о пользователе»

Системные операции сценария варианта использования «Редактировать данные о пользователе» представлены в табл. 4.30 - 4.32.

Таблица 4.30

#### Описание операции «Загрузить данные о пользователе из базы»

Раздел	Описание
Имя	Загрузить данные о пользователе из базы (Идентификатор пользователя)
Обязанности	Получение информации о пользователе устройства (имя, фото, радиус получения заявок)
Тип	Системная
Ссылки	Вариант использования «Редактировать данные о пользователе»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если нет подключения к базе данных, то вывести сообщение об ошибке
Вывод	Данные о пользователе устройства
Предусловия	Предполагает наличие подключения к базе данных
Постусловие	-

Таблица 4.31

## Описание операции «Ввести данные о пользователе»

Раздел	Описание
Имя	Ввести данные о пользователе ()
Обязанности	Обработать ввод данных о пользователе (Имя, Фото, Радиус получения заявок)
Тип	Системная
Ссылки	Вариант использования «Редактировать данные о пользователе»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если не введено значение имени пользователя, то вывести сообщение об ошибке. Если введены недопустимые данные, то выдать сообщение об ошибке
Вывод	Сообщение о правильном или неправильном вводе данных
Предусловия	Предполагает загрузку данных о пользователе из базы
Постусловие	Предполагает сохранение данных в базу

Таблица 4.32

## Описание операции «Сохранить данные в базе»

Раздел	Описание
Имя	Сохранить данные в базе (Имя, Фото, Радиус получения заявок)
Обязанности	Обработать сохранение данных пользователя в базу
Тип	Системная
Ссылки	Вариант использования «Редактировать данные о пользователе»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если нет подключения к базе, то вывести сообщение об ошибке
Вывод	Сообщение о сохранении данных в базе
Предусловия	Предполагает загрузку данных о пользователе из базы
Постусловие	-

Диаграммы последовательностей системы для варианта использования «Просмотреть заявки» представлены на рис. 4.4 - 4.6.

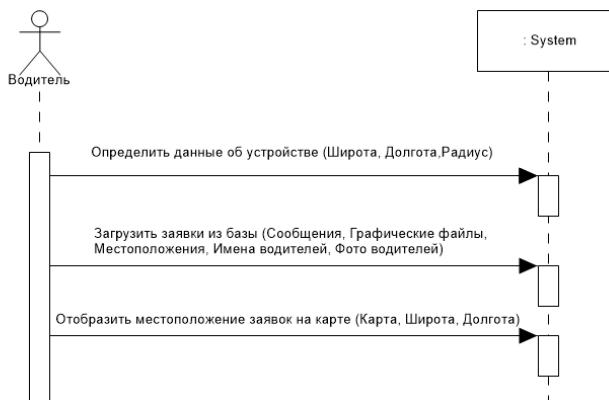


Рис. 4.4. Диаграмма последовательностей системы сценария «Просмотреть на карте» варианта использования «Просмотреть заявки»

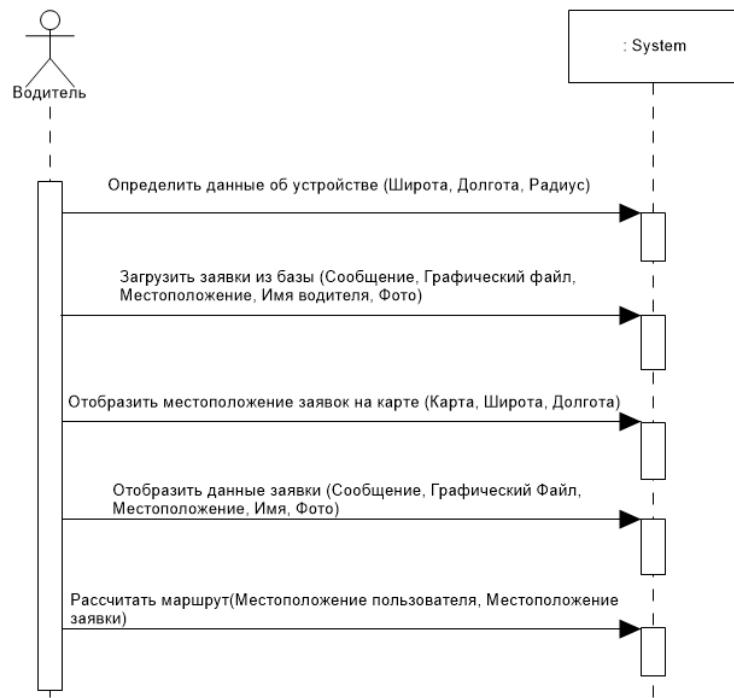


Рис. 4.5. Диаграмма последовательностей системы сценария «Просмотреть заявку и определить маршрут» варианта использования «Просмотреть заявки»

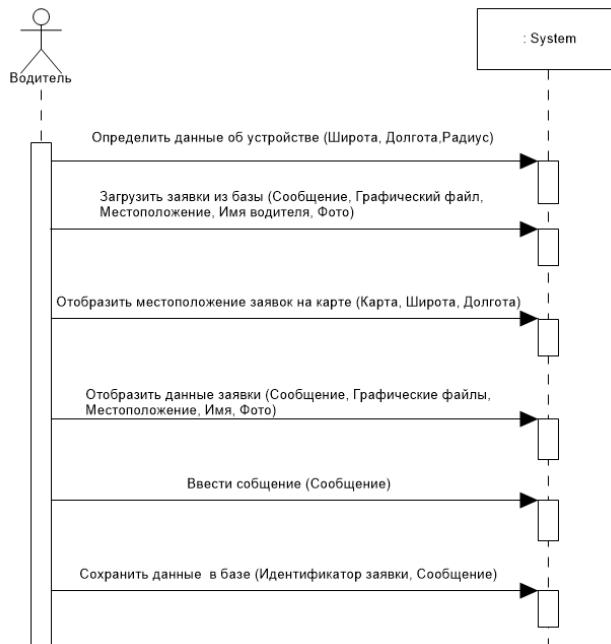


Рис. 4.6. Диаграмма последовательностей системы сценария «Просмотреть заявку и написать сообщение» варианта использования «Просмотреть заявки»

Системные операции сценариев варианта использования «Просмотреть заявки» представлены в табл. 4.33 - 4.40.

Таблица 4.33

## Описание операции «Определить данные об устройстве»

Раздел	Описание
Имя	Определить данные об устройстве (Широта, Долгота, Радиус)
Обязанности	Получение информации о местоположении устройства пользователя и радиусе получения заявок
Тип	Системная
Ссылки	Вариант использования «Просмотреть заявки»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если данные о местоположении недоступны, то вывести сообщение об ошибке
Вывод	Сообщение в случае неуспеха выполнения операции
Предусловия	Предполагает наличие подключенного модуля для определения местоположения
Постусловие	-

Таблица 4.34

## Описание операции «Загрузить заявки из базы»

Раздел	Описание
Имя	Загрузить заявки из базы (Сообщение, Графический файл, Местоположение, Имя водителя, Фото водителя)
Обязанности	Получение списка заявок из базы данных
Тип	Системная
Ссылки	Вариант использования «Просмотреть заявки»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если данные о местоположении недоступны, то вывести сообщение об ошибке. Если нет подключения к базе данных, то вывести сообщение об ошибке
Вывод	Сообщение в случае неуспеха выполнения операции
Предусловия	Предполагает наличие подключения к базе данных. Предполагает наличие подключенного модуля для определения местоположения
Постусловие	Возможность просмотра заявок пользователем

Таблица 4.35

## Описание операции «Отобразить местоположение заявок на карте»

Раздел	Описание
1	2
Имя	Отобразить местоположение заявок на карте (Карта, Широта, Долгота)
Обязанности	Отображение списка отметок о местоположении заявок
Тип	Системная
Ссылки	Вариант использования «Просмотреть заявки»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если данные о местоположении недоступны, то вывести сообщение об ошибке.

## Продолжение табл. 4.35

1	2
Вывод	Сообщение в случае неуспеха выполнения операции. Карта с отметками заявок
Предусловия	Предполагает наличие подключенного модуля для определения местоположения. Предполагает предварительную загрузку заявок из базы
Постусловие	-

Таблица 4.36

## Описание операции «Отобразить данные заявки»

Раздел	Описание
Имя	Отобразить данные заявки (Сообщение, Графический Файл, Местоположение, Имя, Фото)
Обязанности	Отображение данных о заявке
Тип	Системная
Ссылки	Вариант использования «Просмотреть заявку»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	-
Вывод	Список заявок
Предусловия	Предполагает наличие подключенного модуля для определения местоположения. Предполагает предварительную загрузку заявок из базы
Постусловие	-

Таблица 4.37

## Описание операции «Рассчитать маршрут»

Раздел	Описание
Имя	Рассчитать маршрут (Местоположение пользователя, Местоположение заявки)
Обязанности	Вычисление маршрута от пользователя до заявки
Тип	Системная
Ссылки	Вариант использования «Определить маршрут»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если данные о местоположении недоступны, то вывести сообщение об ошибке. Если нет подключения к сети интернет, то вывести сообщение об ошибке
Вывод	Сообщение в случае неуспеха выполнения операции. Карта с отметками заявок
Предусловия	Предполагает наличие подключенного модуля для определения местоположения. Предполагает наличие подключения к сети интернет
Постусловие	-

Таблица 4.38

## Описание операции «Ввести сообщение»

Раздел	Описание
1	2
Имя	Ввести сообщение (Сообщение)

Продолжение табл. 4.38

1	2
Обязанности	Обработка ввода сообщения пользователем
Тип	Системная
Ссылки	Вариант использования «Написать сообщение»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Исключение, если не введено сообщение
Вывод	Сообщение в случае неуспеха выполнения операции
Предусловия	-
Постусловие	Отправка данных сообщения в базу

Таблица 4.40

## Описание операции «Сохранить данные в базе»

Раздел	Описание
Имя	Сохранить данные в базе (Идентификатор заявки, Сообщение)
Обязанности	Сохранение сообщения в базе
Тип	Системная
Ссылки	Вариант использования «Написать сообщение»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Исключение, если нет подключения к базе
Вывод	Сообщение в случае неуспеха выполнения операции. Сообщение об успешном сохранении сообщения в базе
Предусловия	Загрузка из базы данных о заявке
Постусловие	-

Диаграмма последовательностей системы варианта использования «Получить уведомления» представлена на рис. 4.7.

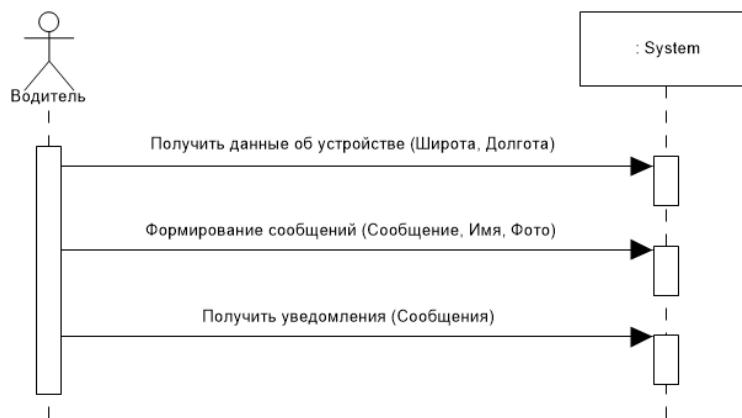


Рис. 4.7. Диаграмма последовательностей системы для варианта использования «Получить уведомления»

Системные операции варианта использования «Получить уведомления» представлены в табл. 4.41 - 4.43.

Таблица 4.41

## Описание операции «Получить данные об устройстве»

Раздел	Описание
Имя	Получить данные об устройстве (Широта, Долгота)
Обязанности	Получение данных об устройстве, определение местоположения устройства
Тип	Системная
Ссылки	Вариант использования «Получить уведомления»
Примечание	-
Исключения	В случае невозможности получения данные об устройстве
Вывод	Сообщение в случае неуспеха выполнения операции
Предусловия	Предполагает наличие подключенного модуля для определения местоположения
Постусловие	Формирование сообщений

Таблица 4.42

## Описание операции «Формирование сообщений»

Раздел	Описание
Имя	Формирование сообщений (Сообщение, Имя, Фото)
Обязанности	Сформировать список сообщений, которые будут переданы уведомлениями пользователю
Тип	Системная
Ссылки	Вариант использования «Получить уведомления»
Примечание	-
Исключения	Если список сообщений пуст
Вывод	-
Предусловия	-
Постусловие	Отправка уведомлений на устройство

Таблица 4.43

## Описание операции «Получить уведомления»

Раздел	Описание
Имя	Получить уведомления (Сообщения)
Обязанности	Отобразить полученные уведомления на экране устройства
Тип	Системная
Ссылки	Вариант использования «Получить уведомления»
Примечание	-
Исключения	-
Вывод	Список уведомлений
Предусловия	Проверка подключения к сети интернет
Постусловие	-

Диаграммы последовательностей системы варианта использования «Добавить заявку» представлены на рис. 4.8, 4.9.

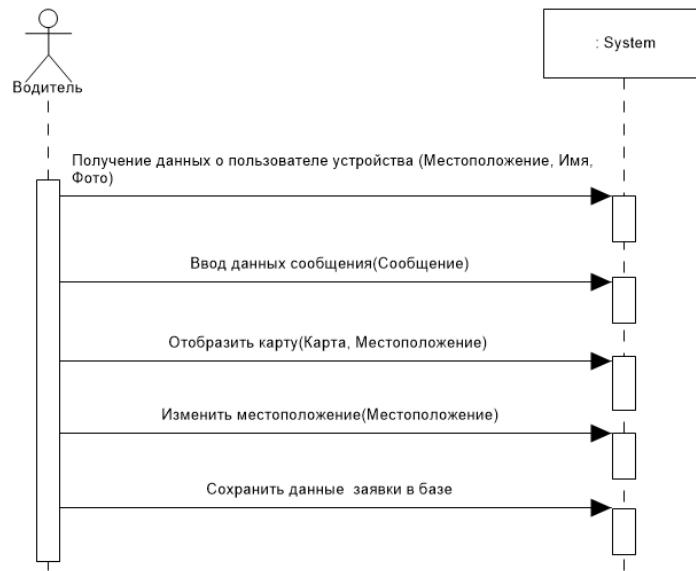


Рис. 4.8. Диаграмма последовательностей системы для сценария «Добавить сообщение и изменить местоположение» варианта использования «Добавить заявку»

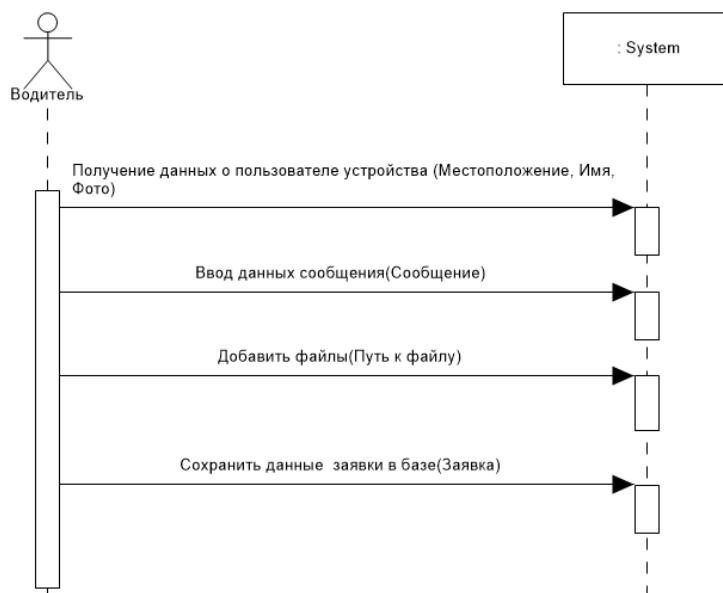


Рис. 4.9. Диаграмма последовательностей системы для сценария «Добавить сообщение и графические файлы» варианта использования «Добавить заявку»

Системные операции варианта использования «Добавить заявку» представлены в табл. 4.44 - 4.49.

Таблица 4.44

## Описание операции «Получение данных о пользователе устройства»

Раздел	Описание
Имя	Получение данных о пользователе устройства (Местоположение, Имя, Фото)
Обязанности	Вернуть данные о пользователе
Тип	Системная
Ссылки	Вариант использования «Добавить заявку»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если данные о местоположении недоступны, то вывести сообщение об ошибке
Вывод	Сообщение в случае неуспеха выполнения операции
Предусловия	Предполагает наличие подключенного модуля для определения местоположения
Постусловие	-

Таблица 4.45

## Описание операции «Ввод данных сообщения»

Раздел	Описание
Имя	Ввод данных сообщения (Сообщение)
Обязанности	Добавить сообщение
Тип	Системная
Ссылки	Вариант использования «Добавить сообщение»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если пользователь не ввел сообщение, то вывести сообщение об ошибке
Вывод	Сообщение в случае неуспеха выполнения операции
Предусловия	Предполагает наличие подключенного модуля для определения местоположения
Постусловие	Сохранение данных в базе

Таблица 4.46

## Описание операции «Отобразить карту»

Раздел	Описание
Имя	Отобразить карту (Карта, Местоположение)
Обязанности	Показать карту с отметкой местоположения пользователя
Тип	Системная
Ссылки	Вариант использования «Добавить графические файлы»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если данные о местоположении недоступны, то вывести сообщение об ошибке
Вывод	Сообщение в случае неуспеха выполнения операции
Предусловия	Предполагает наличие подключенного модуля для определения местоположения
Постусловие	-

Таблица 4.47

## Описание операции «Изменить местоположение»

Раздел	Описание
Имя	Изменить местоположение (Карта, Местоположение)
Обязанности	Изменить данные о местоположении пользователя
Тип	Системная
Ссылки	Вариант использования «Указать местоположение»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если данные о местоположении недоступны, то вывести сообщение об ошибке
Вывод	Сообщение в случае неуспеха выполнения операции
Предусловия	Предполагает наличие подключенного модуля для определения местоположения. Должна быть отображена карта
Постусловие	-

Таблица 4.48

## Описание операции «Добавить файлы»

Раздел	Описание
Имя	Добавить файлы (Путь к файлу)
Обязанности	Выбрать файлы для добавления
Тип	Системная
Ссылки	Вариант использования «Добавить графические файлы»
Примечания	Предусмотреть выбор файлов только графических форматов: .jpg, .png,. Предусмотреть возможность прерывания операции
Исключения	-
Вывод	Сообщение в случае неуспеха выполнения операции
Предусловия	-
Постусловие	-

Таблица 4.49

## Описание операции «Сохранить данные заявки в базе»

Раздел	Описание
Имя	Сохранить данные заявки в базе (Заявка)
Обязанности	Добавляет данные о заявке в базу
Тип	Системная
Ссылки	Вариант использования «Добавить заявку»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если отсутствует подключение к БД, то вывести сообщение об ошибке. Сообщение о добавлении данных заявки
Вывод	Сообщение в случае неуспеха выполнения операции
Предусловия	-
Постусловие	Предполагает наличие введенных данных заявки

Диаграмма последовательностей системы варианта использования «Просмотреть заявки пользователя» представлена на рис. 4.10.

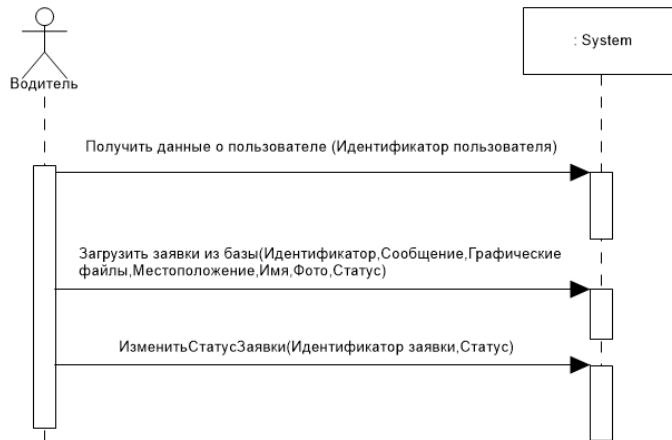


Рис. 4.10. Диаграмма последовательностей системы сценария «Изменить статус заявки» варианта использования «Просмотреть заявки пользователя»

Системные операции сценария варианта использования «Просмотреть заявки пользователя» представлены в табл. 4.50 – 4.52.

Таблица 4.50

#### Описание операции «Получение данных о пользователе»

Раздел	Описание
Имя	Получение данных о пользователе (Идентификатор пользователя)
Обязанности	Вернуть данные о пользователе устройства
Тип	Системная
Ссылки	Вариант использования «Просмотреть заявки пользователя»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если данные о местоположении недоступны, то вывести сообщение об ошибке
Вывод	Сообщение в случае неуспеха выполнения операции
Предусловия	Предполагает наличие подключенного модуля для определения местоположения
Постусловие	-

Таблица 4.51

#### Описание операции «Загрузить заявки из базы»

Раздел	Описание
1	2
Имя	Загрузить заявки из базы (Идентификатор пользователя, Сообщение, Графические файлы, Местоположение, Имя, Фото, Радиус)
Обязанности	Получение списка заявок из базы данных
Тип	Системная
Ссылки	Вариант использования «Просмотреть заявки»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если данные о местоположении недоступны, то вывести сообщение об ошибке. Если нет подключения к базе данных, то вывести сообщение об ошибке
Вывод	Сообщение в случае неуспеха выполнения операции

1	2
Предусловия	Предполагает наличие подключения к базе данных и подключенного модуля для определения местоположения
Постусловие	Возможность просмотра заявок пользователем

Таблица 4.52

## Описание операции «Изменить статус заявки»

Раздел	Описание
Имя	Изменить статус заявки (Идентификатор заявки, Статус)
Обязанности	Изменение статуса выбранной пользователем заявки
Тип	Системная
Ссылки	Вариант использования «Изменить статус заявки»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Сообщение в случае отсутствия подключения к базе
Вывод	Сообщение об успешном изменении статуса заявки
Предусловия	Предполагает наличие подключения к базе данных
Постусловие	-

Диаграмма последовательностей системы варианта использования «Регистрация в приложении» представлена на рис. 4.11.

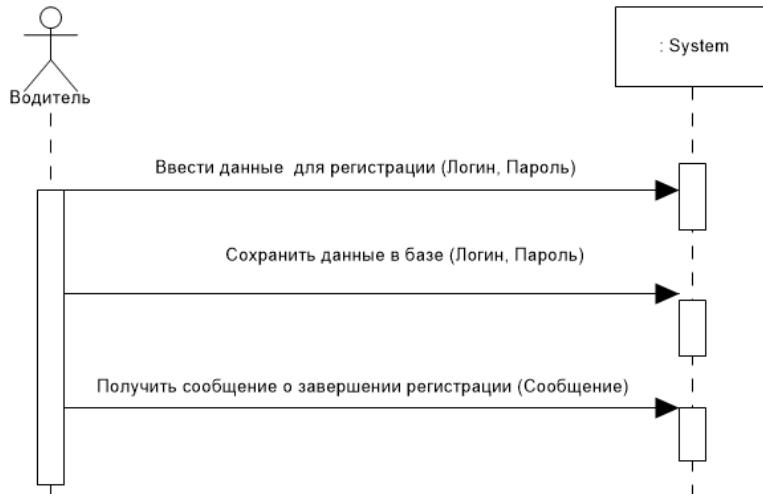


Рис. 4.11. Диаграмма последовательностей системы для варианта использования «Регистрация в приложении»

Системные операции сценария варианта использования «Регистрация в приложении» представлены в табл. 4.53 – 4.55.

Таблица 4.53

## Описание операции «Ввести данные для регистрации»

Раздел	Описание
Имя	Ввести данные для регистрации (Логин, Пароль)
Обязанности	Проверить ввод логина и пароля
Тип	Системная
Ссылки	Вариант использования «Регистрация в приложении»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если введены не все данные, то выдать сообщение об ошибке
Вывод	Сообщение в случае неуспеха выполнения операции
Предусловия	Предполагает наличие подключения к базе
Постусловие	Сохранение логина и пароля в базу

Таблица 4.54

## Описание операции «Сохранить данные в базе»

Раздел	Описание
Имя	Сохранить данные в базе (Логин, Пароль)
Обязанности	Сохранение логина и пароля в базе
Тип	Системная
Ссылки	Вариант использования «Регистрация в приложении»
Примечания	-
Исключения	В случае неполных или повторяющихся данных
Вывод	Сообщение в случае неуспеха выполнения операции
Предусловия	Предполагает наличие подключения к базе данных
Постусловие	Получение сообщения пользователем

Таблица 4.55

## Описание операции «Получить сообщение о завершении регистрации

Раздел	Описание
Имя	Получить сообщение о завершении регистрации (Сообщение)
Обязанности	Подтверждение регистрации
Тип	Системная
Ссылки	Вариант использования «Регистрация в приложении»
Примечания	-
Исключения	В случае неуспешной регистрации
Вывод	Сообщение о завершении процесса регистрации
Предусловия	Предполагает успешную запись данных пользователя в базу
Постусловие	-

Диаграмма последовательностей системы варианта использования «Авторизация в приложении» представлена на рис. 4.12.

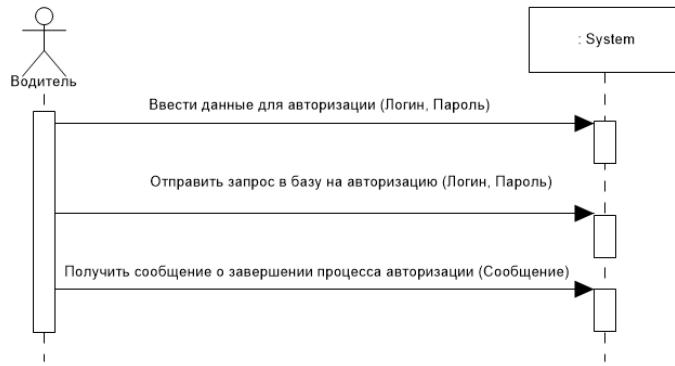


Рис. 4.12. Диаграмма последовательностей системы для варианта использования «Авторизация в приложении»

Системные операции сценария варианта использования «Авторизация в приложении» представлены в табл. 4.56 – 4.58.

Таблица 4.56

#### Описание операции «Ввести данные для авторизации»

Раздел	Описание
Имя	Ввести данные для авторизации (Логин, Пароль)
Обязанности	Проверить ввод логина и пароля
Тип	Системная
Ссылки	Вариант использования «Авторизация в приложении»
Примечания	Предусмотреть возможность прерывания операции пользователем
Исключения	Если введены не все данные, то выдать сообщение об ошибке
Вывод	Сообщение в случае неуспеха выполнения операции
Предусловия	Предполагает наличие подключения к базе
Постусловие	Запрос к базе для получения идентификатора пользователя

Таблица 4.57

#### Описание операции «Сохранить данные в базе»

Раздел	Описание
Имя	Отправить запрос в базу на авторизацию (Логин, Пароль)
Обязанности	Проверка наличия идентификатора пользователя в базе
Тип	Системная
Ссылки	Вариант использования «Авторизация в приложении»
Примечания	-
Исключения	В случае отсутствия подключения к базе
Вывод	Идентификатор пользователя
Предусловия	Предполагает наличие подключения к базе данных
Постусловие	Выход сообщения о завершении авторизации в системе

Таблица 4.58

## Описание операции «Получить сообщение о завершении авторизации»

Раздел	Описание
Имя	Получить сообщение о завершении процесса авторизации (Сообщение)
Обязанности	Подтверждение авторизации
Тип	Системная
Ссылки	Вариант использования «Авторизация в приложении»
Примечания	-
Исключения	В случае неуспешной авторизации
Вывод	Сообщение о завершении процесса авторизации
Предусловия	Предполагает успешную авторизацию в базе
Постусловие	-

#### 4.4 Проектирование структур данных и построение диаграмм отношений компонентов данных

Для хранения данных заявок водителей на помощь требуется спроектировать модель двух реляционных баз данных:

- для хранения данных на удаленном сервере (для серверной части приложения);
- для хранения данных на устройстве пользователя (для клиентской части приложения).

На данном этапе необходимо выполнить инфологическое проектирование базы данных, которое заключается в анализе предметной области, определении основных понятий: сущностей, атрибутов сущностей и связей между ними. Результатом данного этапа является проектирование диаграммы «сущность-связь» (entity-relationship diagram) [3, 15].

##### 4.4.1 Логическое проектирование базы данных серверной части приложения

На этапе логического проектирования базы данных серверной части приложения «Помощь на дороге» выполнен анализ предметной области «Сообщения водителей о помощи» и выделены сущности и атрибуты сущностей (табл. 4.59), которые описывают информацию, которая будет хранится на сервере.

Таблица 4.59

## Сущности предметной области «Сообщения водителей о помощи»

Сущность	Описание	Атрибуты
Пользователь	Показывает информацию о пользователях приложения, которые оставляют заявки на помощь и добавляют сообщения готовности оказать помощь (предложения помощи)	Идентификатор пользователя, логин, пароль, имя, изображение (имя файла с изображением пользователя), каталог (имя каталога на сервере для хранения графических файлов пользователя), дата добавления, дата изменения
Заявка	Отражает информацию о заявках на помощь, которые оставляют пользователи приложения	Идентификатор заявки, сообщение, изображение (имя файла с изображением), местоположение (широта, долгота), дата добавления, дата изменения, идентификатор пользователя, идентификатор статуса заявки
Статус	Сведения о статусе заявки: «требуется помощь», «помощь оказывается», «помощь оказана»	Идентификатор статуса заявки, значение статуса заявки
Предложение	Показывает информацию о предложениях помощи пользователей	Идентификатор предложения, текст сообщения, дата добавления, дата изменения, идентификатор пользователя, идентификатор заявки

Между данными сущностями имеются связи, которые показаны на ER-диаграмме (рис. 4.13), отражающей логическую модель предметной области:

- 1) связь «добавляет» мощностью «один-ко-многим» между сущностями «Пользователь» и «Заявка»: каждый пользователь добавляет ноль и более заявок на помощь. Каждая заявка на помощь может быть добавлена только одним пользователем;
- 2) связь «добавляет» мощностью «один-ко-многим» между сущностями «Пользователь» и «Предложение»: каждый пользователь добавляет в систему ноль и более предложений помощи. Каждое предложение помощи может быть добавлено только одним пользователем;
- 3) связь «относится к» мощностью «один-ко-многим» между сущностями «Заявка» и «Предложение»: для каждой заявки может быть добавлено ноль и более предложений помощи. Каждое предложение помощи относится только к одной заявке;
- 4) связь «имеет» мощностью «один-ко-многим» между сущностями «Заявка» и «Статус»: каждая заявка может иметь только один статус.

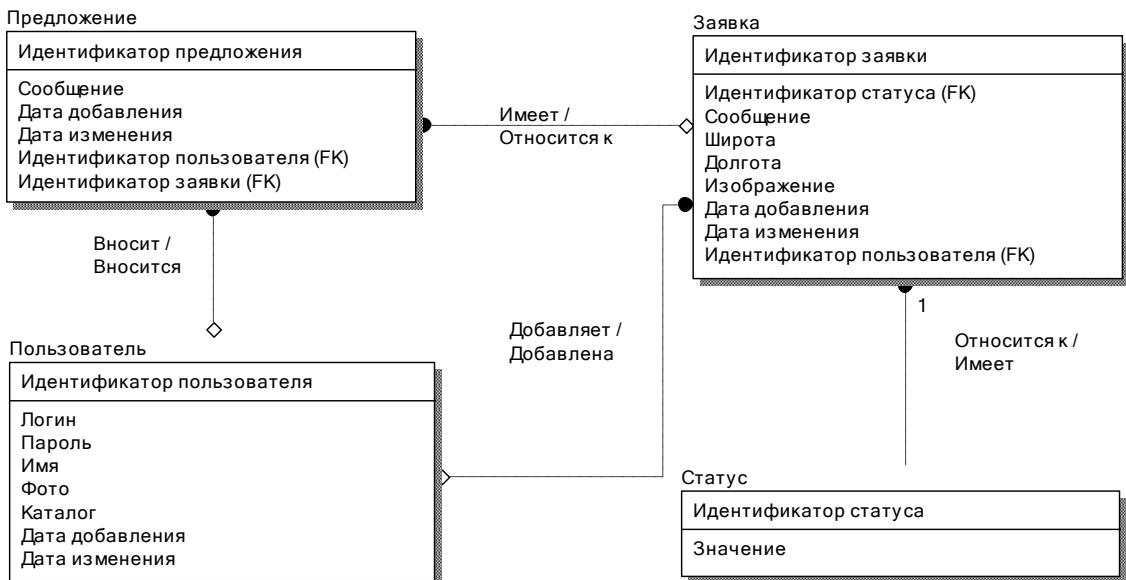


Рис. 4.13. ER-диаграмма, представляющая логическую модель предметной области «Сообщения водителей о помощи» (для серверной части)

#### 4.4.2 Логическое проектирование базы данных клиентской части

На этапе логического проектирования базы данных клиентской части приложения «Помощь на дороге» выполнен анализ предметной области «Сообщения водителей о помощи» и выделены сущности, атрибуты сущностей, которые описывают информацию, которая будет храниться на мобильном устройстве пользователя (табл. 4.60).

Таблица 4.60

Сущности предметной области «Сообщения водителей о помощи»

Сущность	Описание	Атрибуты
1	2	3
Пользователь	Показывает информацию о пользователях приложения, которые оставляют заявки на помощь и добавляют сообщения готовности оказать помощь	Идентификатор пользователя, имя, наличие файла изображения пользователя, дата добавления, дата изменения, дата загрузки на устройство
Заявка	Отражает информацию о заявках на помощь, которые оставляют пользователи приложения	Идентификатор заявки, сообщение, местоположение: широта, долгота, наличие файла изображения, дата добавления, дата изменения, дата загрузки на устройство, заявка является новой (логическое значение), уведомление получено (логическое значение), идентификатор пользователя, статус заявки
Предложение	Показывает информацию о предложениях помощи	Идентификатор предложения, текст сообщения, дата добавления, дата изменения, дата загрузки, предложение является новым

1	2	3
		(логическое значение), уведомление получено (логическое значение), идентификатор пользователя, идентификатор заявки

Между сущностями имеются отношения, показанные на ER-диаграмме (рис. 4.14), отражающей логическую модель предметной области

- 1) связь «добавляет» мощностью «один-ко-многим» между сущностями «Пользователь» и «Заявка»: каждый пользователь добавляет ноль и более заявок на помощь. Каждая заявка может быть добавлена только одним пользователем;
- 2) связь «добавляет» мощностью «один-ко-многим» между сущностями «Пользователь» и «Предложение»: каждый пользователь добавляет в систему ноль и более предложений помощи. Каждое предложение может быть добавлено только одним пользователем;
- 3) связь «относится к» мощностью «один-ко-многим» между сущностями «Заявка» и «Предложение»: для каждой заявки добавляется ноль и более предложений помощи. Каждое предложение относится только к одной заявке.

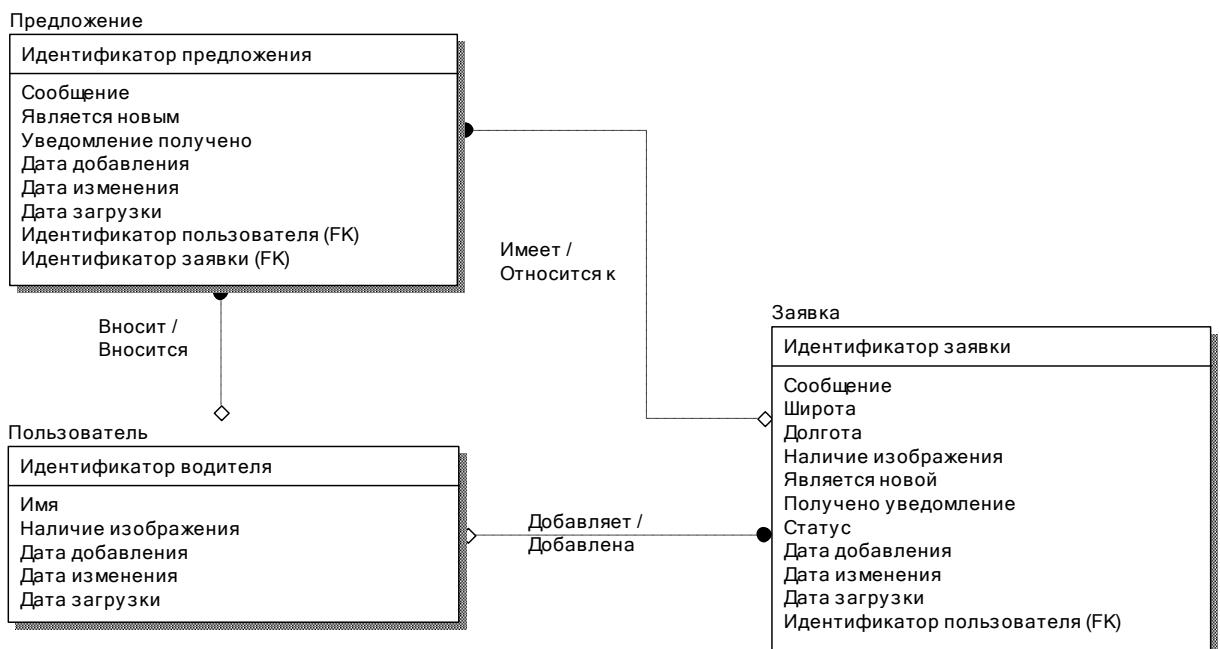


Рис. 4.14. ER-диаграмма, представляющая логическую модель предметной области «Сообщения водителей о помощи» (для клиентской части)

## 5 Проектирование системы

### 5.1 Проектирование структуры системы и построение диаграмм пакетов

Диаграмма пакетов показывает то, из каких частей состоит программное обеспечение, и как эти части связаны друг с другом. Связь между пакетами фиксируют, если изменения в одном пакете могут повлечь за собой изменения в другом. Она определяется внешними связями классов и других ресурсов, объединенных в пакет [5, 19].

Программное обеспечение для мобильных устройств «Помощь на дороге» включает пакеты, описание которых содержится в табл. 5.1, в которой представлены пакеты серверной и клиентской частей приложения, пакеты, являющиеся библиотеками классов системы Android, которые используются для реализации пакетов клиентской части приложения.

Таблица 5.1

Пакеты ПО для мобильных устройств «Помощь на дороге»

Название пакета	Описание
1	2
Пакеты серверной части приложения	
База данных MySQL	База данных MySQL, расположенная на удаленном сервере
Интерфейс к базе данных MySQL (MySQL data base interface)	Классы серверной части приложения, реализующие интерфейс к базе данных MySQL
Интерфейс серверной части приложения	Пакет серверной части приложения, содержащий набор скриптов на языке PHP, использующих методы классов пакета «Интерфейс к базе данных MySQL». Скрипты вызываются клиентской частью приложения, посредством протокола HTTP, для получения и записи данных в базу данных MySQL
Пакеты клиентской части приложения	
Графические ресурсы (Resources)	Набор файлов формата .xml, описывающих разметку окон пользовательского интерфейса, строковые и графические ресурсы
Объекты модели (Object of model)	Классы, реализующие объекты предметной области мобильного приложения «Помощь на дороге» (заявка пользователя на помощь, водитель, пользователь, предложение помощи)
Управление объектами модели (Management object of model)	Классы, предназначенные для работы с объектами предметной области
JSON-парсеры (JSON-parsers)	Классы, предназначенные для заполнения объектов предметной области данными, переданными серверной частью в формате JSON
Сетевые операции» (Network operation)	Классы, реализующие сетевые операции протокола HTTP: GET, POST-запросы

## Продолжение табл. 5.1

1	2
База данных SQLite	База данных SQLite клиентской части приложения
Интерфейс к базе данных SQLite (SQLite data base interface)	Классы, реализующие интерфейс для взаимодействия с базой данных SQLite клиентской части приложения
Управление местоположением (Location manager)	Классы, реализующие взаимодействие со службами Google Play Services для определения местоположения устройства пользователя
Управление программой (Application controller)	Пакет, который содержит вложенные пакеты, классы которых реализуют варианты использования. Включает пакеты: «Управление добавлением заявок», «Управление пользователем», «Управление списком заявок»
Управление добавлением заявок (Management add new request)	Классы, реализующие добавление новой заявки на помощь в приложение
Управление списком заявок (Management list request)	Классы, реализующие формирование списка заявок на помощь, просмотр деталей заявки на помощь и добавление предложений помощи к заявке
Управление пользователем (Management user)	Классы, реализующие добавление нового пользователя в систему, регистрацию, авторизацию и просмотр учетной записи пользователя
Библиотеки системы Android	
Библиотека обработки ошибок (global)	Библиотека классов исключений, реализующих обработку нештатных ситуаций
Библиотека структур данных (global)	Библиотека классов, реализующих внутренние структуры данных, такие, как деревья, n-связные списки и т. п.
API системы Android	Библиотека классов для взаимодействия с операционной системой Android
Библиотека поддержки» (Support library)	Библиотека классов для написания приложения, совместимого с различными версиями операционной системы Android
Библиотека для работы с JSON	Библиотека классов для работы с JSON-форматом
Библиотека сетевых операций	Библиотека классов для выполнения сетевых операций по протоколу HTTP/HTTPS
Библиотека сервисов Google	Библиотека классов приложения Google Play Services, реализующих операции: определение местоположения устройства, построение карты Google, работа с уведомлениями
Библиотека Picasso	Библиотека для работы с изображениями в системе Android (загрузка, кэширование изображений)

Также в процессе разработки структуры программного обеспечения для мобильных устройств «Помощь на дороге» были уточнены связи (зависимости) между проектируемыми пакетами.

Описание зависимостей между пакетами программного обеспечения для мобильных устройств «Помощь на дороге» представлено в табл. 5.2.

Диаграмма пакетов проектируемой системы представлена на рис. 5.1.

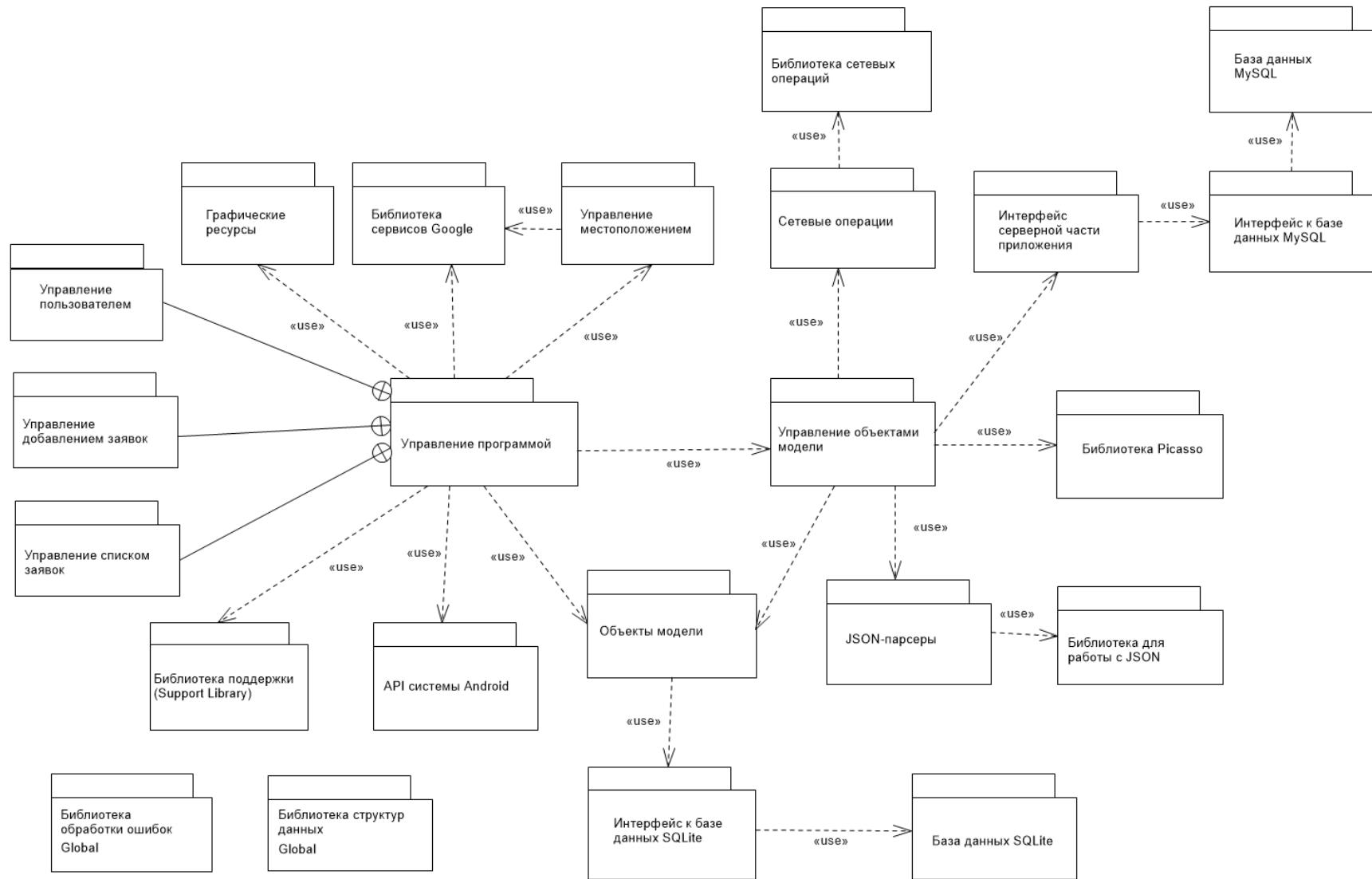


Рис. 5.1. Диаграмма пакетов программного обеспечения для мобильных устройств «Помощь на дороге»

Таблица 5.2

## Зависимости между пакетами ПО «Помощь на дороге»

Пакет	Описание зависимостей
Объекты модели	Классы пакета «Объекты модели» используют классы пакета «Интерфейс к базе данных SQLite» для заполнения объектов модели данными из базы SQLite клиентской части приложения
Управление объектами модели	Использует классы пакетов: «Объекты модели», «Сетевые операции», «JSON-парсеры», «Библиотека Picasso» - для заполнения объектов модели данными, полученными от серверной части приложения. Вызывает скрипты серверного пакета «Интерфейс серверной части приложения» посредством HTTP-запросов
Управление местоположением	Использует классы «Библиотеки сервисов Google» для определения текущего местоположения устройства пользователя
Сетевые операции	Использует классы пакета «Библиотека сетевых операций»
Управление программой	Включает пакеты: «Управление пользователем», «Управление добавлением заявок», «Управление списком заявок». Классы указанных пакетов используют библиотеки: «Библиотека системы Android», «Библиотека поддержки», «Графические ресурсы» - для реализации работы пользовательского интерфейса и обработки взаимодействия пользователя с ним

## 5.2 Проектирование классов пакета «Объекты модели»

## 5.2.1 Построение исходной диаграммы классов

Список классов-кандидатов пакета «Объекты модели», полученный на основе анализа предметной области, представляет следующие классы (табл. 5.3).

Таблица 5.3

## Классы пакета «Объекты модели»

Класс	Описание
1	2
Request (Заявка)	Объекты данного класса описывают заявку пользователя на помощь, включающую: текст сообщения, наличие изображения, местоположение: широту, долготу, сведения о пользователе, дату добавления и обновления в системе
Status (Статус)	Объекты данного класса характеризуют статус заявки в системе, который принимает значения: «требуется помощь», «помощь оказывается», «помощь оказана»
Offer (Предложение)	Объекты данного класса описывают предложение помощи к заявке, включающее текст сообщения, сведения о пользователе, дату добавления и обновления в системе
Driver (Водитель)	Объекты данного класса предназначены для работы с данными о пользователях, которые добавляют заявки. Каждого пользователя, добавляющего заявку, характеризует имя, наличие фотографии

## Продолжение табл. 5.3

1	2
User (Пользователь)	Объекты класса «User» предназначены для работы с данными о пользователе устройства. Каждого пользователя характеризует имя, наличие фотографии, логин, пароль, дата регистрации и изменения учетной записи
RequestList (Список заявок)	Объекты данного класса описывают список добавленных в систему заявок на помощь
OfferList (Список предложений помощи)	Объекты данного класса описывают список добавленных в систему предложений помощи к выбранной заявке на помощь

Указанные классы имеют зависимости, представленные табл. 5.4.

Таблица 5.4

## Зависимости между классами пакета «Объекты модели»

Класс	Описание зависимостей
Driver	Связан ассоциативными отношениями «добавляет» с классами: «Request», «Offer». Каждая заявка и предложение содержат сведения о водителе, который их добавил в систему
User	Наследуется от класса «Driver», поскольку каждый пользователь устройства является пользователем, который добавляет заявки и предложения в систему
Status	Связан ассоциативным отношением «имеется у» с классом «Request»: каждая добавленная заявка характеризуется определенным статусом
Request	Связан с классом «RequestList», который содержит список заявок, ассоциативным отношением «входит в состав»
Offer	Связан с классом «OfferList», который содержит список предложений помощи, ассоциативным отношением «входит в состав»

Исходная диаграмма классов проектируемого пакета «Объекты модели» представлена на рис. 5.2.

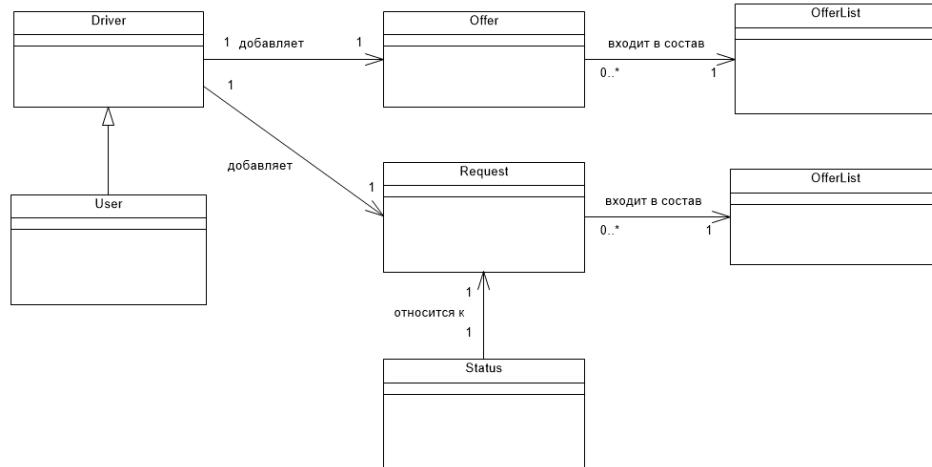


Рис. 5.2. Исходная диаграмма классов пакета «Объекты модели»

### 5.2.2 Построение диаграмм последовательностей действий

Диаграммы последовательностей этапа проектирования отображают взаимодействие объектов, упорядоченное по времени. На диаграмме показывают взаимодействующие объекты, последовательность сообщений, которыми они обмениваются в процессе реализации сценария варианта использования [5].

Для реализации сценария «Добавление новой заявки» (ввод данных сообщения заявки пользователем, уточнение местоположения пользователя и передача данных заявки серверной части для сохранения в базе данных MySQL) построены диаграммы последовательностей действий, которые представлены на рис. 5.3 - 5.5 (для типичного хода событий, прерывания выполнения сценария пользователем и возникновения исключительной ситуации).

Указанные диаграммы показывают последовательность обмена сообщениями объектов: «Request», «User», «Driver», «Status» - пакета «Объекты модели» («Object of model»); объекта класса «RequestManager» («Управление заявками»), который управляет процессом отправки данных заявки серверной части приложения, пакета «Management objects of model» («Управление объектами модели»). Управляет рассматриваемым сценарием объект класса «AddNewRequestActivity» («Активность для добавления новой заявки»).

Сценарий заключается в следующем: пользователь запускает активность (окно пользовательского интерфейса) «AddNewRequestActivity», при создании которой заполняется объект «User» данными о пользователе из файла настроек приложения. Затем происходит заполнение объекта «Request»: пользователь вводит сообщение (message), устройством определяется местоположение пользователя (location). В процессе выполнения сценария происходит создание объекта «Driver» из объекта «User». Когда пользователь дает сообщение - отправить данные заявки, создается объект «AddNewRequestTask» («Задание для отправки данных заявки»), который посредством объекта «RequestManager» осуществляет преобразование данных объекта «Request» в JSON-формат и передает их по протоколу HTTP серверной части приложения.

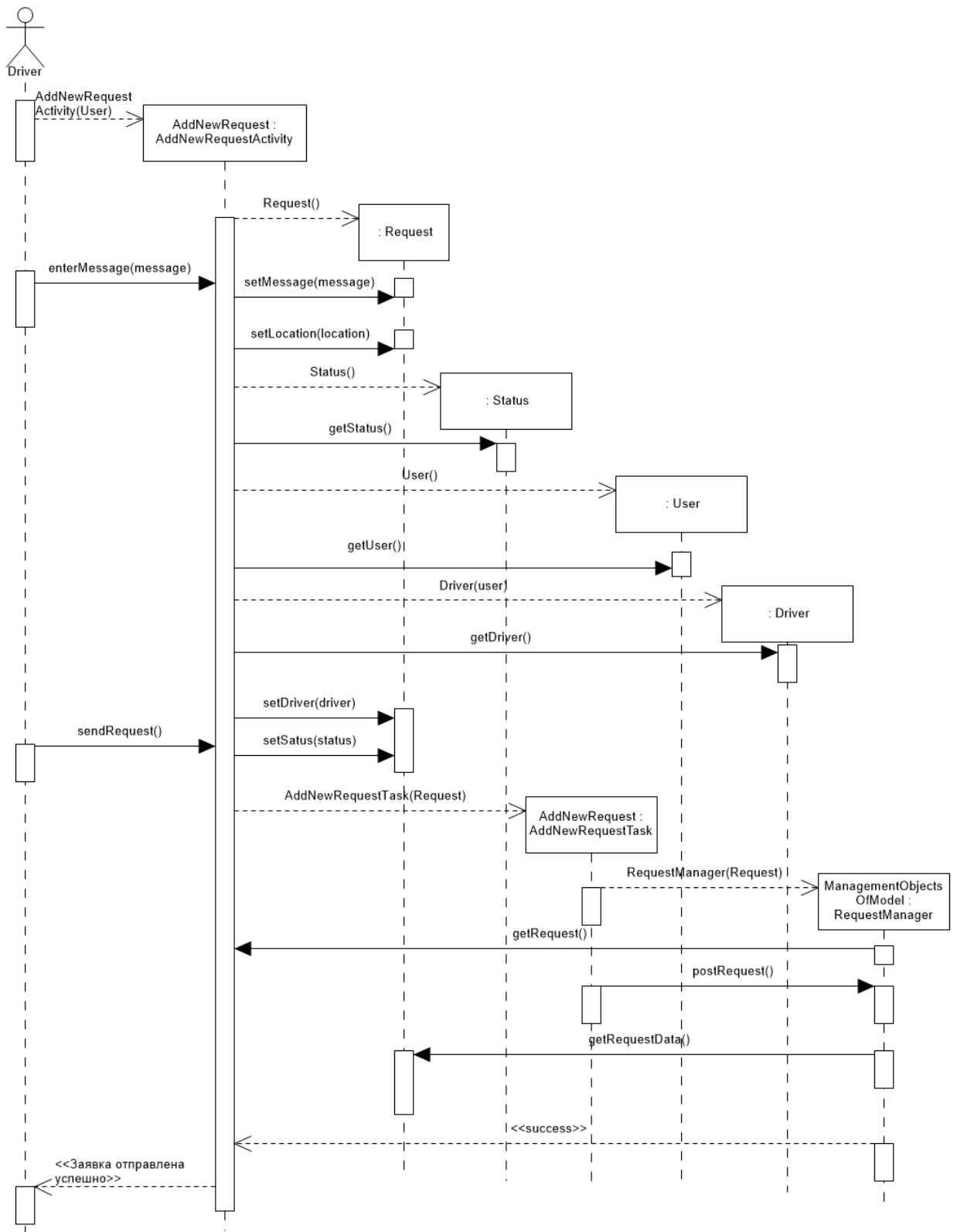


Рис. 5.3. Диаграмма последовательностей действий сценария «Добавление новой заявки». Типичный ход событий

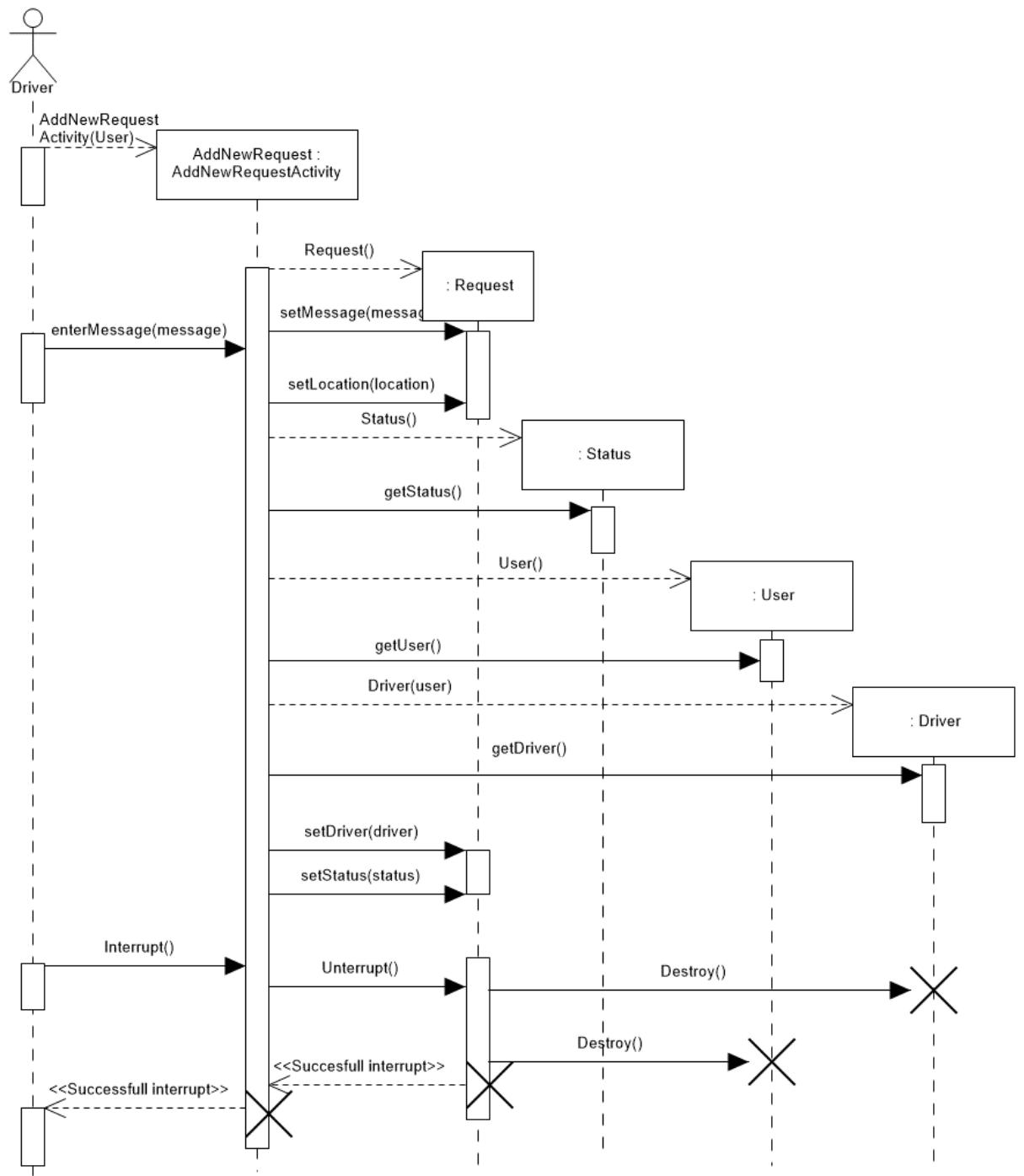


Рис. 5.4. Диаграмма последовательностей действий сценария «Добавление новой заявки». Прерывание выполнения сценария пользователем

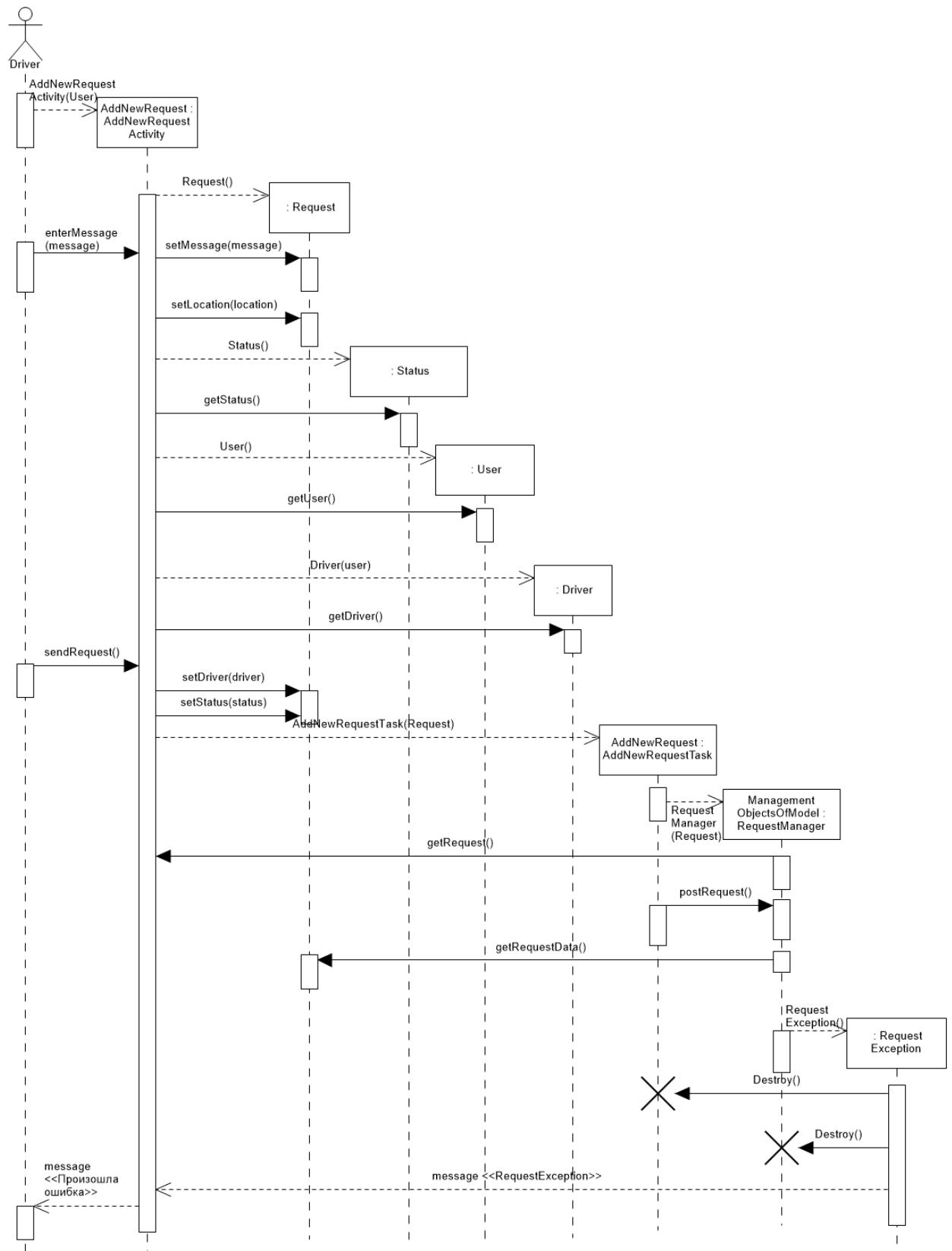


Рис. 5.5. Диаграмма последовательностей действий сценария «Добавление новой заявки». Возникновение исключительной ситуации

### 5.2.3 Построение уточненной диаграммы классов

На данном этапе уточняют отношения между классами (рис. 5.6).

Поскольку объект класса «Driver» может существовать независимо от объектов классов «Request» и «Offer», то для связи его с указанными классами используется отношение агрегации. Класс «Status» является частью класса «Request» - для связи указанных классов также используется отношение агрегации. Для связи класса «RequestList», являющегося списком объектов «Request», также используется отношение агрегации (подобным образом связаны классы «OfferList» и «Offer»).

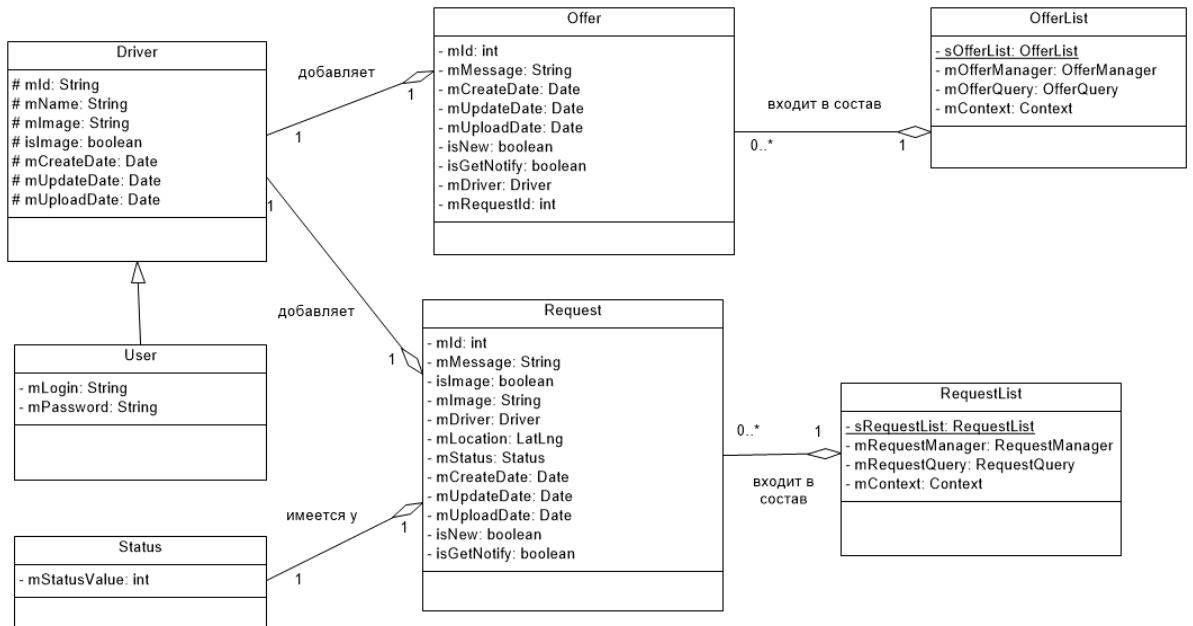


Рис. 5.6. Уточненная диаграмма классов пакета «Объекты модели»

### 5.2.4 Детальное проектирование классов

На этом этапе детализированы атрибуты и методы проектируемых классов.

Детальное описание атрибутов и методов представлено в п. 1.1 прил. 4 «Спецификации на модули» для модуля, реализующего проектируемый пакет «Объекты модели», детальная диаграмма классов которого представлена на рис. 5.7.

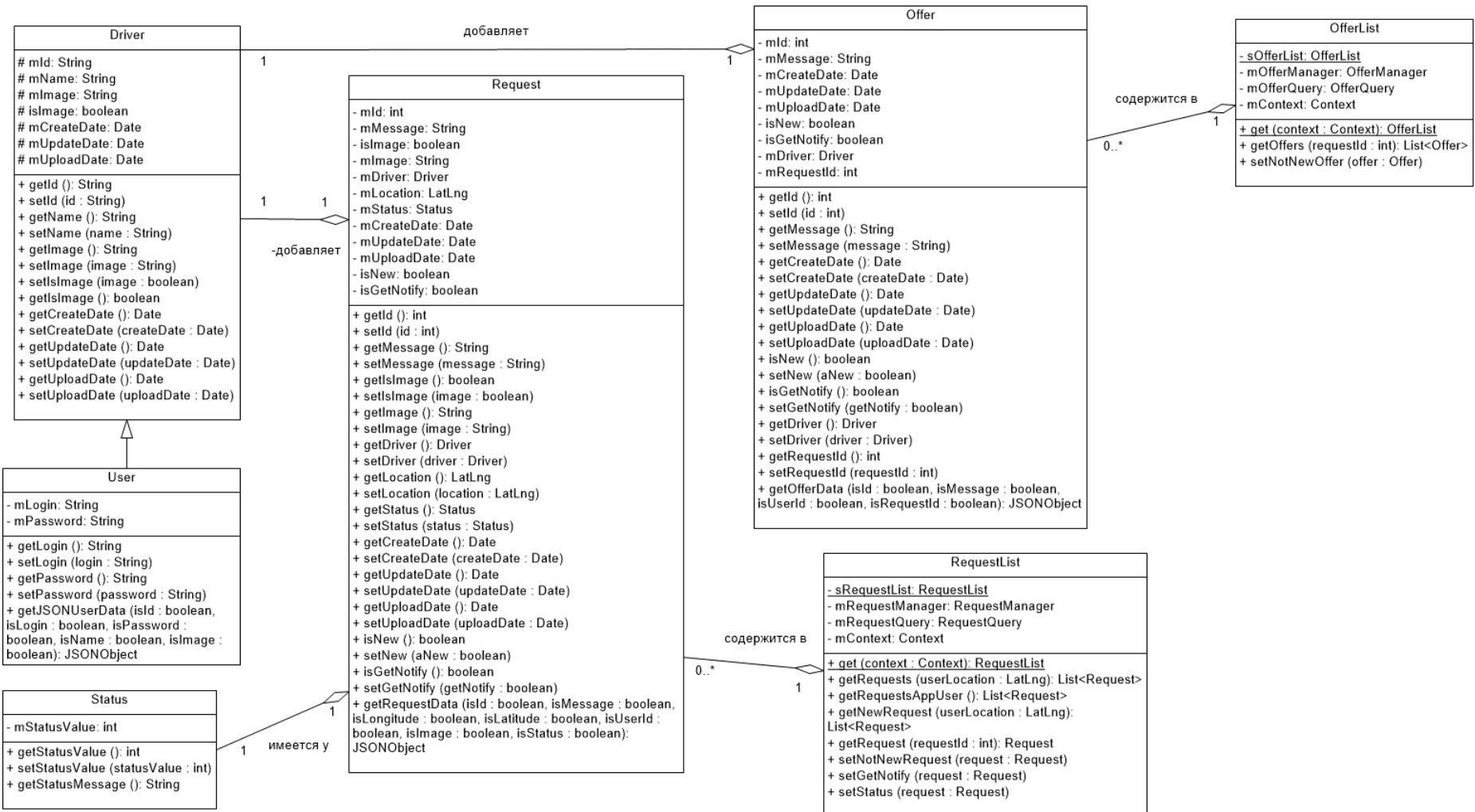


Рис.5.7. Детальная диаграмма классов пакета «Объекты модели»

### 5.3 Проектирование классов пакета «Управление местоположением»

#### 5.3.1 Построение исходной диаграммы классов

Данный пакет предназначен для получения сведений о местоположении пользователя, определенного мобильным устройством на платформе Android.

Список классов-кандидатов, полученный на основе анализа теоретических вопросов определения местоположения [6], представлен в табл. 5.5.

Таблица 5.5

#### Классы пакета «Управление местоположением»

Класс	Описание
LocationListener GPSServices (Слушатель GPS)	Основной класс пакета. Объект данного класса будет запускать Google API-клиент для определения местоположения и контролировать изменение местоположения устройства пользователя
LocationPermission (Разрешение для определения местоположения)	Объект данного класса описывает разрешение, которое требуется получить приложению у системы Android, чтобы определять местоположение
GoogleApiClient. ConnectionCallbacks	Стандартный класс-интерфейс системы Android, который необходимо реализовать для обновления данных местоположения
LocatingListener	Объект данного класса будет извещать объекты пакета «Управление программой» об изменении местоположения
LocationManagerException	Исключение при определении местоположения

Указанные классы имеют следующие зависимости (табл. 5.6).

Таблица 5.6

#### Зависимости между классами пакета «Управление местоположением»

Класс	Описание зависимостей
LocationListenerGPSServices	Наследует интерфейс «GoogleApiClient.ConnectionCallbacks». Связан ассоциативной связью «использует» с классами «LocationPermission», «LocationManagerException»
LocationManagerException	Наследует стандартный класс «Exception»

Исходная диаграмма классов пакета «Управление местоположением» представлена на рис. 5.8.

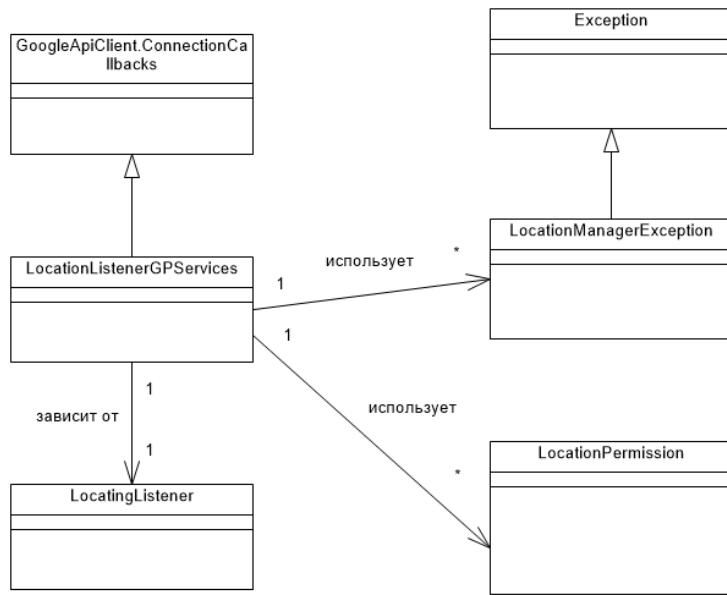


Рис. 5.8. Исходная диаграмма классов пакета «Управление местоположением»

### 5.3.2 Построение диаграмм последовательностей действий

Для сценария «Определения местоположения при добавлении заявки» реализованы диаграммы последовательностей действий, представленные на рис. 5.9 - 5.11.

Указанные диаграммы показывают последовательность обмена сообщениями объектов: «LocationListenerGPServices», «GoogleAPIClient», «LocationPermissions», «GoogleAPIClientConnectionCallback» - пакета «Управление местоположением»; объекта класса «AddNewRequestActivity» пакета «Добавление заявки», который управляет процессом определения местоположения устройством пользователя.

Сценарий заключается в следующем: пользователь создает объект-окно «AddNewRequestActivity», при создании которого создается объект «LocationListenerGPServices», который создает объекты: «GoogleAPIClient», «LocationPermissions» - для проверки разрешений на определение местоположения. «GoogleAPIClient» делает запрос на определение местоположения к системе Android и запускает процесс, который получает обновления местоположения устройства каждые 10 секунд. Процесс

выполняется циклически до тех пор, пока пользователь не закроет окно «AddNewRequestActivity». При возникновении исключительной ситуации вызывается объект класса «LocationManagerException».

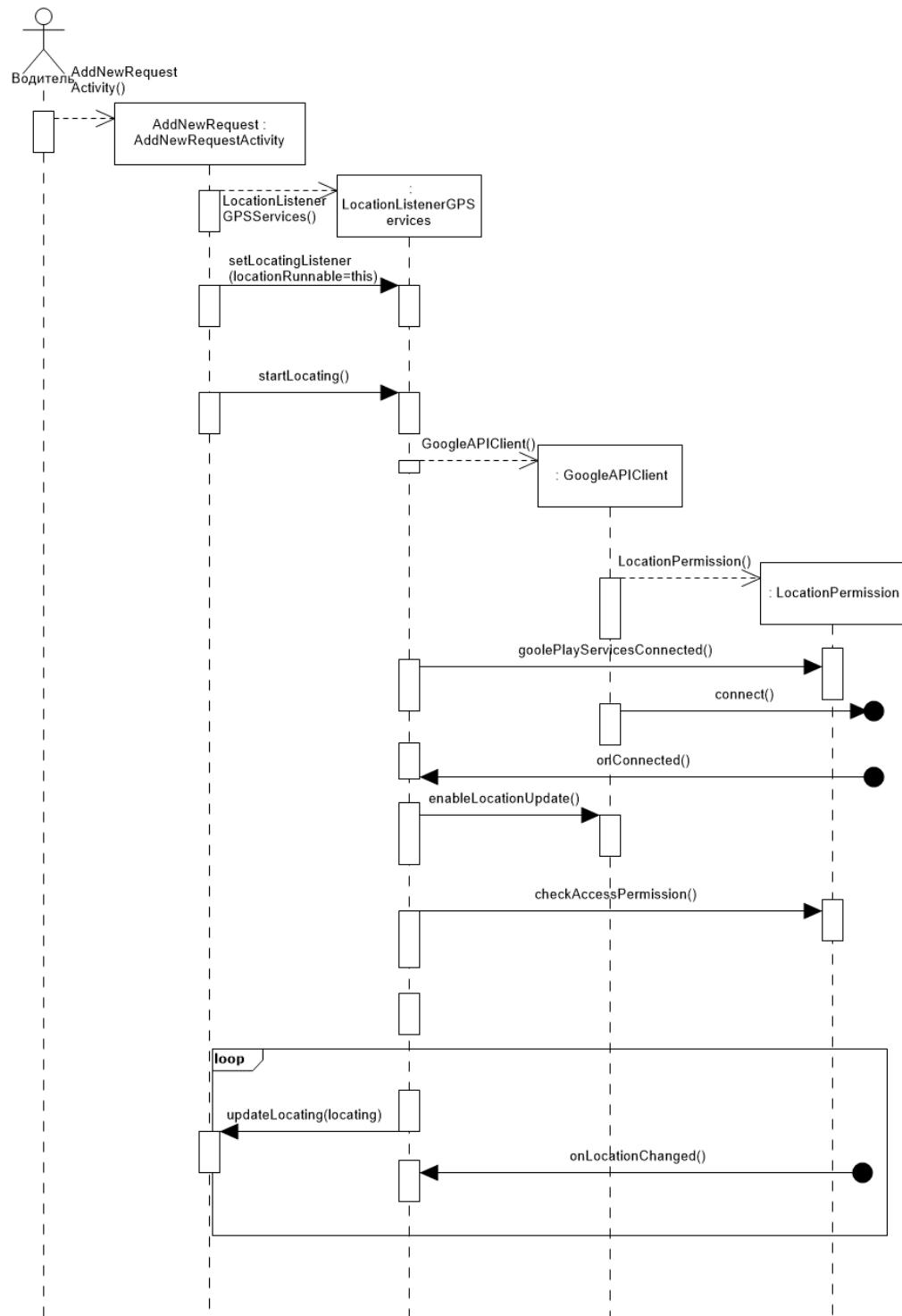


Рис. 5.9. Диаграмма последовательностей действий сценария «Определение местоположения при добавлении заявки». Типичный ход событий

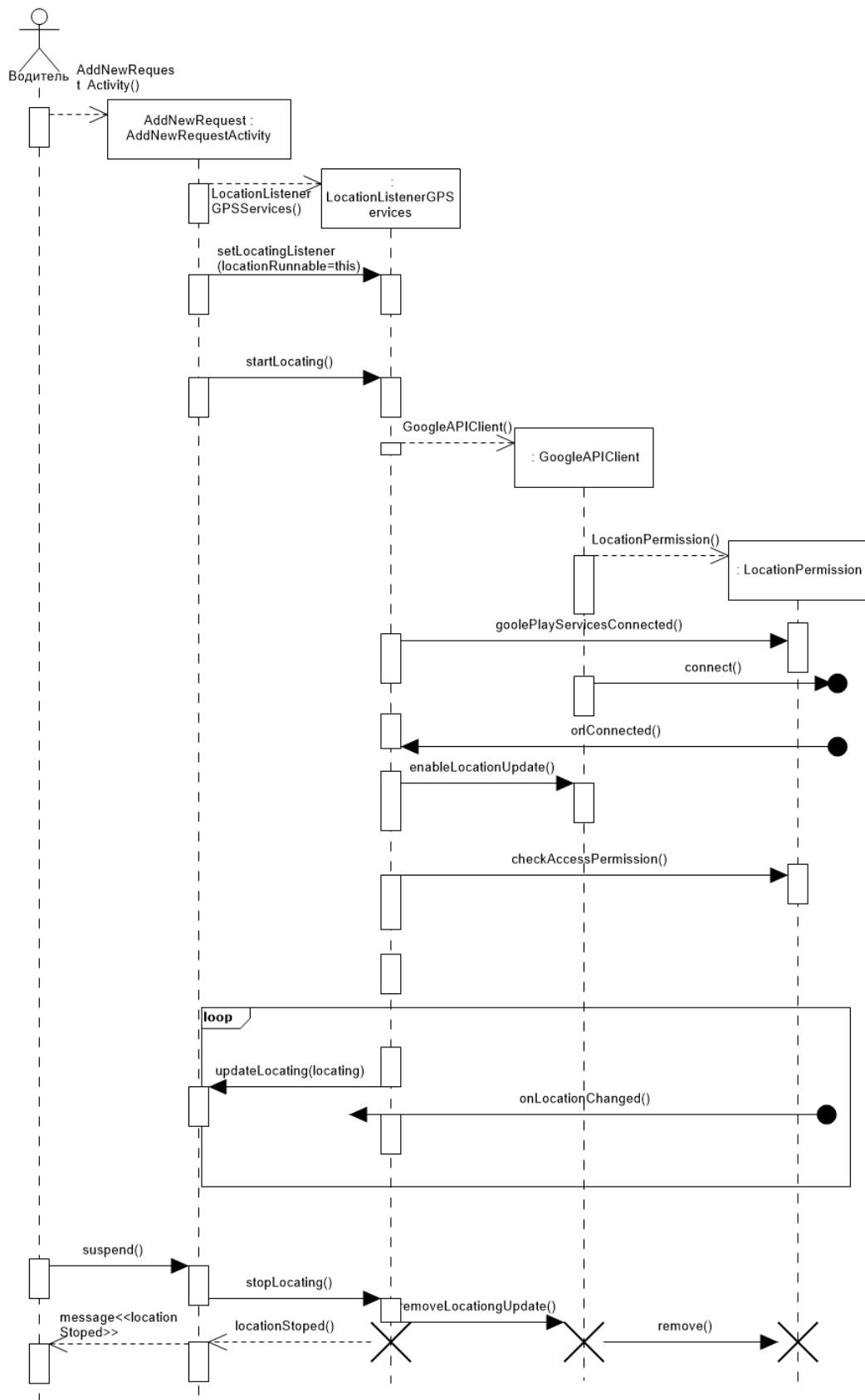


Рис. 5.10. Диаграмма последовательностей действий сценария «Определение местоположения при добавлении заявки». Прерывание пользователя

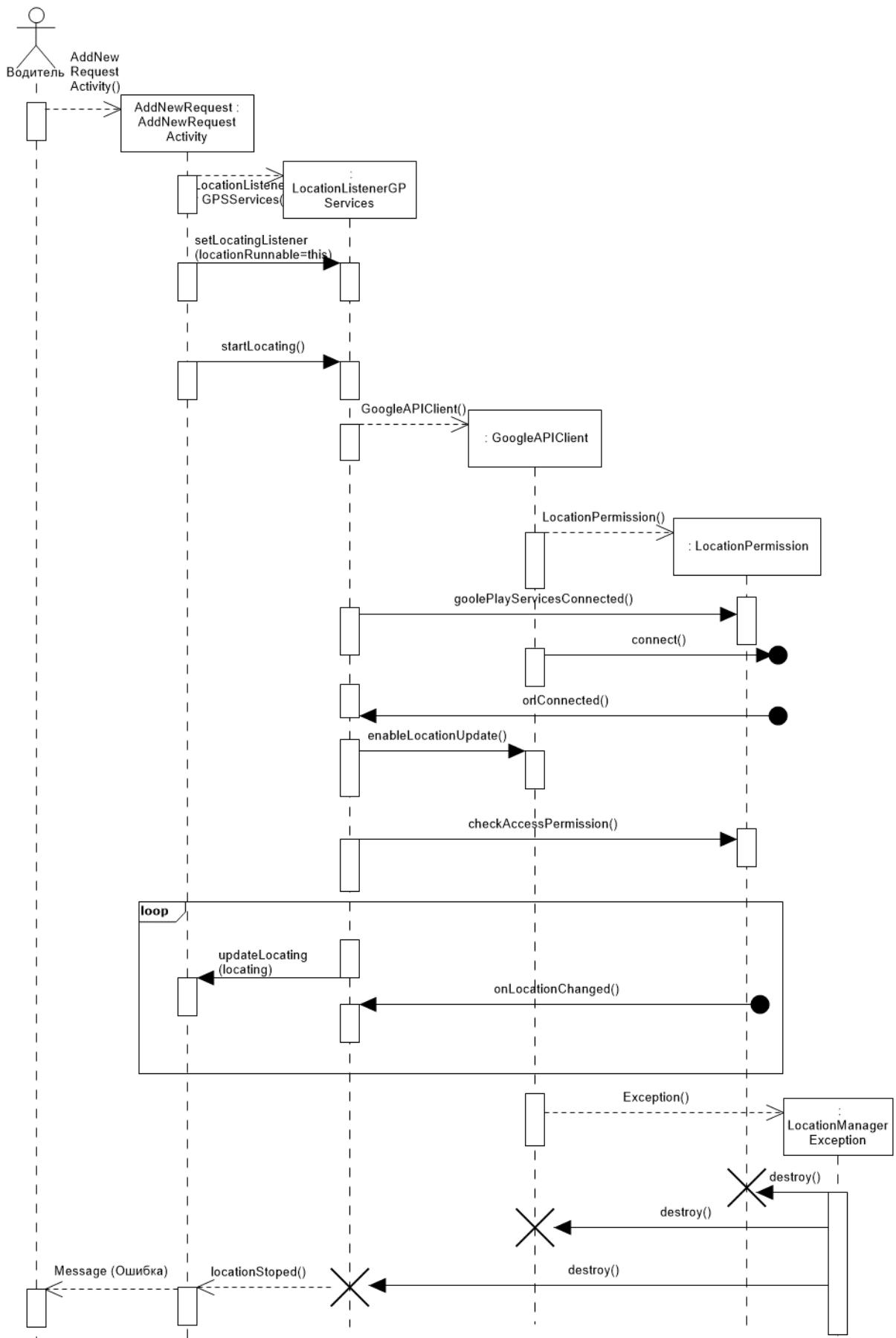


Рис. 5.11. Диаграмма последовательностей действий сценария «Определение местоположения при добавлении заявки». Исключительная ситуация

### 5.3.3 Построение уточненной диаграммы классов

На данном этапе уточнены связи между классами проектируемого пакета.

Класс «*LocationListenerGPSevices*» наследует интерфейс «*GoogleApiClient.ConnectionCallback*».

Класс «*LocatingListener*» является интерфейсным классом, его задача обеспечить интерфейс обратного вызова, чтобы класс, который использует «*LocationListenerGPSevices*», мог получать обновления местоположения устройства пользователя.

Уточненная диаграмма классов пакета «Управление местоположением» представлена на рис. 5.12.

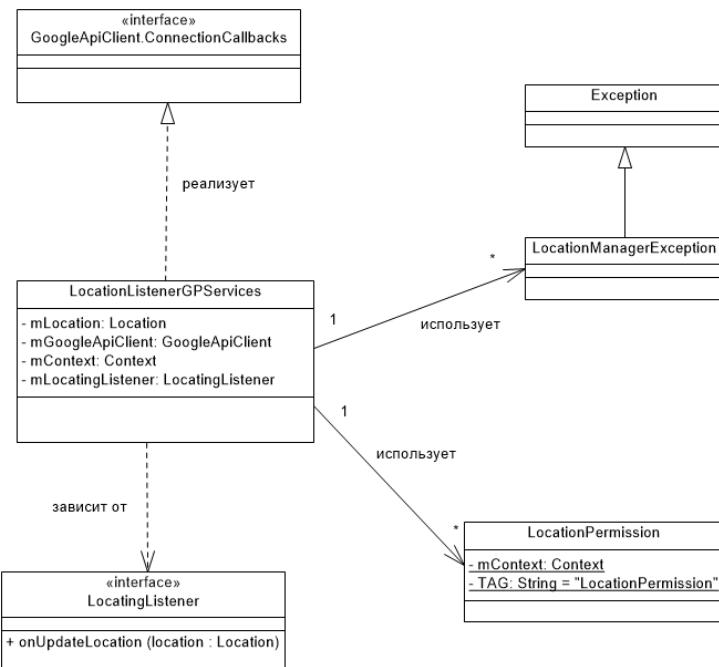


Рис. 5.12. Уточненная диаграмма классов пакета «Управление местоположением»

### 5.3.4 Детальное проектирование классов

В рамках детального проектирования уточнены методы и атрибуты каждого из классов проектируемого пакета (рис. 5.13).

Детальное описание атрибутов и методов представлено в п. 1.2 прил. 4 «Спецификации на модули» для модуля, реализующего проектируемый пакет «Управление местоположением».

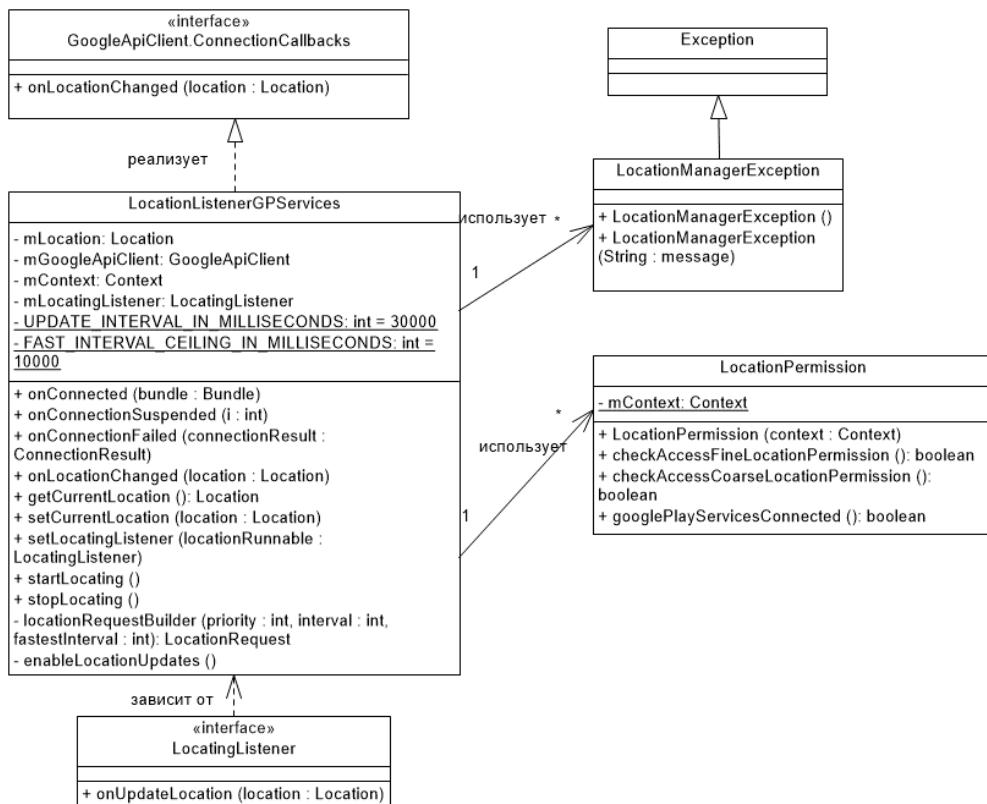


Рис. 5.13. Детальная диаграмма классов пакета «Управление местоположением»

## 5.4 Проектирование классов пакета «Интерфейс к базе данных SQLite»

### 5.4.1 Построение исходной диаграммы классов

Данный пакет предназначен для работы с базой данных SQLite клиентской части приложения: запись данных о заявках на помощь и предложений помощи, полученных от серверной части приложения и их выборку из базы данных. Список классов-кандидатов пакета «Интерфейс к базе данных SQLite» представлен в табл. 5.7.

Указанные классы имеют зависимости, представленные в табл. 5.8.

Исходная диаграмма классов данного пакета представлена на рис. 5.14.

Таблица 5.7

## Классы пакета «Интерфейс к базе данных SQLite»

Класс	Описание
HelpOnRoadBase (База данных «Help on Road»)	Предназначен для создания базы данных SQLite при первом запуске устройства
RequestQuery (Запросы к таблице заявок на помощь)	Предназначен для выполнения SQL-запросов к таблице «Request»
OfferQuery (Запросы к таблице предложений помощи)	Предназначен для выполнения SQL-запросов к таблице «Offers»
UserQuery (Запросы к таблице пользователей)	Предназначен для выполнения SQL-запросов к таблице «Users»
SQLiteDataBaseSchema (Схема БД)	Описывает структуру базы данных SQLite мобильного приложения (содержит названия полей таблиц базы данных)
OfferCursorWrapper («Курсор для таблицы Offers»)	Предназначен для преобразования полученных данных из базы о предложениях помощи в объект класса «Offers»
RequestCursorWrapper («Курсор для таблицы Request»)	Предназначен для преобразования полученных данных из базы о заявках в объект класса «Request»

Таблица 5.8

## Зависимости между классами пакета «Интерфейс к базе данных SQLite»

Класс	Описание зависимостей
HelpOnRoadBase	Создает подключение к базе данных объектам классов: «RequestQuery», «OfferQuery», «UserQuery»
RequestQuery, UserQuery, OfferQuery	Связаны ассоциативной связью «использует» с классами: «RequestTable», «OfferTable», «UserTable» - для доступа к названиям полей базы данных SQLite

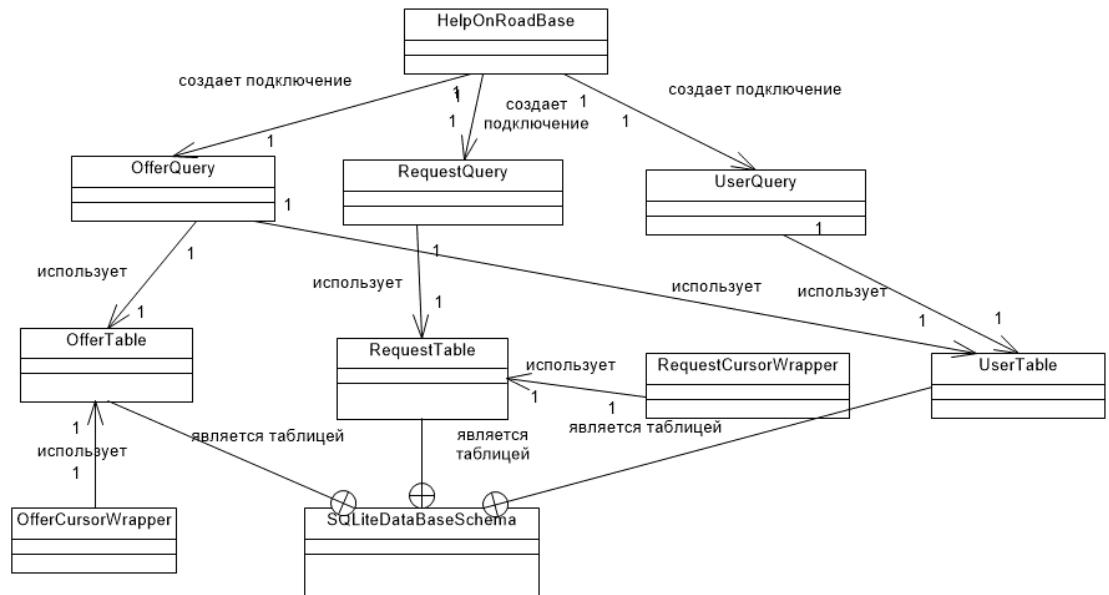


Рис. 5.14. Исходная диаграмма классов пакета «Интерфейс к базе данных SQLite»

### 5.4.2 Построение диаграмм последовательностей действий

Для реализации сценария «Просмотр заявок» (получение сведений о заявках из базы SQLite и отображение полученных данных на экране устройства) построены диаграммы последовательностей действий, которые представлены на рис. 5.15 - 5.17.

Указанные диаграммы показывают последовательность обмена сообщениями объектов классов: «RequestQuery», «RequestCursorWrapper», «RequestTable», «UserTable», «HelpOnRoadBase» - пакета «Интерфейс к базе данных SQLite»; «Request» («Заявка»), «RequestList» («Список заявок») - пакета «Объекты модели»; объекта «RequestListFragment» пакета «Управление списком заявок» («Management Request List»).

Сценарий заключается в следующем: пользователь запускает фрагмент окна пользовательского интерфейса «RequestListFragment», который создает объект «RequestList». Объект «RequestList» создает объект «RequestQuery», который выполняет запрос к базе данных SQLite для получения списка актуальных заявок на помощь.

Процессом извлечения данных из базы управляет объект «RequestQuery». Он обращается к объектам: «RequestTable», «UserTable» - для получения названий полей таблиц: «Request», «User» - базы данных приложения мобильного устройства. Затем объект «RequestQuery» выполняет SQL-запрос к базе данных, извлекает список заявок в виде стандартного курсора и посредством объекта «RequestQursorWrapper» выполняет заполнение списка объектов «Request», полученными данными из базы.

После окончания процесса заполнения списка объект «RequestList» закрывает подключение к базе данных и передает список объектов «Request» объекту «RequestListFragment», который выводит полученный список заявок на экран пользователю приложения.

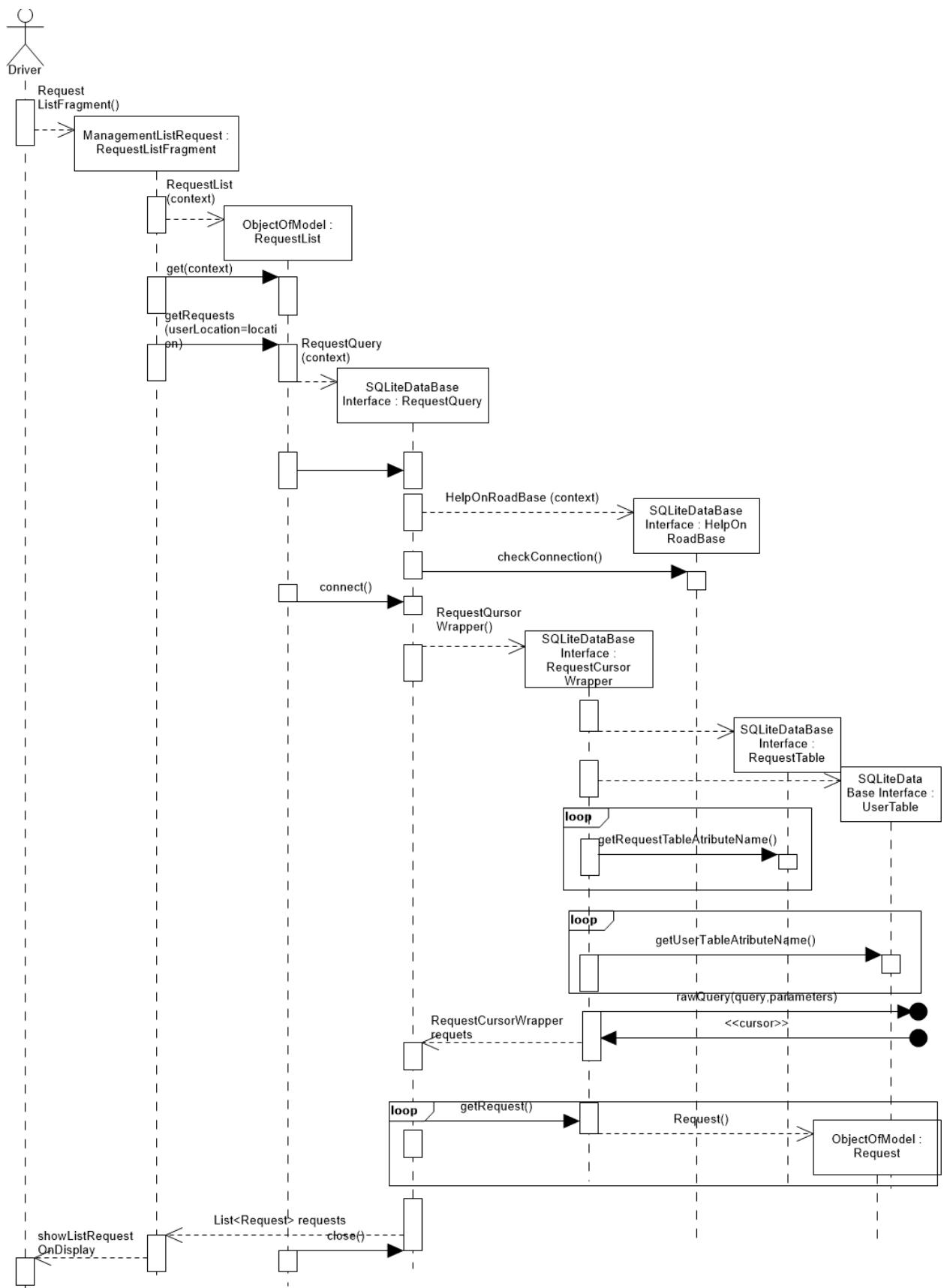


Рис. 5.15. Диаграмма последовательностей действий сценария «Просмотр заявок». Типичный ход событий

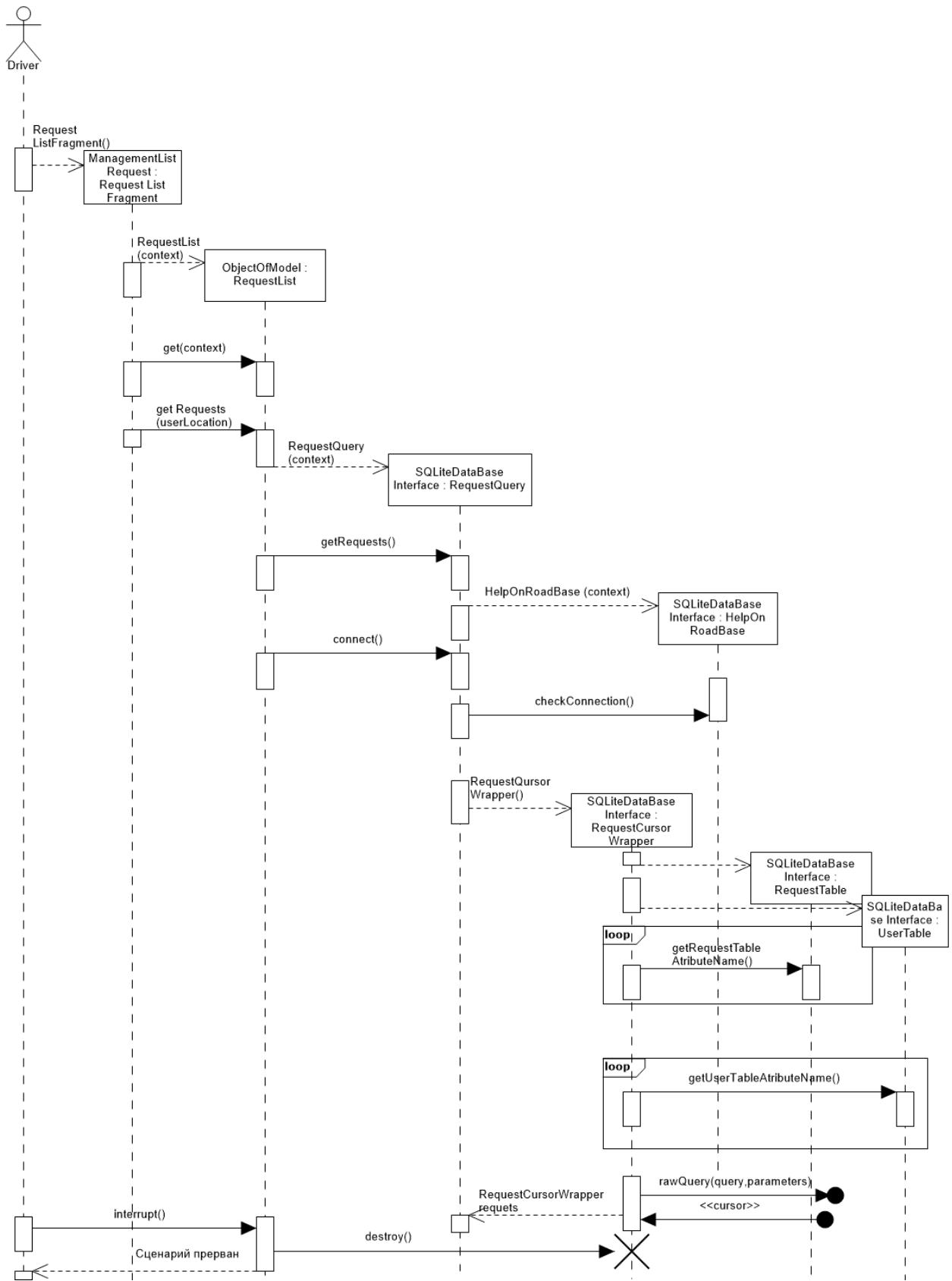


Рис. 5.16. Диаграмма последовательностей действий сценария «Просмотр заявок». Прерывание пользователя

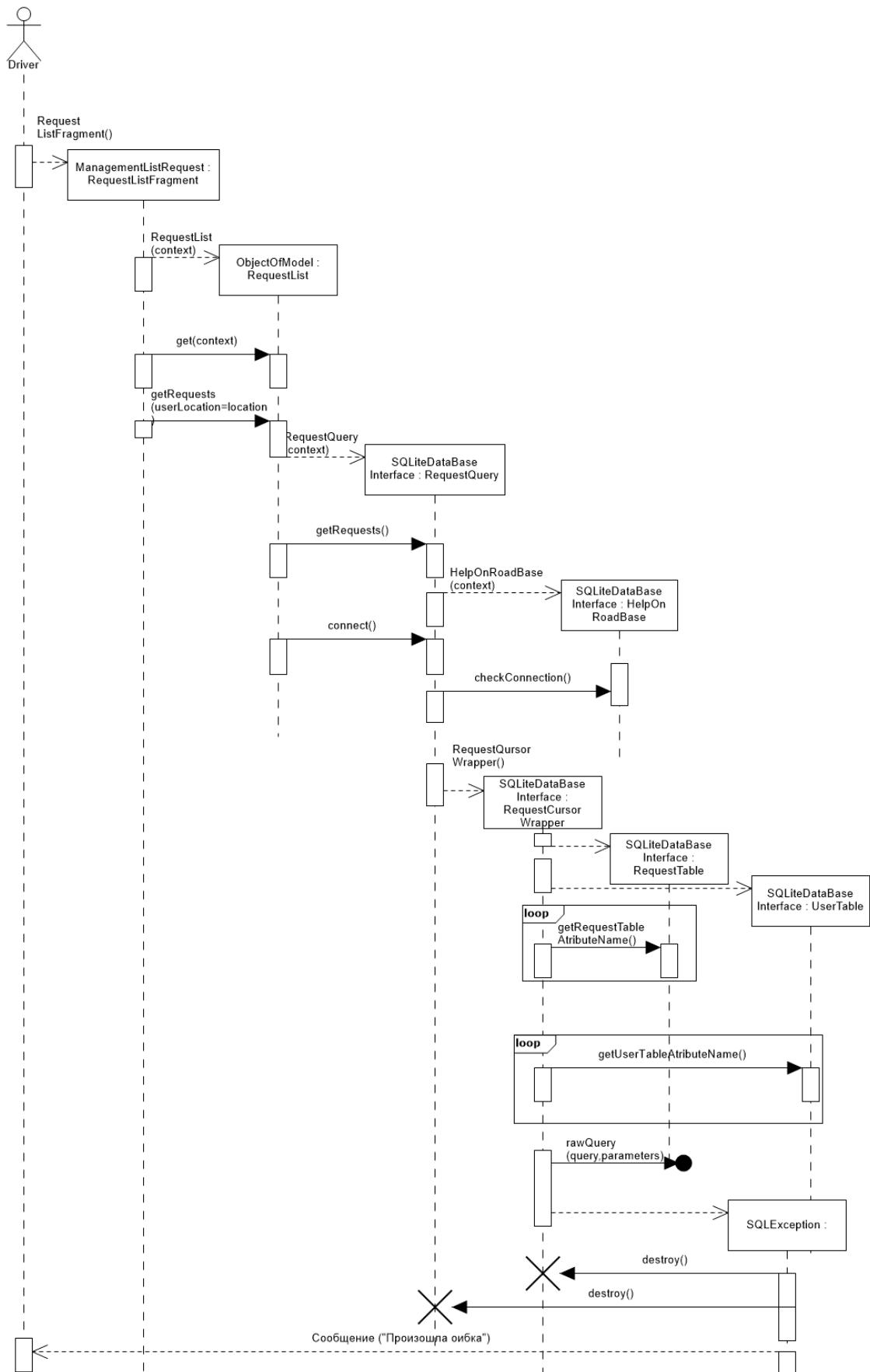


Рис. 5.17. Диаграмма последовательностей действий сценария «Просмотр заявок». Возникновение исключительной ситуации

### 5.4.3 Построение уточненной диаграммы классов

На уточненной диаграмме классов показаны атрибуты класса «HelpOnRoadBase»: версия и имя базы данных.

Классы: «OfferQuery», «RequestQuery», «UserQuery» - имеют системные атрибуты: mContext, mDatabase - для доступа к базе данных.

Классы: «OfferTable», «RequestTable», «UserTable» - имеют набор константных строк, которые являются названиями полей соответствующих таблиц в базе данных.

Также уточнены связи между классами. Классы: «OfferQuery», «RequestQuery», «UserQuery» - содержат ссылку на объект класса «HelpOnRoadBase» для выполнения подключения к базе данных приложения.

Уточненная диаграмма классов данного пакета представлена на рис. 5.18.

### 5.4.4 Детальное проектирование классов

На данном этапе проектирования уточнены методы каждого из классов.

Класс «HelpOnRoadBase» содержит методы: onCreate, onUpdate - для создания и обновления схемы базы данных, соответственно.

Классы: «OfferQuery», «RequestQuery», «UserQuery» - содержат методы: openConnection, closeConnection – для открытия и закрытия подключения к базе данных; методы: addOffer, addRequest, addUser – для вставки записи в соответствующую таблицу базы данных; методы: updateOffer, updateRequest, updateUser – для обновления записи в соответствующей таблице базы данных и методы: getUserValue, getRequestValue, getOfferValue – для преобразования соответствующего объекта в объект ContentValues, который используется в SQL-запросах.

Детальное описание атрибутов и методов представлено в п. 1.3. прил. 4 «Спецификации на модули» для модуля, реализующего проектируемый пакет «Интерфейс к БД SQLite», детальная диаграмма которого представлена на рис. 5.19.

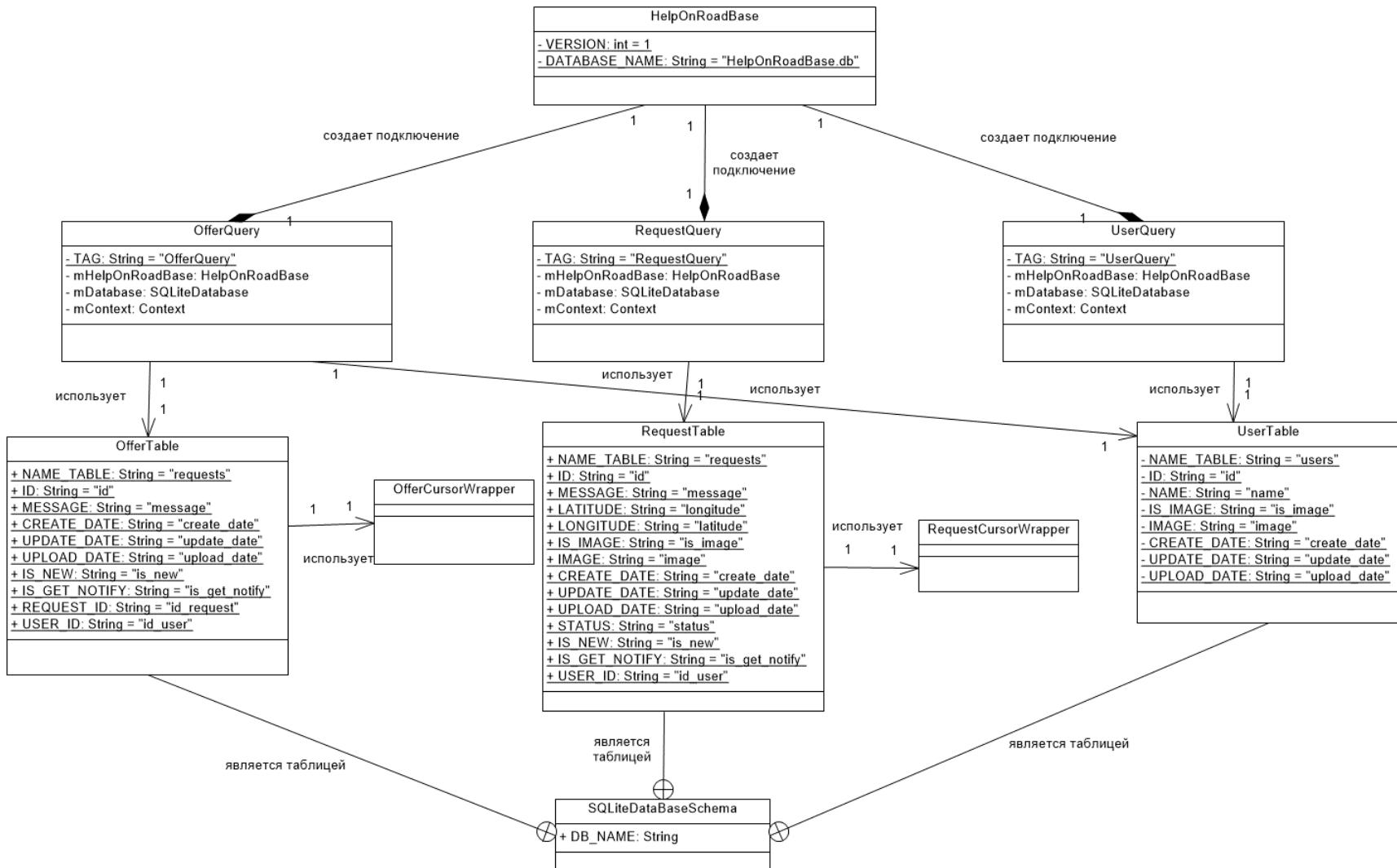


Рис. 5.18. Уточненная диаграмма классов пакета «Интерфейс к базе данных SQLite»

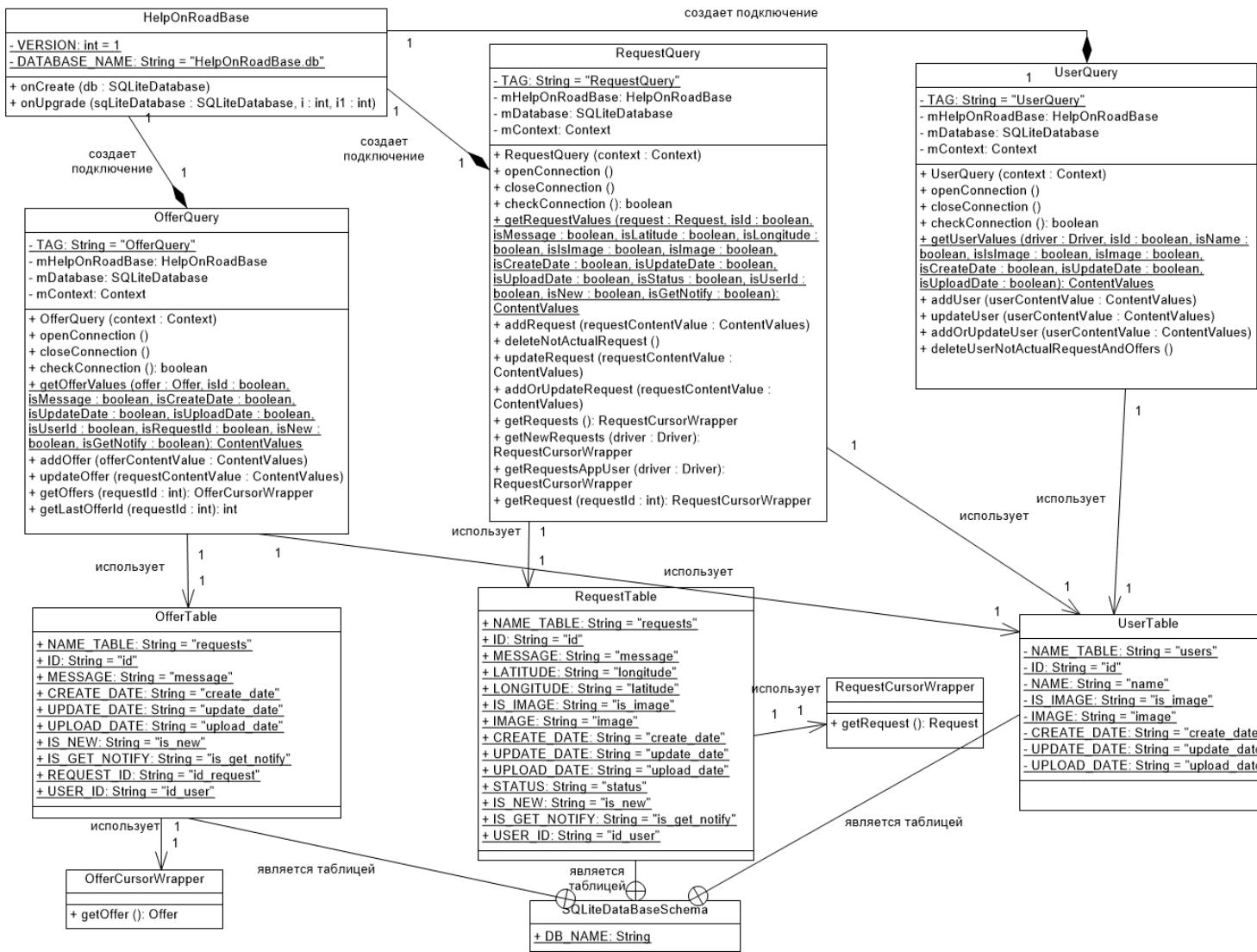


Рис. 5.19. Детальная диаграмма классов пакета «Интерфейс к базе данных SQLite»

## 5.5 Проектирование классов пакета «JSON-парсеры»

### 5.5.1 Построение исходной диаграммы классов

Данный пакет предназначен для преобразования данных: о заявках на помощь, о пользователях приложения, о предложениях помощи, о маршруте между пользователем и заявкой на помощь, полученном в формате JSON от серверной части приложения по протоколу HTTP, в объекты классов пакета «Объекты модели». Список классов-кандидатов пакета представлен в табл. 5.9, связи между классами - в табл. 5.10.

Таблица 5.9

#### Классы пакета «JSON-парсеры»

Класс	Описание
ReplayParser (Разбор ответа сервера)	Предназначен разбора ответа сервера об успешности выполнения запроса на получение данных
RequestParser (Разбор данных заявки)	Предназначен для преобразования полученных данных заявки в объект класса «Request»
OfferParser (Разбор данных предложения помощи)	Предназначен для преобразования полученных данных предложения помощи в объект класса «Offer»
UserParser (Разбор данных пользователя)	Предназначен для преобразования полученных данных пользователя в объект класса «Driver»
DateParser (Разбор даты)	Предназначен для преобразования строкового представления даты в объект класса «Date» и наоборот
RouteParser (Разбор данных маршрута)	Предназначен для разбора данных о маршруте, полученного от сервера Google

Таблица 5.10

#### Зависимости между классами пакета «JSON-парсеры»

Класс	Описание зависимостей
ReplayParser	Является базовым классом для классов: «RequestParser», «OfferParser», «UserParser»
DateParser	Используется классами: «RequestParser», «OfferParser», «UserParser», «RouteParser» - для работы со строковыми представлениями даты

Исходная диаграмма классов пакета «JSON-парсеры» представлена на рис. 5.20.

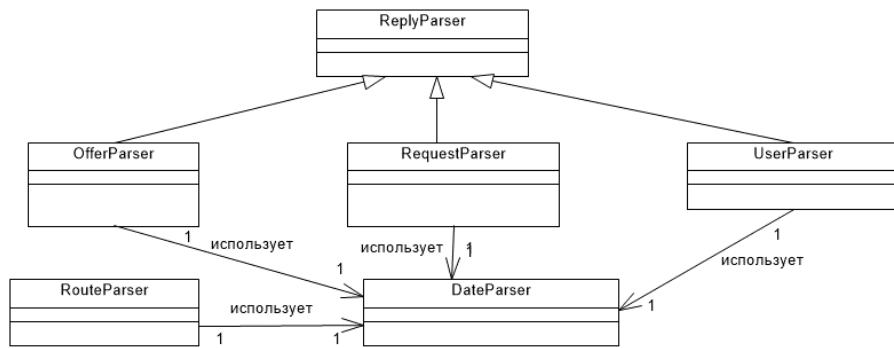


Рис. 5.20. Исходная диаграмма классов пакета «JSON-парсеры»

### 5.5.2 Построение диаграмм последовательностей действий

Для получения данных о заявках от серверной части приложения в рамках сценария «Просмотр заявок» построены диаграммы последовательностей действий, которые показывают обмен сообщениями объектов классов: «RequestParser», «UserParser», «DateParser» - пакета «JSON-парсеры» («JSONParsers»); объекта «Request» пакета «Объекты модели», объекта «RequestManager» («Управление заявками») пакета «Управление объектами модели» («ManagementObjectsOfModel»), объекта «RequestListFragment» («Окно для просмотра заявок») пакета «Управление списком заявок» («Management Request List»), «HttpsQuery» («HTTPS-запрос») пакета «Сетевые операции».

Сценарий следующий: пользователь запускает окно «RequestListFragment», которое создает объект «RequestManager», который, используя объект «HttpsQuery», выполняет HTTP-запрос серверной части приложения для получения списка заявок на помощь в формате JSON. Получив требуемые данные, объект «RequestManager» создает объекты: «RequestParser», «DateParser», которые выполняют разбор полученных данных заявок в список объектов «Request». Разобранные данные «RequestManager» сохраняет в базе данных SQLite и передает управление объекту «RequestListFargment», который отображает полученные данные на экране устройства. Диаграмма последовательностей действий рассматриваемых объектов представлена на

рис. 5.21 - 5.23 (для типичного хода событий, прерывания сценария пользователем, исключительной ситуации).

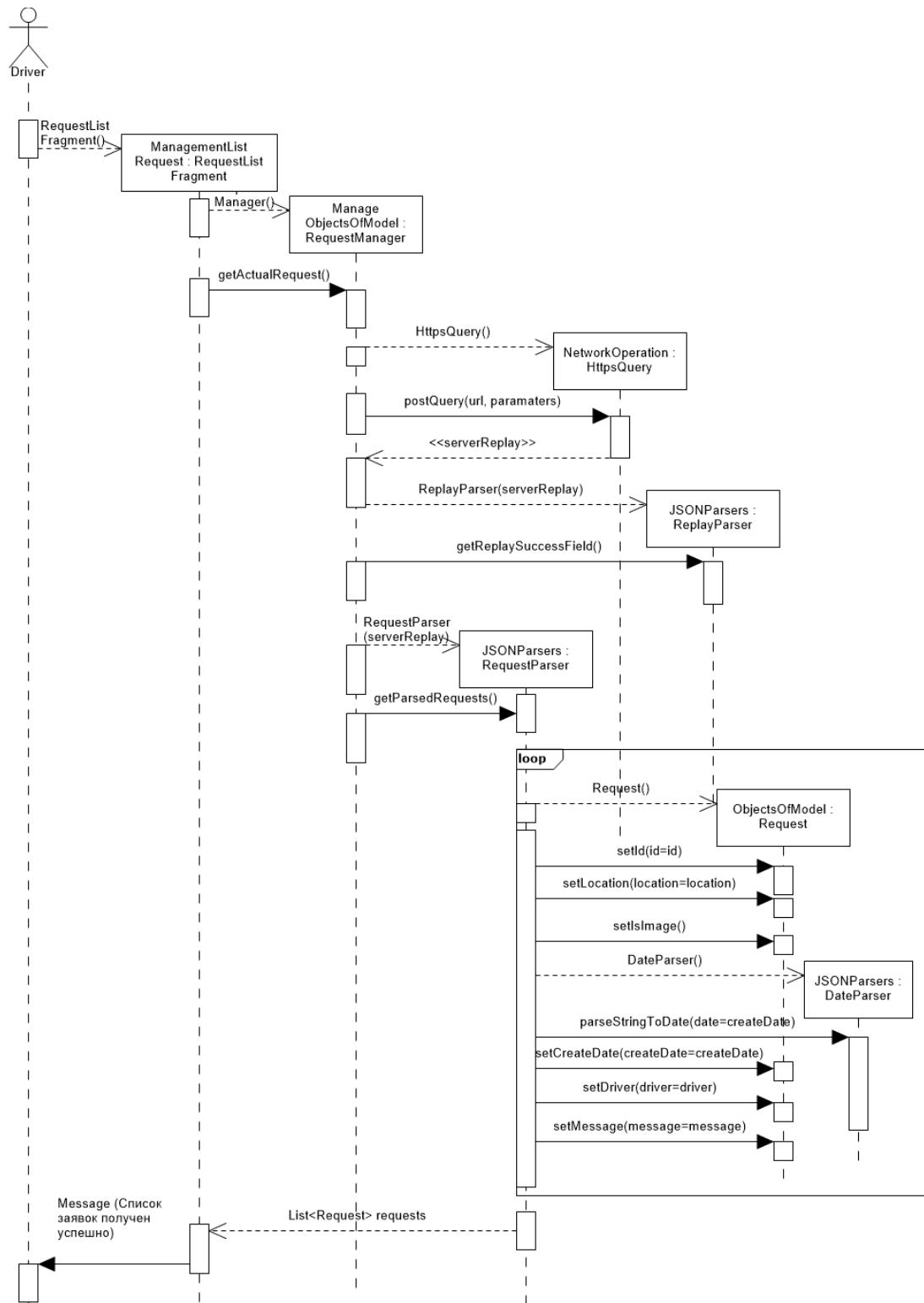


Рис. 5.21. Диаграмма последовательностей действий сценария «Получение списка заявок». Типичных ход событий

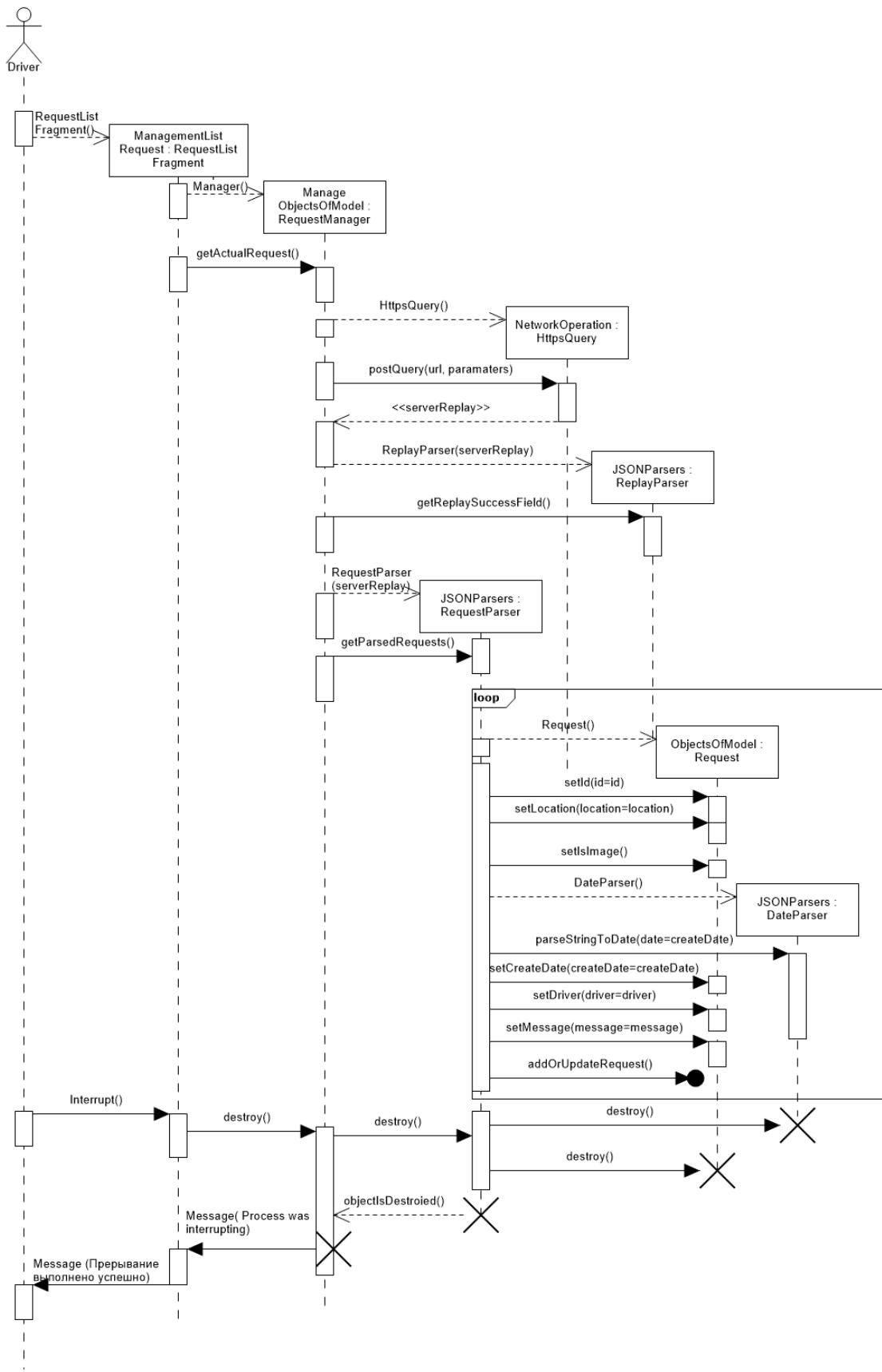


Рис. 5.22. Диаграмма последовательностей действий сценария «Получение списка заявок». Прерывание пользователя

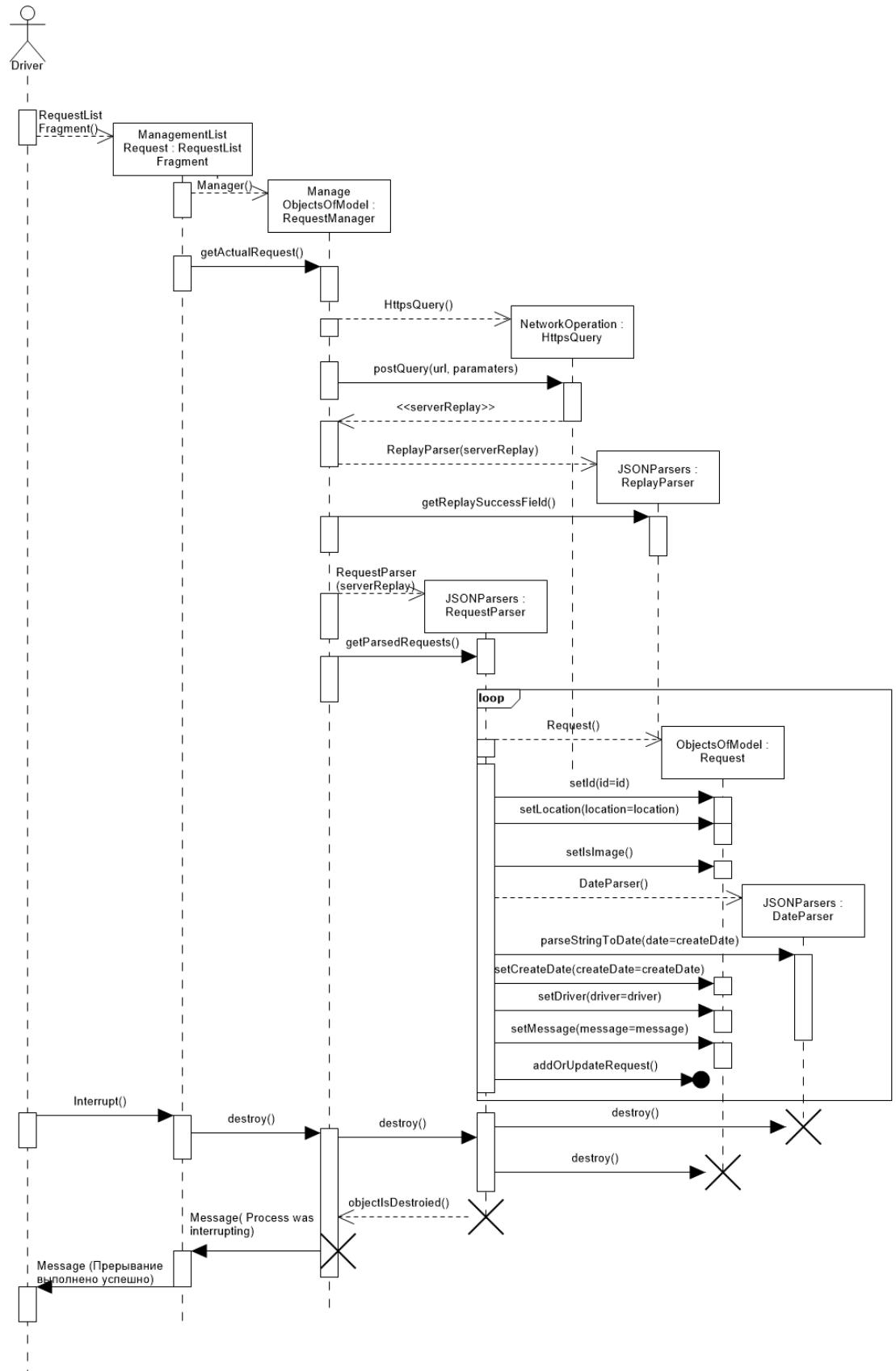


Рис. 5.23. Диаграмма последовательностей действий сценария «Получение списка заявок». Возникновение исключительной ситуации

### 5.5.3 Построение уточненной диаграммы классов

На данном этапе уточнены атрибуты проектируемых классов.

Класс «*ReplayParser*» содержит поля: *mByteReplay* – ответ, полученный от серверной части, представленный массивом байтов, *mStringReplay* – ответ, полученный от серверной части в строковом представлении.

Класс «*RouteParser*» содержит поле *mPoints*, которое представляет собой строковое представление, описывающее точки линии маршрута.

Уточненная диаграмма классов данного пакета представлена на рис. 5.24.

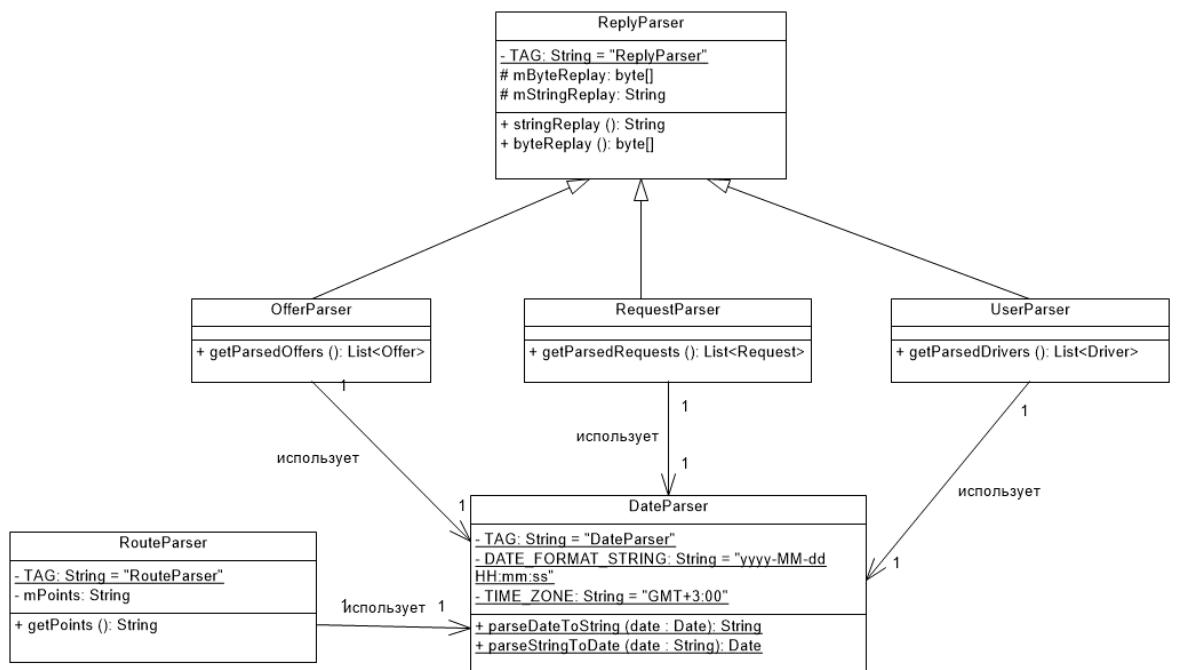


Рис. 5.24. Уточненная диаграмма классов пакета «JSON-парсеры»

### 5.5.4 Детальное проектирование классов

На данном этапе проектирования выполнено дальнейшее уточнение методов проектируемых классов.

Детальное описание атрибутов и методов представлено в п. 1.4 прил. 4 «Спецификации на модули» для модуля, реализующего проектируемый пакет «JSON-парсеры», детальная диаграмма которого представлена на рис. 5.25.

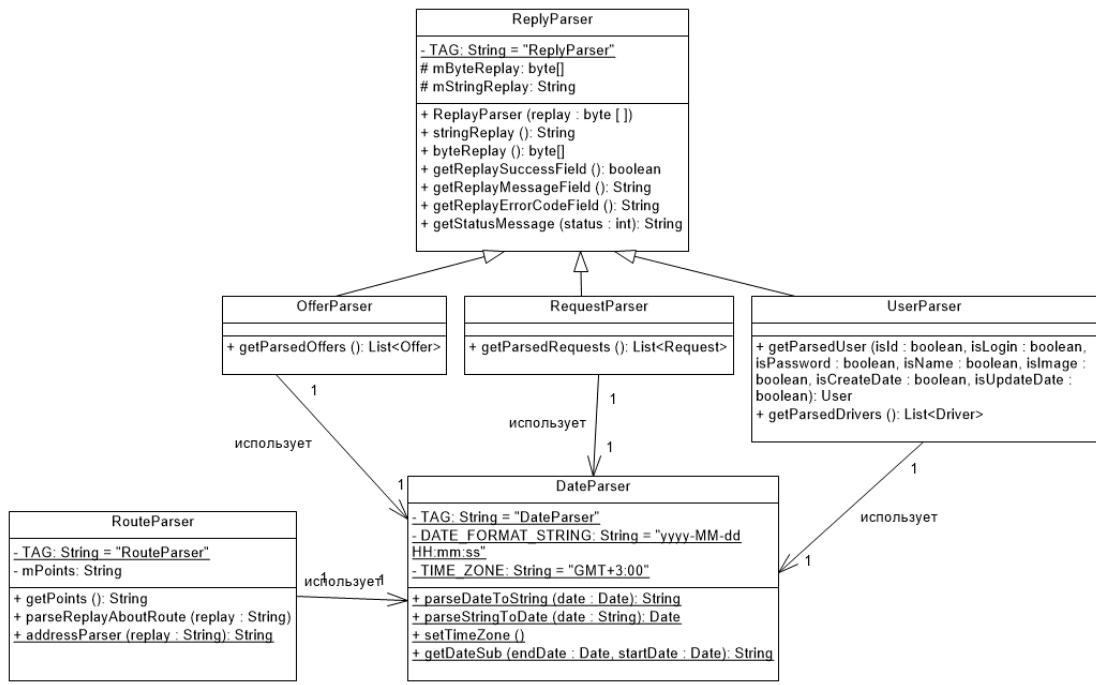


Рис. 5.25. Детальная диаграмма классов пакета «JSON-парсеры»

## 5.6 Проектирование классов пакета «Управление объектами модели»

### 5.6.1 Построение исходной диаграммы классов

Данный пакет предназначен для работы с объектами предметной области: формирование запросов к серверной части приложения на получение и добавление данных: о заявках на помощь, о предложениях помощи, о пользователях. Список классов-кандидатов пакета «Управление объектами модели» представлен в табл. 5.11. Исходная диаграмма классов проектируемого пакета представлена на рис. 5.26.

Таблица 5.11

### Классы пакета «Управление объектами модели»

Класс	Описание
1	2
UserManager (Управление пользователем приложения)	Предназначен для формирования запросов к серверной части приложения на регистрацию, авторизацию пользователя
UserPreferences (Управление настройками пользователя)	Предназначен для формирования запросов к стандартному файлу настроек приложения для получения информации о пользователе
RequestManager (Управление заявками)	Предназначен для формирования запросов к серверной части приложения на добавление, получение списка заявок

1	2
OfferManager (Управление предложениями помощи)	Предназначен для управления списком предложений помощи
FileManager (Управление файлами)	Предназначен для создания, удаления файла в памяти устройства
PhotoManager (Управление фотографиями)	Предназначен для выполнения операций обработки графических файлов
GoogleAPIManager (Управление запросами к API Google)	Предназначен для формирования запросов к серверу Google для получения сведений о маршруте и адресе местоположения

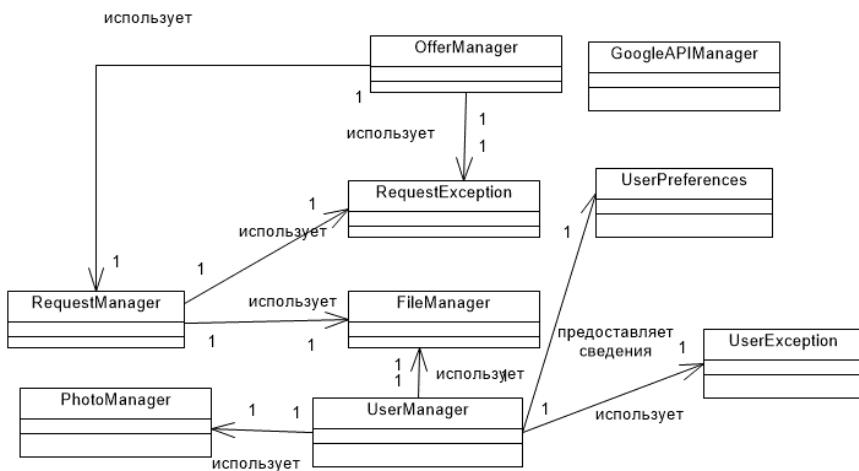


Рис. 5.26. Исходная диаграмма классов пакета «Управление объектами модели»

### 5.6.2 Построение диаграмм последовательностей действий

Для реализации изменения фотографии пользователя в процессе выполнения сценария «Редактировать данные о пользователе» построены диаграммы последовательностей действий (рис. 5.27 - 5.29), которые показывают последовательность обмена сообщениями объектов классов: «UserPreferences», «UserManager», «PhotoManager», «FileManager» пакета «Управление объектами модели»; объекта класса «User» пакета «Объекты модели», объекта «UserAccountFragment» пакета «Управление пользователем».

Сценарий заключается в следующем: пользователь запускает окно «UserAccountFragment», которое создает объект «UserManager». Объект «UserManager» создает объект «User», заполняя его данными из файла настроек приложения, используя объект «UserPreferences».

Когда пользователь выбрал фотографию из галереи, объект «UserAccountFragment», используя объекты: «PhotoManager», «FileManager» - выполняет: масштабирование и сохранение фотографии в памяти устройства, сохранение пути к изображению в файле настроек приложения и отправку фотографии серверной части приложения.

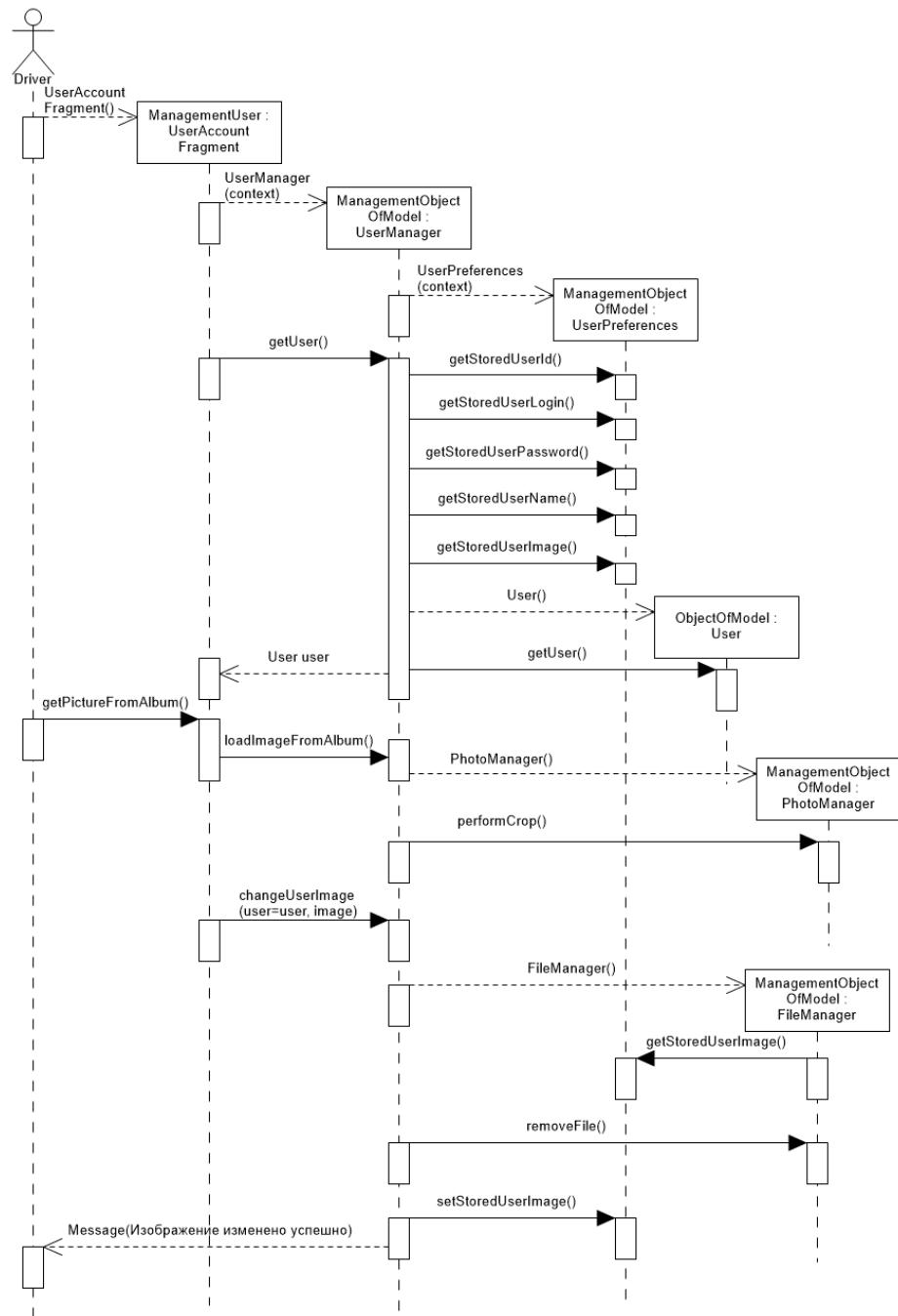


Рис. 5.27. Диаграмма последовательностей действий сценария «Изменение фотографии пользователя». Типичных ход событий

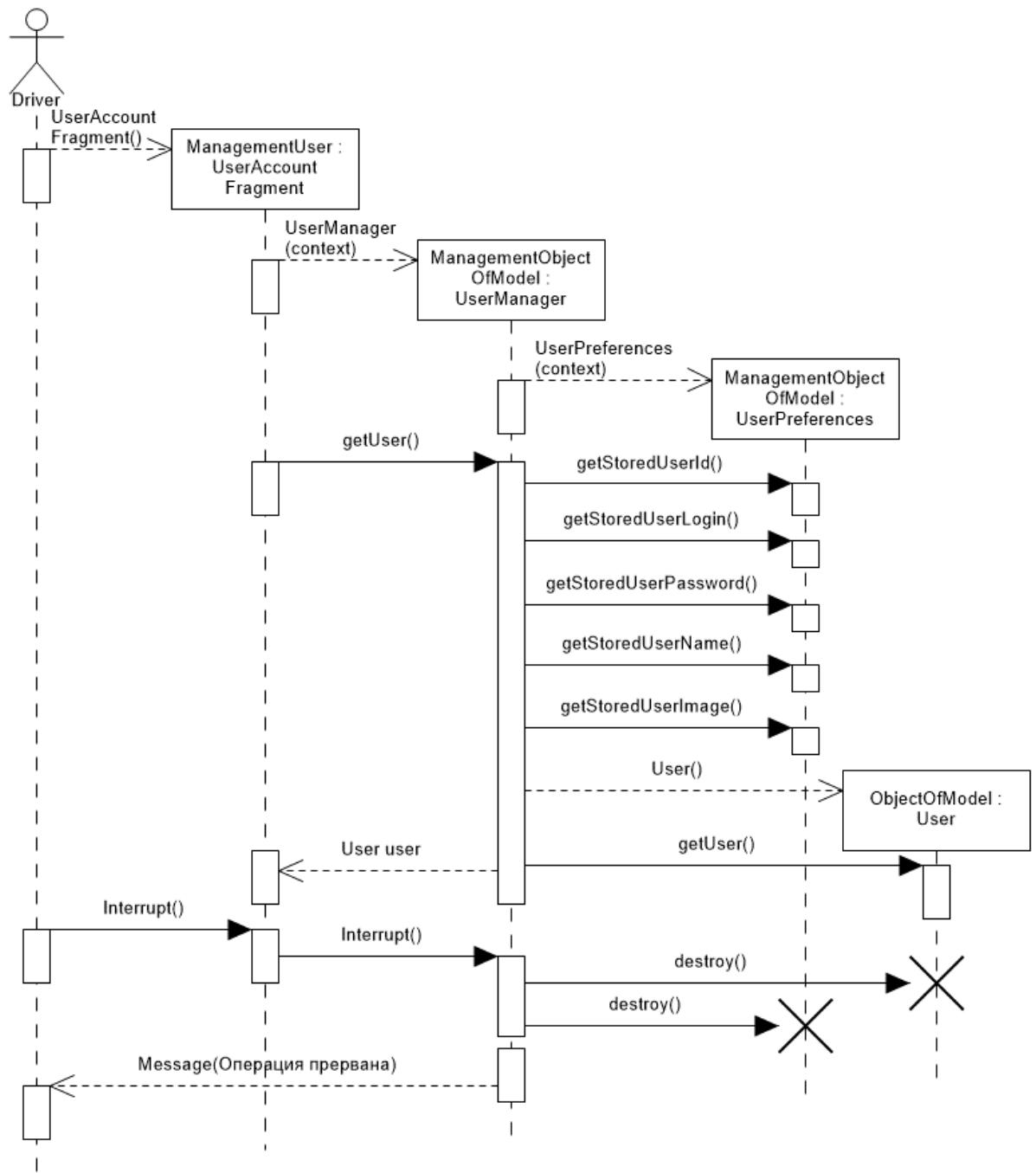


Рис. 5.28. Диаграмма последовательностей действий сценария «Изменение фотографии пользователя». Прерывание пользователя

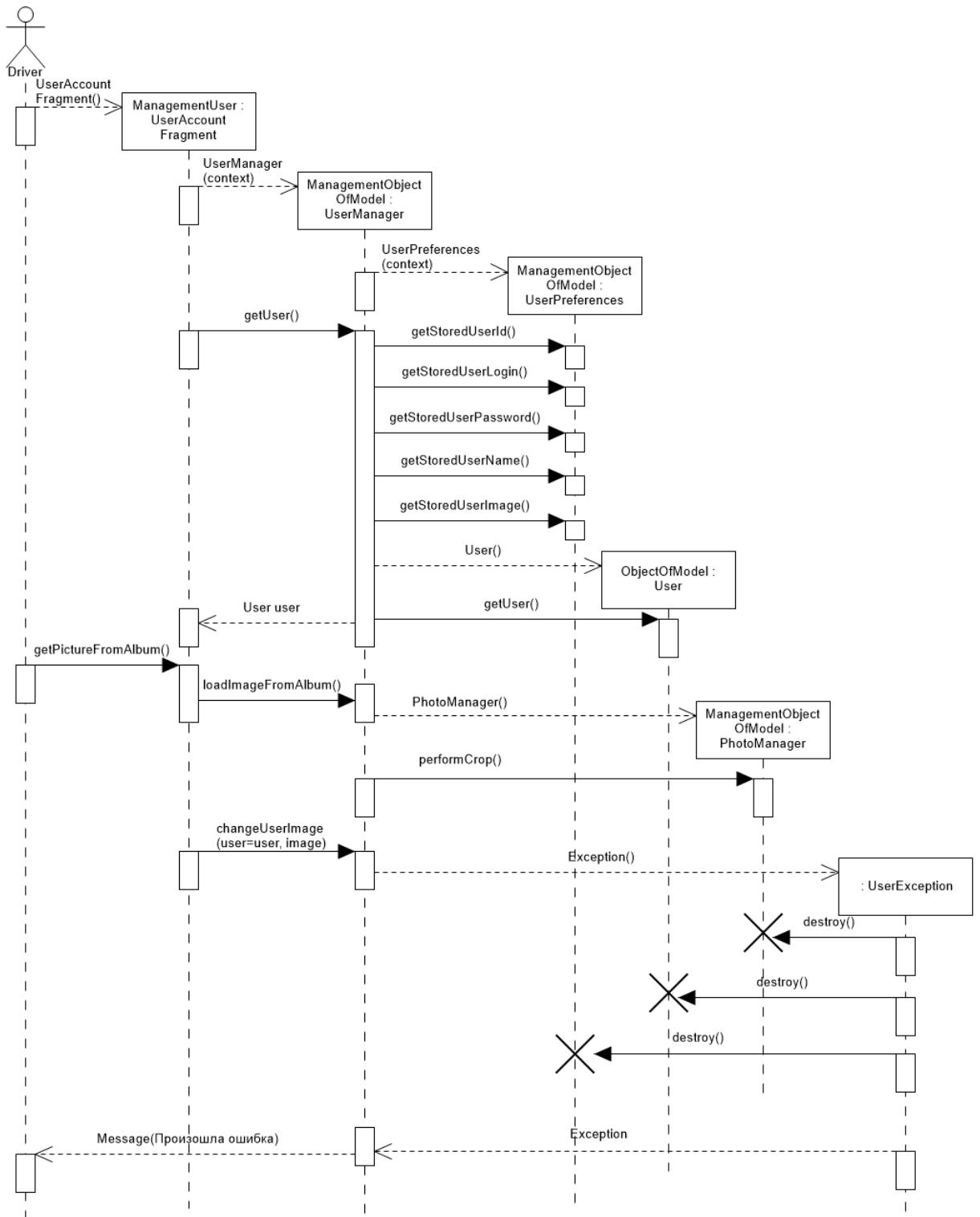


Рис. 5.29. Диаграмма последовательностей действий сценария «Изменение фотографии пользователя». Возникновение исключительной ситуации

### 5.6.3 Построение уточненной диаграммы классов

На уточненной диаграмме классов детализированы атрибуты проектируемых классов.

Уточненная диаграмма классов данного пакета представлена на рис. 5.30.

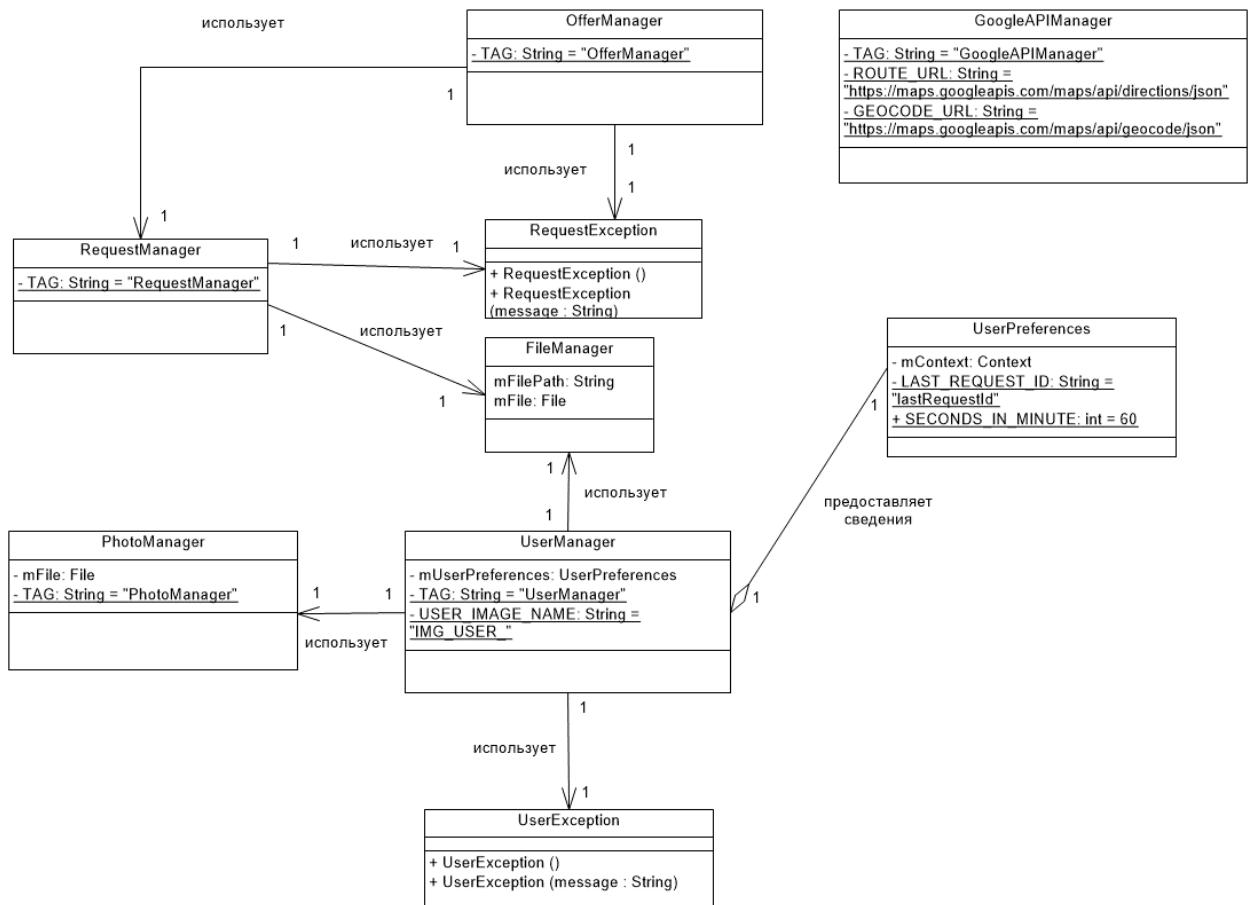


Рис. 5.30. Уточненная диаграмма классов пакета «Управление объектами модели»

### 5.6.4 Детальное проектирование классов

На данном этапе проектирования выполнено уточнение методов проектируемых классов.

Детальное описание атрибутов и методов представлено в п. 1.5 прил. 4 «Спецификации на модули» для модуля, реализующего проектируемый пакет «Управление объектами модели».

Детальная диаграмма классов пакета «Управление объектами модели» представлена на рис. 5.31.

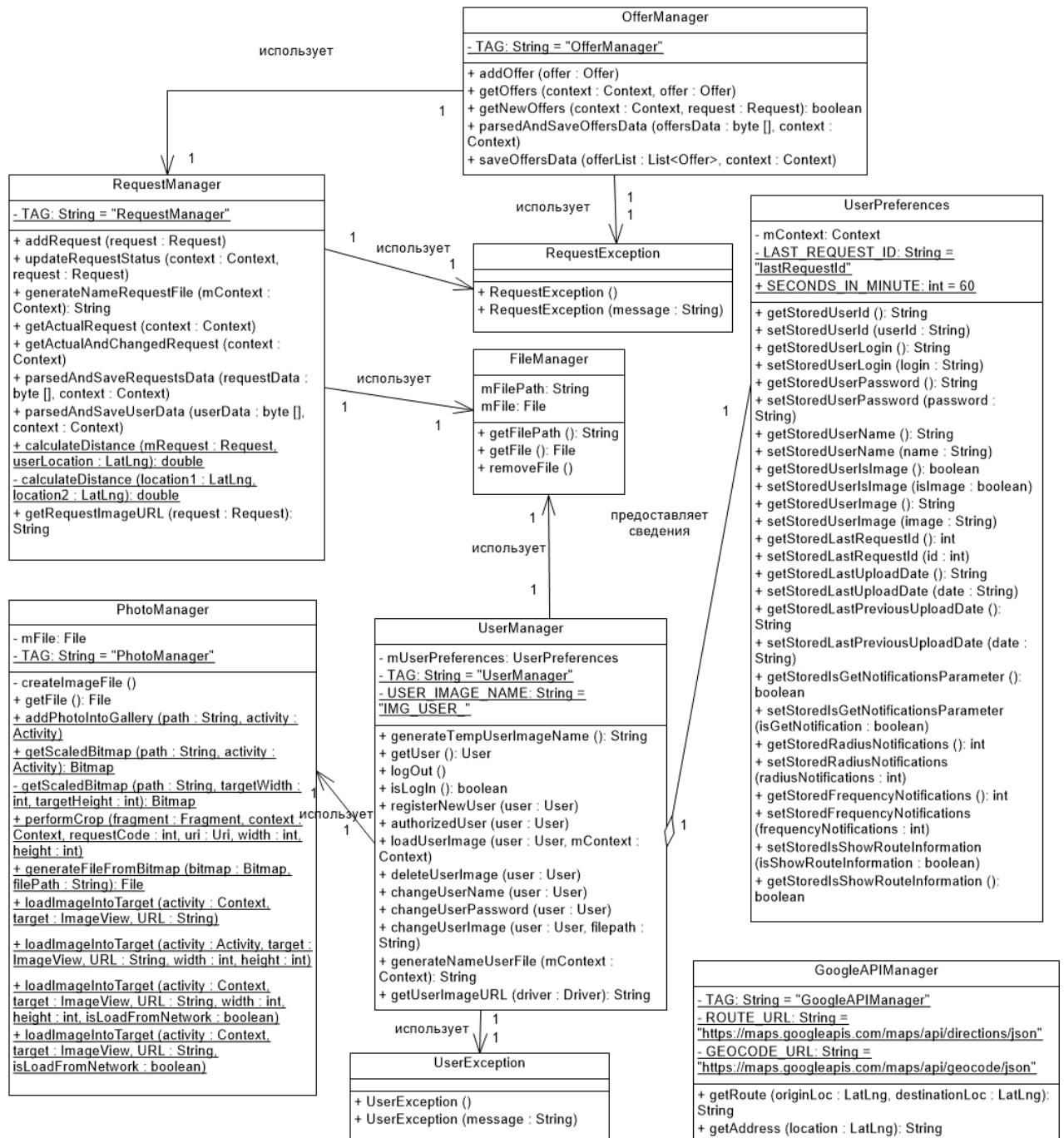


Рис. 5.31. Детальная диаграмма классов пакета «Управление объектами модели»

## 5.7 Проектирование классов пакета «Управление добавлением заявок»

### 5.7.1 Построение исходной диаграммы классов

Данный пакет предназначен для реализации окон мобильного приложения для добавления новой заявки на помощь в систему.

Список классов-кандидатов пакета «Управление добавлением заявок» представлен в табл. 5.12, связи между классами – в табл. 5.13.

Таблица 5.12

#### Классы пакета «Управление добавлением заявок»

Класс	Описание
«AddNewRequestActivity» (Активность для добавления новой заявки)	Реализует окно для добавления новой заявки
«AddNewRequestFragment» (Фрагмент для добавления новой заявки)	Реализует фрагмент окна для добавления новой заявки, который предназначен для ввода текста сообщения и выбора изображения
«AddNewRequestMapFragment» (Фрагмент-карта для добавления новой заявки)	Реализует фрагмент окна для просмотра и выбора местоположения пользователя
«SelectRequestPhotoListener» (Слушатель выбора изображения заявки)	Предназначен для передачи выбранного местоположения фрагменту окна от окна для добавления новой заявки
«EnterMessageListener» (Слушатель ввода сообщения)	Предназначен для передачи сообщения заявки окну для добавления новой заявки от фрагмента окна
«UpdateLocationListener» (Слушатель изменения местоположения)	Предназначен для передачи измененного местоположения от фрагмента окна для добавления новой заявки окну приложения
«AddNewRequestTask» (Задание для добавления заявки)	Предназначен для формирования запроса на добавление заявки в отдельном потоке

Таблица 5.13

#### Зависимости между классами пакета «Управление добавлением заявок»

Класс	Описание зависимостей
«AddNewRequestActivity»	Включает фрагменты окон: ««AddNewRequestFragment», «AddNewRequestMapFragment». Связан с классом ««AddNewRequestTask»» ассоциативной связью «запускает поток». Наследует классы: ««SelectRequestPhotoListener», «EnterMessageListener» - для получения выбранного изображения и заявки от фрагментов окна. Наследует класс ««SelectLocationListener»» для получения выбранного местоположения от фрагмента окна ««AddNewRequestMapFragment»»
«AddNewRequestMapFragment»	Наследует класс ««UpdateLocationListener»» для получения местоположения от окна ««AddNewRequestActivity»»

Исходная диаграмма классов проектируемого пакета представлена на рис. 5.32.

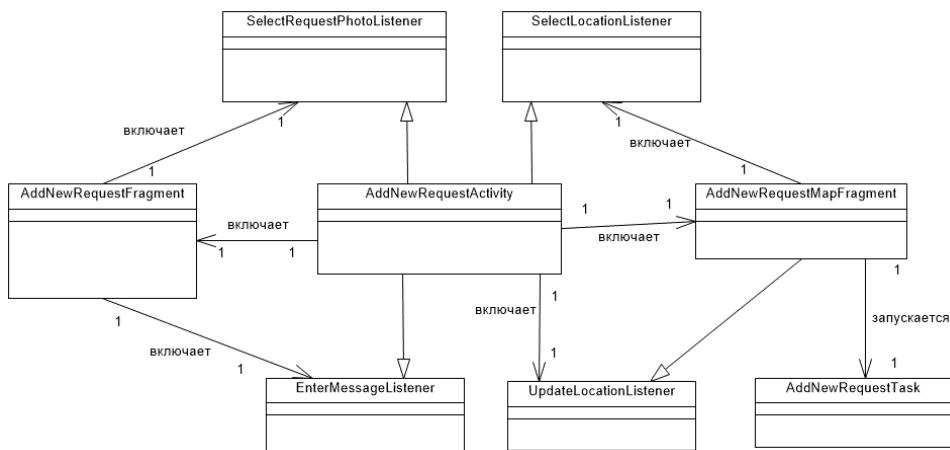


Рис. 5.32. Исходная диаграмма классов пакета «Управление добавлением заявок»

### 5.7.2 Построение диаграмм последовательностей действий

Для реализации сценария добавления заявки на помощь построены диаграммы последовательностей действий (рис. 5.33 – 5.35), которые показывают последовательность обмена сообщениями объектов проектируемых классов: «AddNewRequestActivity», «AddNewRequestFragment», «AddNewRequestMapFragment», «AddNewRequestTask» - проектируемого пакета «Управление добавлением заявок» («ManagementAddNewRequest»); объекта класса «Request» пакета «Объекты модели» («Object Of Model»).

Сценарий заключается в следующем: пользователь запускает окно для добавления новой заявки «AddNewRequestActivity», которое включает вкладки: «AddNewRequestFragment», «AddNewRequestMapFragment», «AddNewRequestActivity». При получении значения местоположения пользователя окно передает это значение фрагменту «AddNewRequestMapFragment» для построения карты с отметкой текущего местоположения пользователя. Далее пользователь выбирает местоположение на карте вкладки «AddNewRequestMapFragment», вводит сообщение в окне ввода и выбирает фотографию на вкладке «AddNewRequestMapFragment». Выбранные

пользователем данные передаются от фрагментов окну «AddNewRequestActivity». При нажатии кнопки для сохранения данных создается объект «Request», который заполняется введенными данными заявки и при помощи объекта «AddNewRequestTask» передается серверной части приложения для сохранения в базе данных MySQL. По окончании процесса сохранения пользователю показывается сообщение о добавлении новой заявки на помощь в систему.

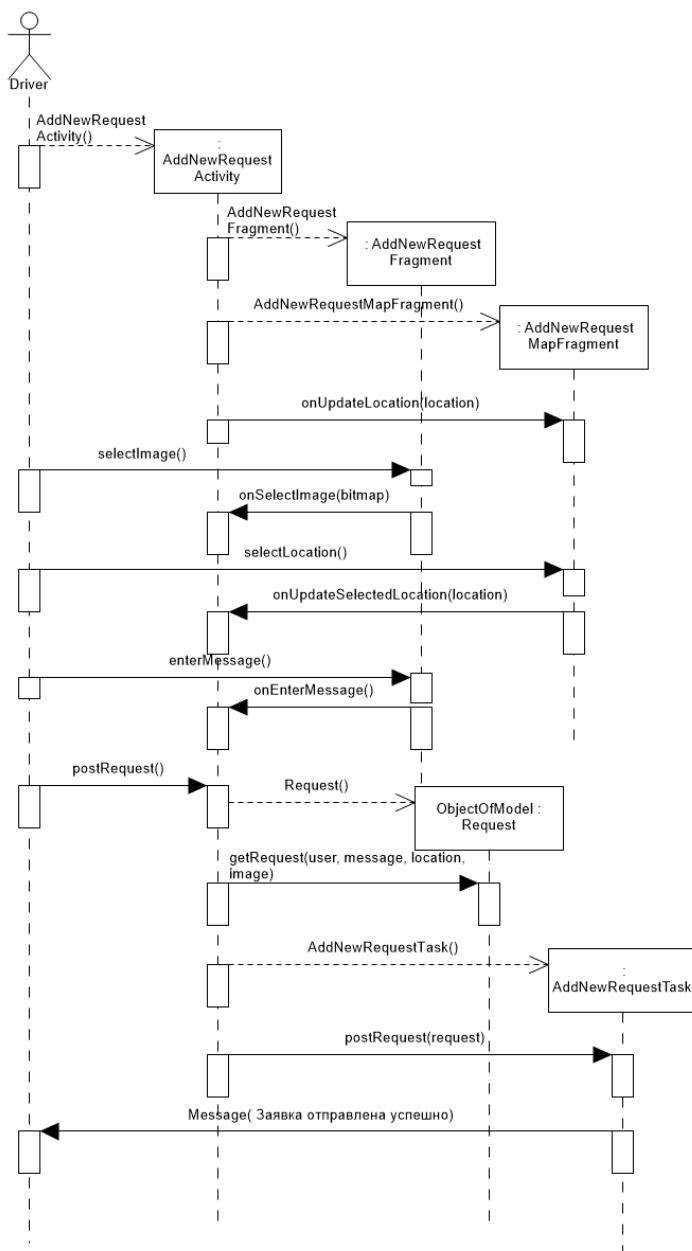


Рис. 5.33. Диаграмма последовательностей действий сценария «Добавление новой заявки». Типичных ход событий

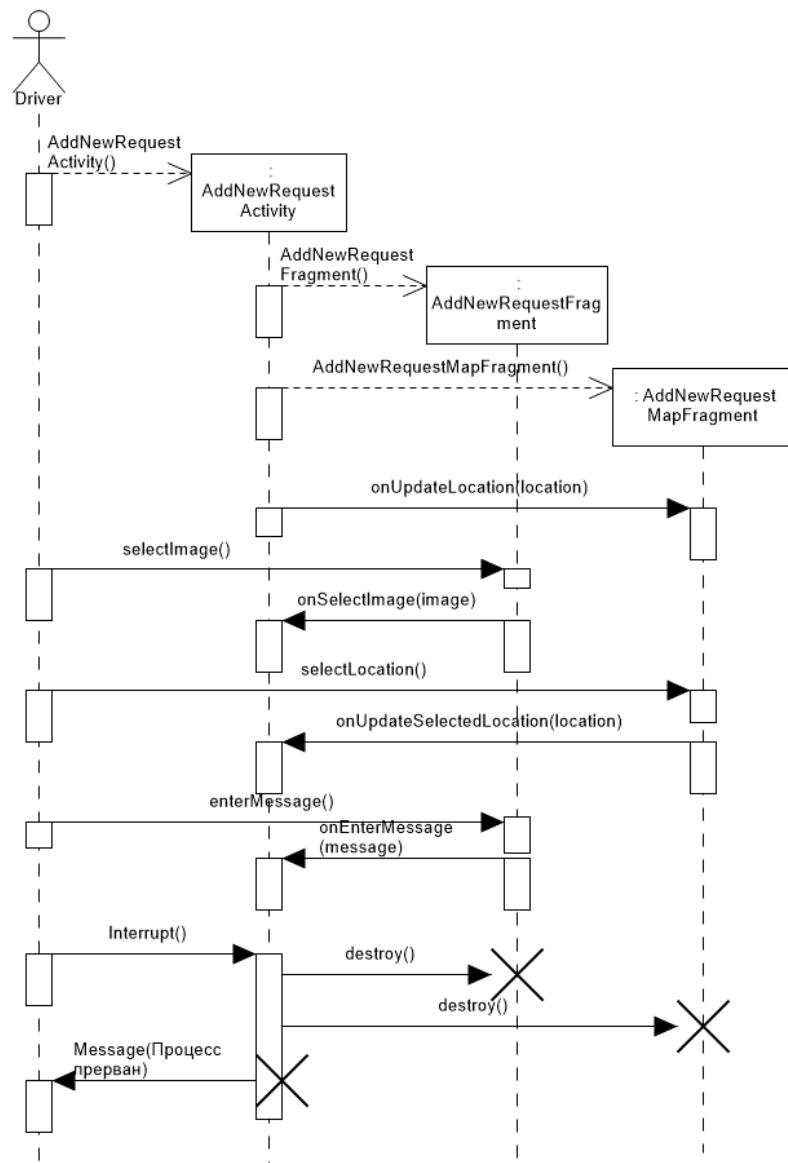


Рис. 5.34. Диаграмма последовательностей действий сценария «Добавление новой заявки». Прерывание пользователя

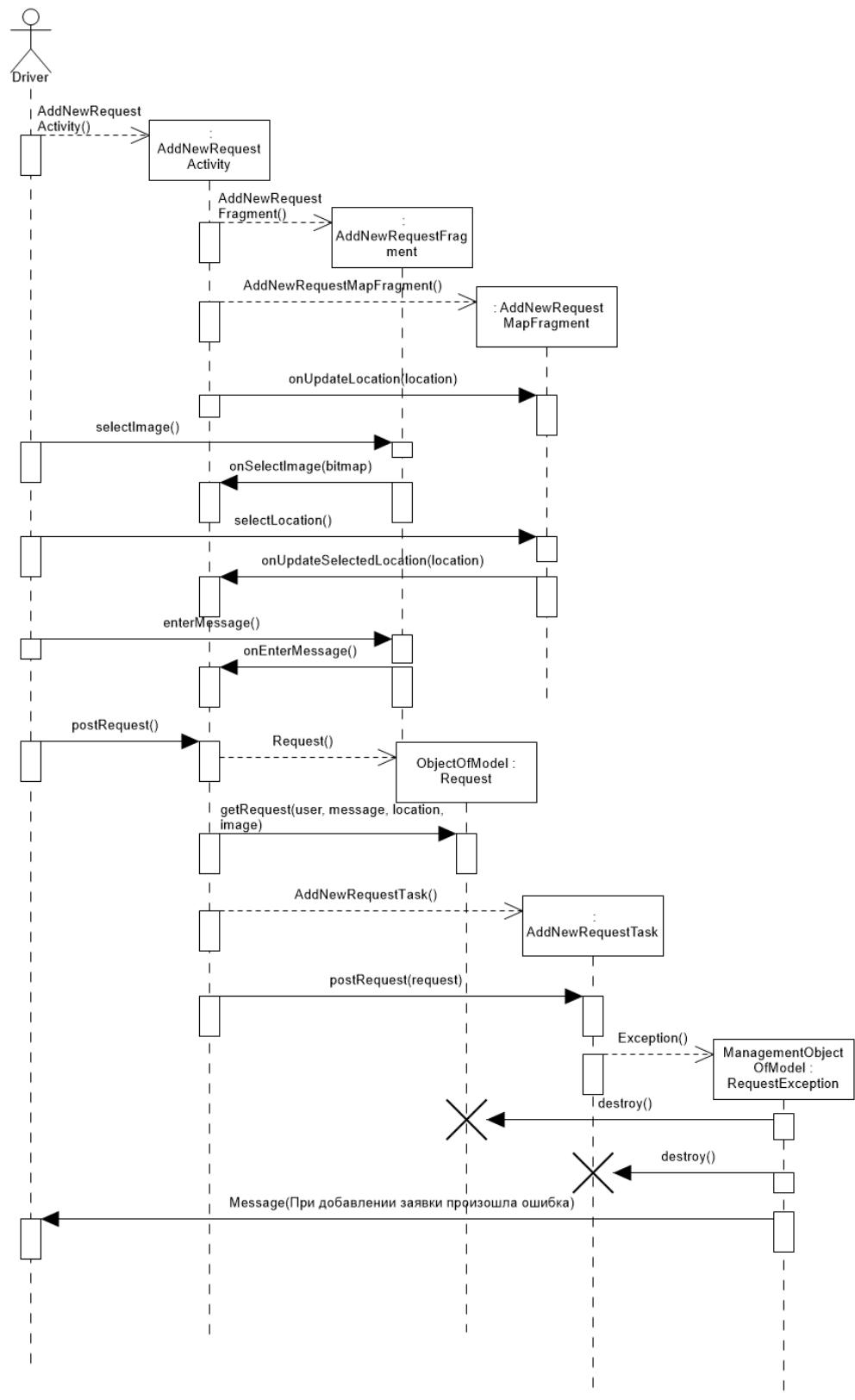


Рис. 5.35. Диаграмма последовательностей действий сценария «Добавление новой заявки». Возникновение исключительной ситуации

### 5.7.3 Построение уточненной диаграммы классов

На данном этапе уточнены связи проектируемых классов.

Классы: «SelectRequestPhotoListener», «SelectLocationListener», «EnterMessageListener» - являются интерфейсными классами и содержат методы для передачи выбранных значений: изображения, местоположения, сообщения заявки - объекту класса «AddNewRequestActivity», который наследует данный интерфейс. Для реализации передачи данных окну приложения от его фрагментов класс «AddNewRequestFragment» содержит ссылки на объекты классов: «SelectRequestPhotoListener», «EnterMessageListener»; класс «AddNewRequestMap Fragment» содержит ссылку на объект «EnterMessageListener».

Класс «UpdateLocationListener» является интерфейсным классом, содержит метод для передачи выбранного значения местоположения объекту класса «AddNewRequestMapFragment», который наследует данный интерфейс.

Уточненная диаграмма классов данного пакета представлена на рис. 5.36.

### 5.7.4 Детальное проектирование классов

На данном этапе проектирования выполнено уточнение атрибутов, методов проектируемых классов. Классы: «AddNewRequestActivity», «AddNewRequestMapFragment», «AddNewRequestFragment» - содержат атрибуты, представляющие объекты пользовательского интерфейса для добавления новой заявки, и методы для управления ими. Класс «AddNewRequestTask» содержит методы для организации процесса передачи данных заявки серверной части приложения в фоновом режиме.

Детальное описание атрибутов и методов представлено в п. 1.6 прил. 4 «Спецификации на модули» для модуля, реализующего проектируемый пакет «Управление добавлением заявок».

Детальная диаграмма классов пакета «Управление добавлением заявок» представлена на рис. 5.37.

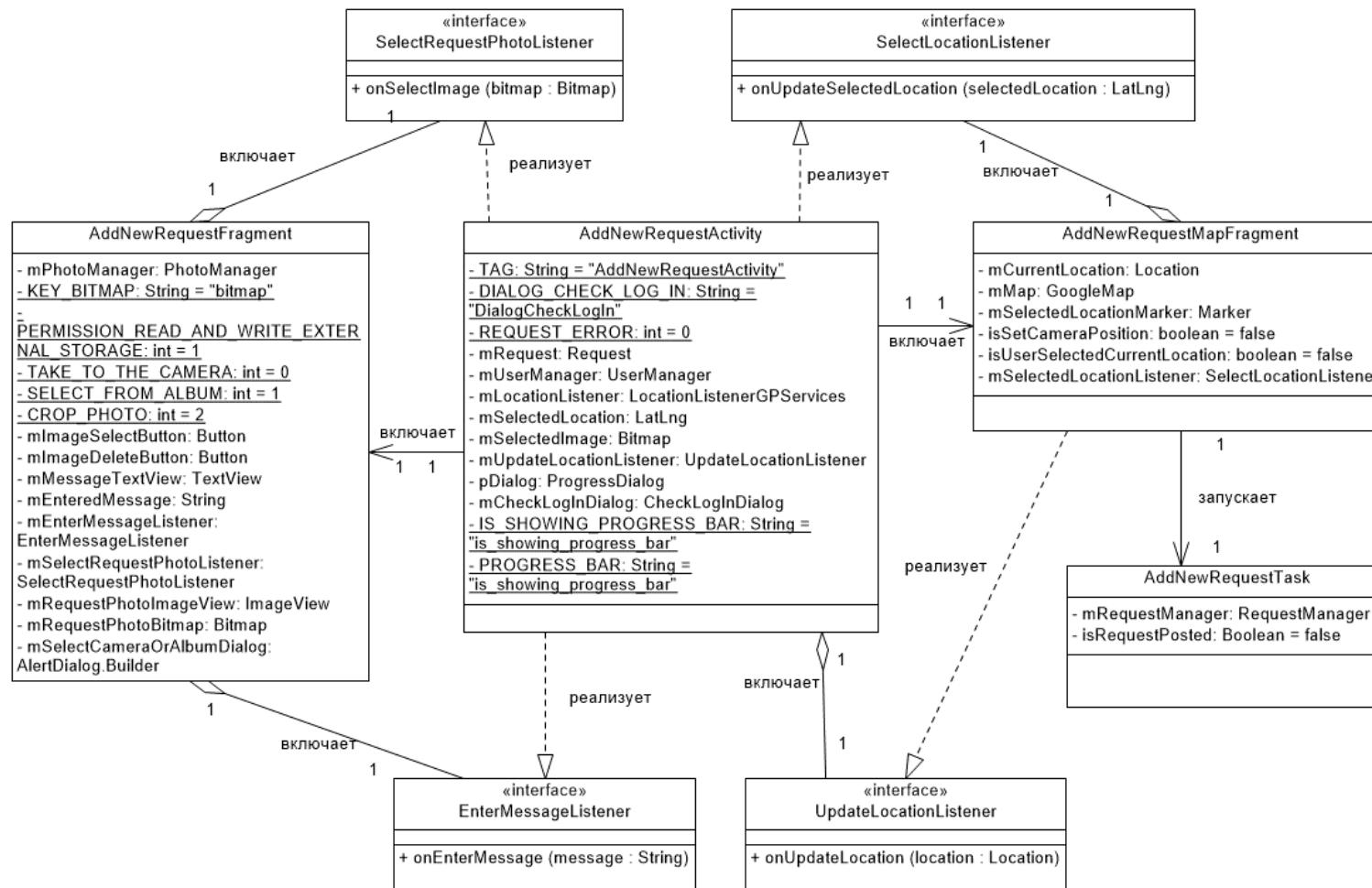


Рис. 5.36. Уточненная диаграмма классов пакета «Управление добавлением заявок»

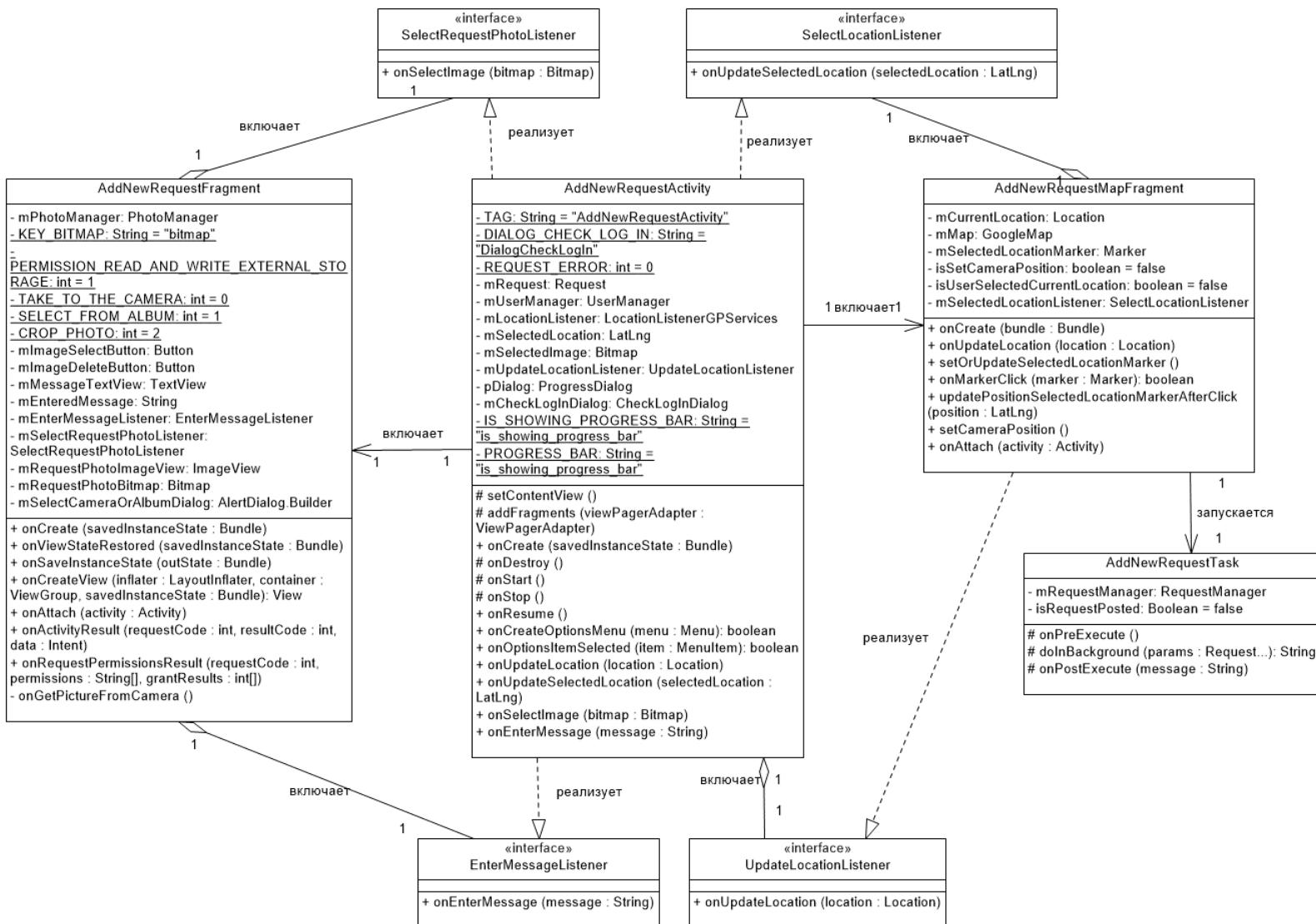


Рис. 5.37. Детальная диаграмма классов пакета «Управление добавлением заявок»

## 5.8 Проектирование классов пакета «Управление пользователем»

### 5.8.1 Построение исходной диаграммы классов

Пакет предназначен для реализации окон приложения для управления регистрацией, авторизацией пользователя, учетной записью. Список классов пакета представлен в табл. 5.14, зависимости между классами – в табл. 5.15.

Таблица 5.14

#### Классы пакета «Управление пользователем»

Класс	Описание
UserSettingsActivity	Окно для изменения настроек пользователя
UserSettingsFragment	Фрагмент окна для изменения настроек пользователя
EnterActivity	Окно для авторизации пользователя
EnterFragment	Фрагмент окна для авторизации пользователя
AuthorizeNewUserTask	Задание для авторизации пользователя
RegistrationActivity	Окно для авторизации пользователя в приложении
RegistrationFragment	Фрагмент окна для регистрации пользователя
ReqisterNewUserTask	Задание для регистрации пользователя
UserAccountActivity	Окно учетной записи пользователя
UserAccountFragmant	Фрагмент окна учетной записи пользователя
ChangeImageUserTask	Задание для изменения фотографии пользователя
DeleteImageUserTask	Задание для удаления фотографии пользователя
SetNameDialog	Диалог для изменения имени пользователя
ChangeNameUserTask	Задание для изменения имени пользователя
SetUserPasswordDialog	Диалоговое окно для изменения пароля пользователя
ChangePasswordUserTask	Задание для изменения пароля пользователя

Таблица 5.15

#### Зависимости классов пакета «Управление пользователем»

Классы	Описание
EnterActivity, UserAccountActivity, RegistrationActivity	Включают фрагменты окон: «EnterFragment», «UserAccountFragment», «RegistrationFragment»
UserAccountFragmant	Запускает диалоговые окна: «SetNameDialog», «SetUserPasswordDialog», которые запускают фоновые задания: «ChangeNameUserTask», «ChangePasswordUserTask», соответственно
UserAccountFragment	Запускает фоновые задания: «ChangeImageUserTask», «DeletePasswordUserTask»
RegistrationFragment, EnterFragment»	Запускают фоновые задания: «ReqisterNewUserTask», «AuthorizeNewUserTask», соответственно
AuthorizeNewUserTas	Задание для авторизации пользователя

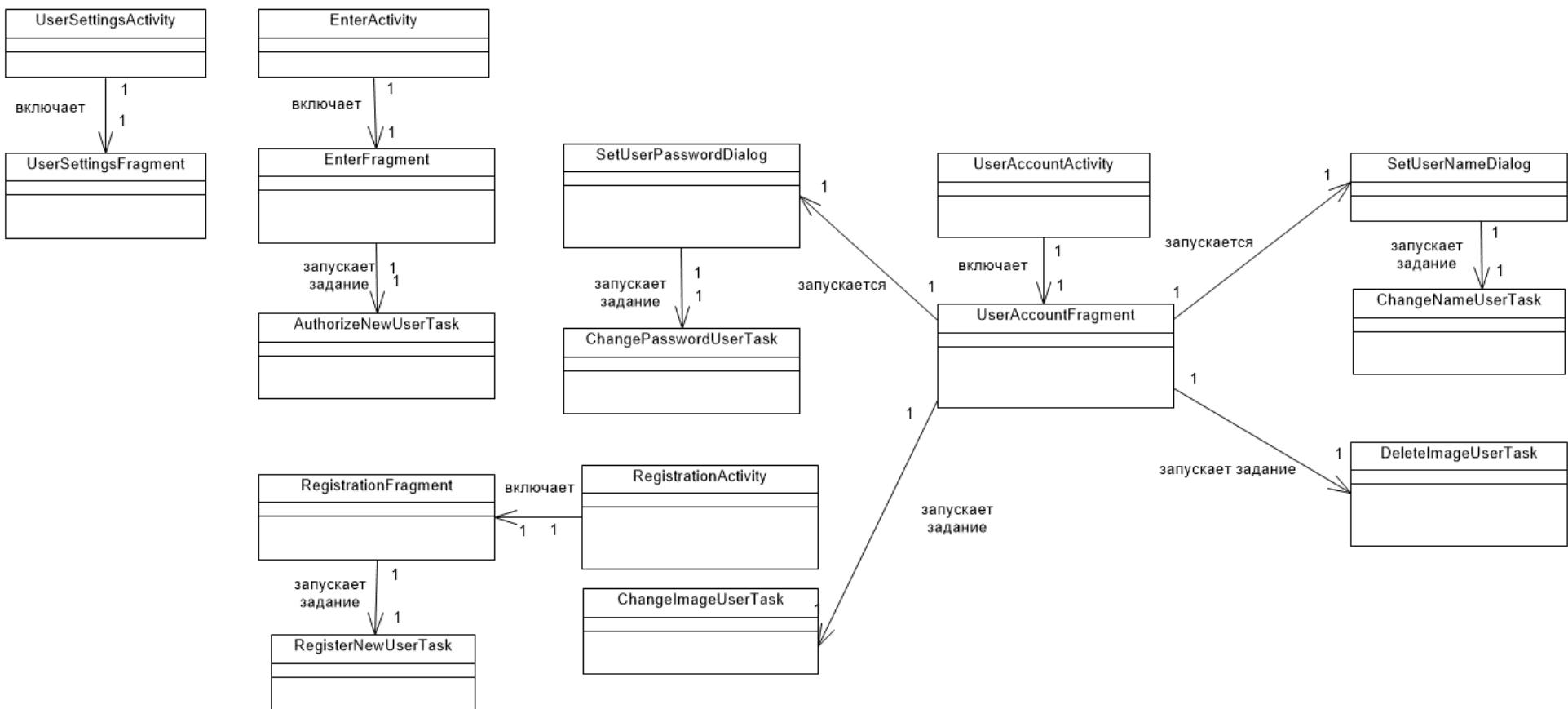


Рис. 5.38. Исходная диаграмма классов пакета «Управление пользователем»

### 5.8.2 Построение диаграмм последовательностей действий

Для реализации сценария авторизации пользователя в приложении и редактирования имени и фотографии пользователя построены диаграммы последовательностей действий, которые показывают последовательность обмена сообщениями объектов классов: «EnterActivity», «EnterFragment», «AuthorizeNewUserTask», «UserAccountActivity», «UserAccountFragment», «SetUserNameDialog», «ChangeNameUserTask», «ChangeImageUserTask» - проектируемого пакета «Управление пользователем».

Сценарий заключается в следующем: пользователь запускает окно для авторизации в приложении, создавая объекты: «EnterActivity», «EnterFragment». Затем он вводит логин и пароль, которые объект «AuthorizeNewUserTask» принимает и передает серверной части приложения для авторизации пользователя.

Пройдя авторизацию, пользователь запускает окно «UserAccountActivity», содержащее фрагмент «UserAccountFragment», и изменяет имя: для этого запускает диалог «SetUserNameDialog», в котором вводит значение имени, отправляет данные имени серверной части приложения, используя объект «ChangeNameUserTask». Далее пользователь загружает изображение из памяти устройства и сохраняет его, используя объект «ChangeUserImageTask».

Диаграммы последовательности действий объектов в процессе авторизации в системе и редактировании настроек пользователя представлены на рис. 5.39 - 5.41.

### 5.8.3 Детальное проектирование классов

На данном этапе проектирования выполнено уточнение методов и атрибутов проектируемых классов. Детальное описание классов представлено в п 1.8 прил. 4 «Спецификации на модули» для модуля, реализующего проектируемый пакет «Управление пользователем», детальная диаграмма которого представлена на рис. 5.42.

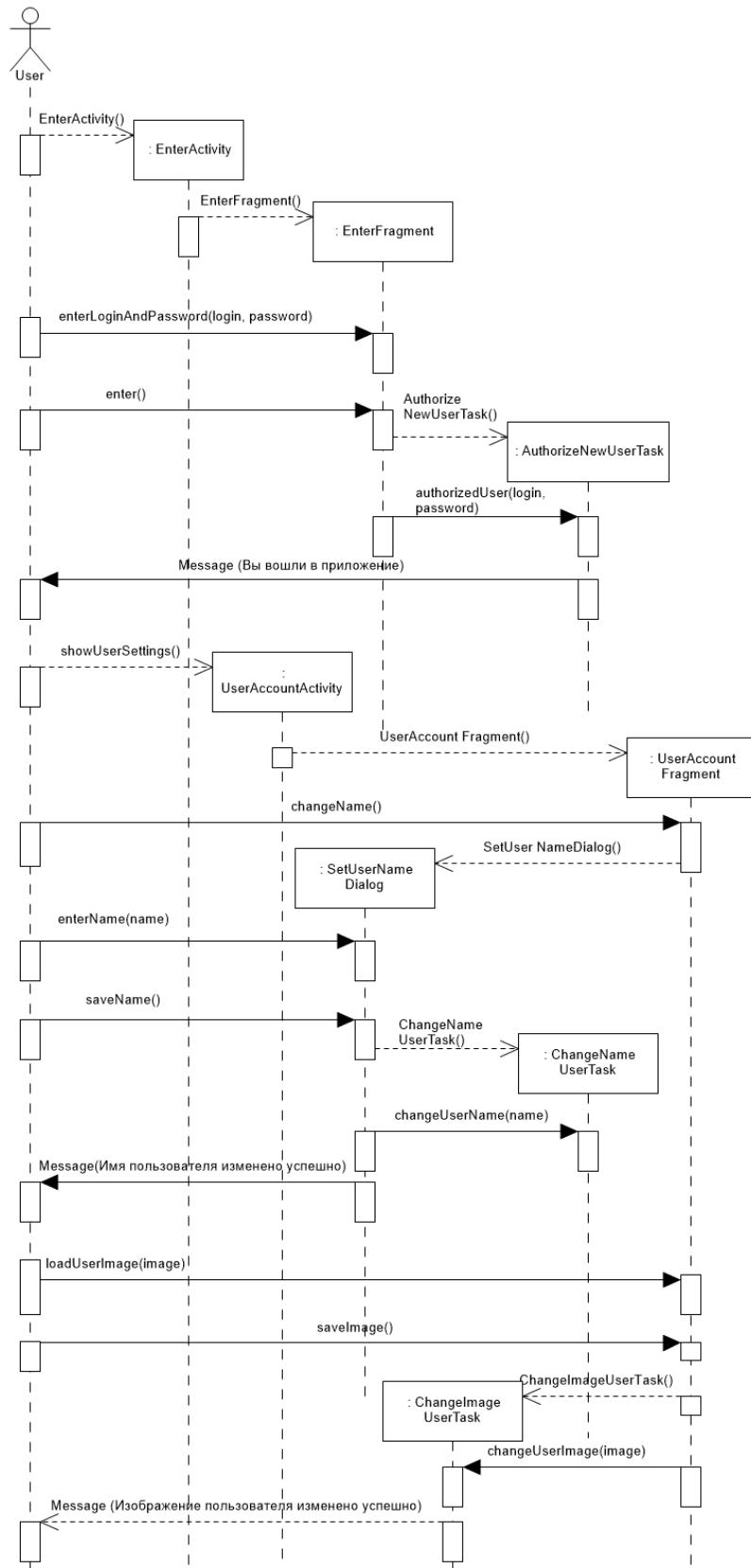


Рис. 5.39. Диаграмма последовательностей действий сценария «Авторизация и редактирование настроек пользователя». Типичных ход событий

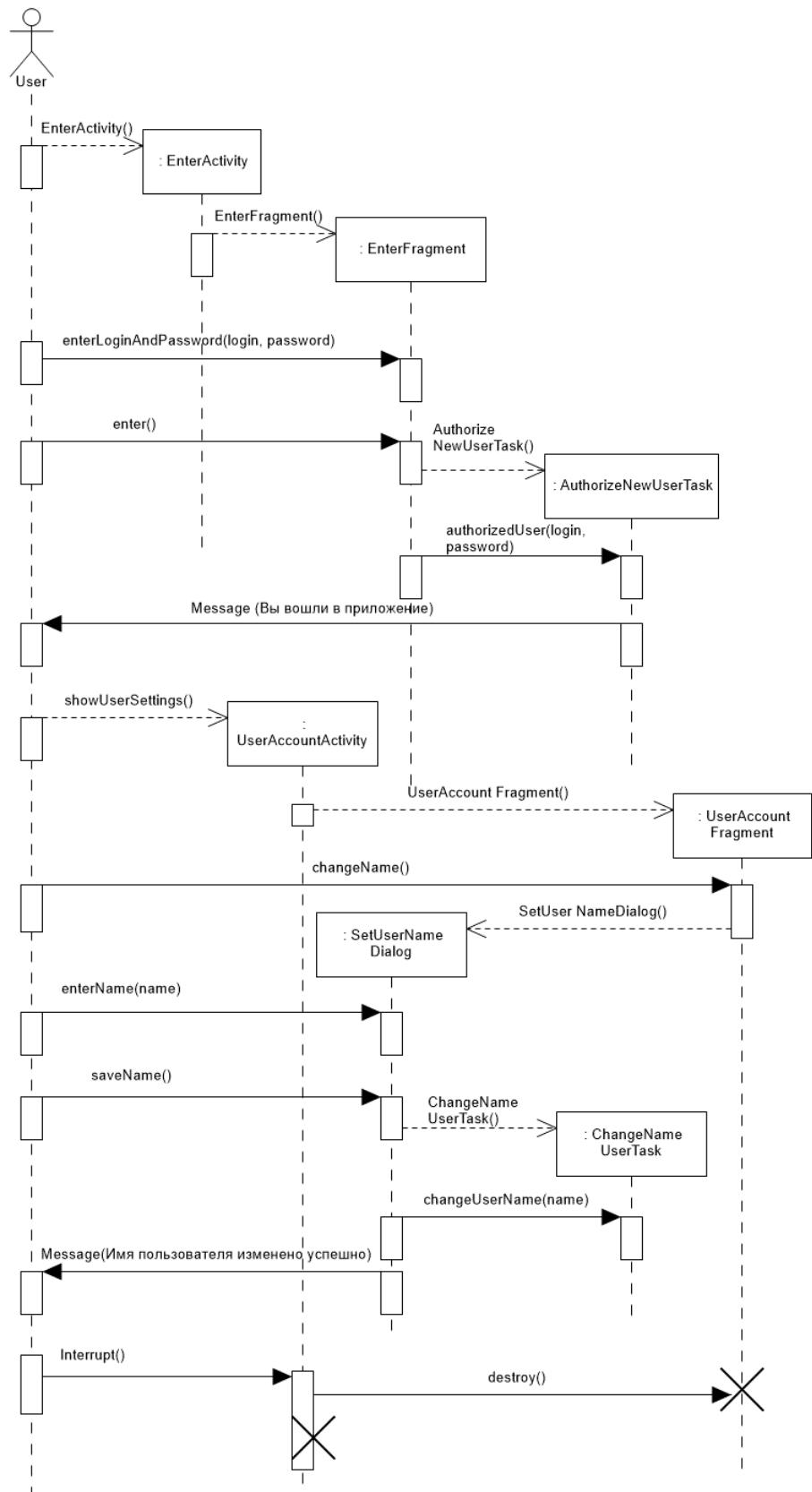


Рис. 5.40. Диаграмма последовательностей действий сценария «Авторизация и редактирование настроек пользователя». Прерывание пользователя

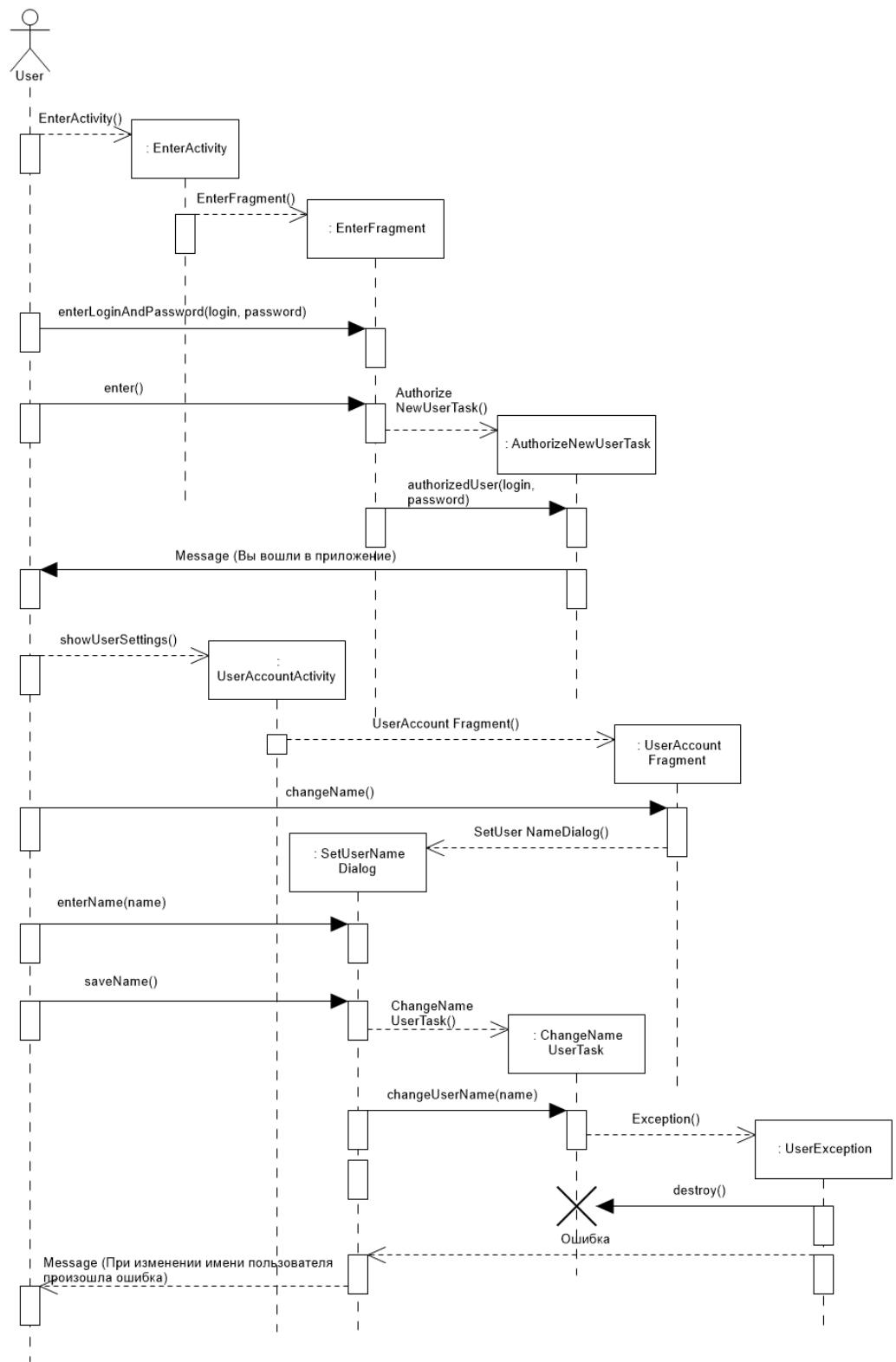


Рис. 5.41. Диаграмма последовательностей действий сценария «Авторизация и редактирование настроек пользователя». Исключительная ситуация

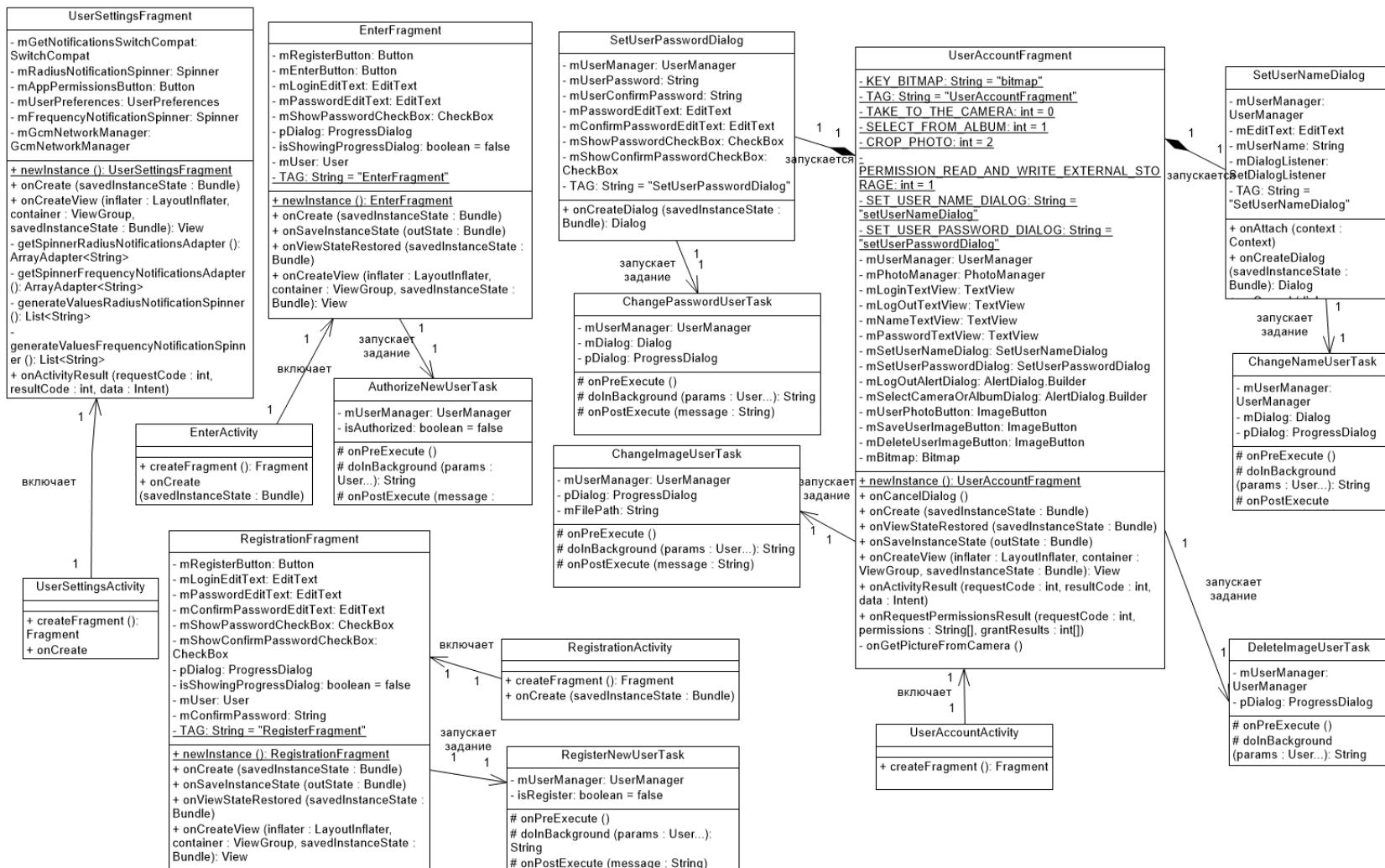


Рис. 5.42. Детальная диаграмма классов пакета «Управление пользователем»

## 5.9 Проектирование классов пакета «Управление списком заявок»

### 5.9.1 Построение исходной диаграммы классов

Данный пакет предназначен для реализации окон мобильного приложения для отображения актуальных заявок на помощь в виде списка и отметок маркеров заявок на карте, просмотра сведений выбранной заявки, добавления сообщений к выбранной заявке, построения маршрута до выбранной заявки.

Список классов-кандидатов пакета «Управление списком заявок» представлен в табл. 5.16.

Таблица 5.16

Классы пакета «Управление списком заявок»

Класс	Описание
MainActivity	Реализует окно для просмотра списка заявок и отметок заявок на карте
RequestListFragment	Фрагмент окна для просмотра списка заявок
RequestHolder	Элемент списка заявок
RequestAdapter	Управление элементом списка заявок
RequestListMapFragment	Фрагмент окна для просмотра отметок заявок на карте
AuthorizeActivity	Окно для авторизации пользователя
GetActualRequestTask	Задание для загрузки списка актуальных заявок от серверной части приложения
RequestDetailActivity	Активность для просмотра сведений детальной заявки
RequestDetailFragment	Фрагмент окна для просмотра сведений детальной заявки
RequestDetailMapFragment	Реализует окно для просмотра маршрута заявки
OfferHolder	Реализует объект для управления списком сообщений пользователей к заявке
OfferAdapter	Реализует объект для управления списком сообщений пользователей к заявке
AddNewOfferTask	Задание для добавления нового предложения помощи
UpdateOffersTask	Задание для обновления списка предложений помощи
UpdateRequestStatusTask	Задание для изменения статуса заявки
GetNewOffersTask	Задание для получения списка новых предложений помощи
GetAddressRequestTask	Задание для получения адреса заявки: реализует запрос к серверу Google для получения адреса заявки
GetRouteTask	Задание для построения маршрута заявки: реализует запрос к серверу Google для получения маршрута между местоположением добавленной заявки и текущим местоположением пользователя

Исходная диаграмма классов проектируемого пакета представлена на рис. 5.43.

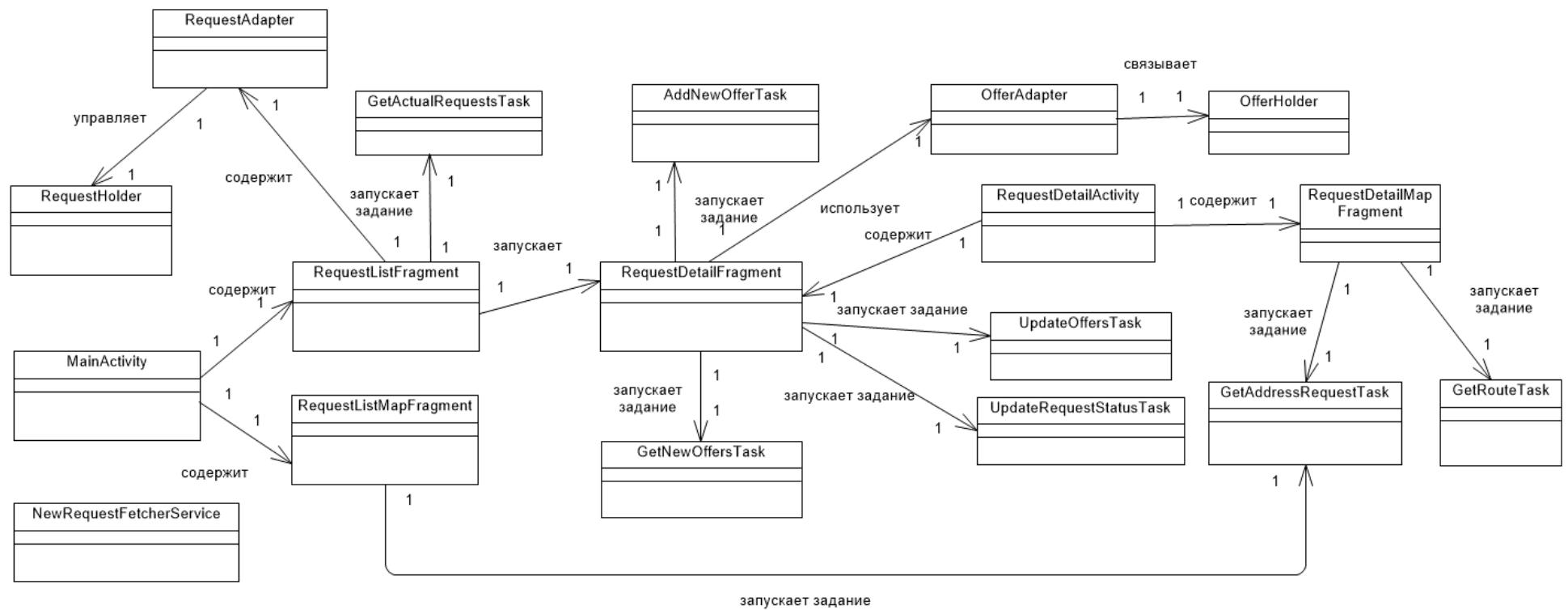


Рис. 5.43. Исходная диаграмма классов пакета «Управление списком заявок»

### 5.9.2 Построение диаграмм последовательностей действий

Для реализации сценария получения списка заявок показана последовательность обмена сообщениями объектов классов: «*MainActivity*», «*RequestListFragment*», «*RequestListMapFragment*», «*GetActualRequestsTask*», «*RequestAdapter*», «*RequestHolder*» - проектируемого пакета «Управление списком заявок».

Сценарий заключается в следующем: пользователь запускает приложение, при запуске которого создается окно «*MainActivity*», которое запускает: фрагмент окна «*RequestListFragment*» для просмотра списка заявок и фрагмент «*RequestMapFragment*» для просмотра отметок заявок на карте. Далее объект «*GetActualRequestTask*» получает список актуальных заявок.

После этого создается объект «*RequestAdapter*», который циклически создает объект «*RequestHolder*» для каждой полученной заявки из списка. Объект «*RequestHolder*» связывает данные заявки с её графическим представлением в списке.

По завершению данного процесса пользователю выдается сообщение об успешной загрузке списка актуальных заявок с сервера.

В случае возникновения исключительной ситуации создается объект «*RequestException*», который прерывает указанный процесс получения заявок и показывает сообщение пользователю о том, что произошла ошибка при загрузке списка актуальных заявок на помощь.

Диаграмма последовательностей действий объектов в процессе загрузки списка заявок с сервера на мобильное устройство пользователя представлена на рис. 5.44.

Ситуация пользовательского прерывания в ходе выполнения рассматриваемого сценария представлена на рис. 5.45.

Ситуация возникновение исключительной ситуации при загрузке списка заявок с сервера на мобильное устройство пользователя представлена на рис. 5.46.

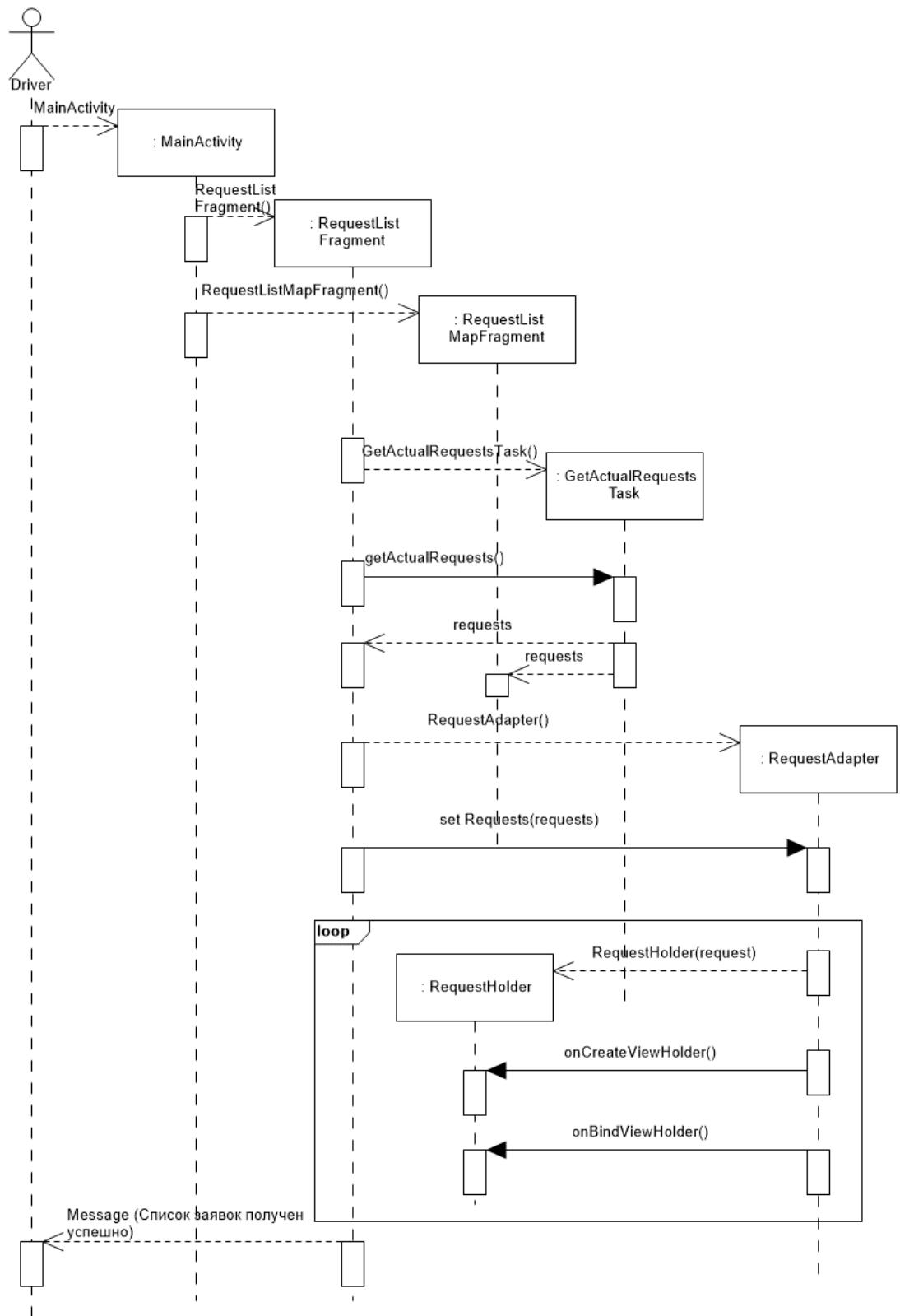


Рис. 5.44. Диаграмма последовательностей действий сценария «Получение списка заявок». Типичных ход событий

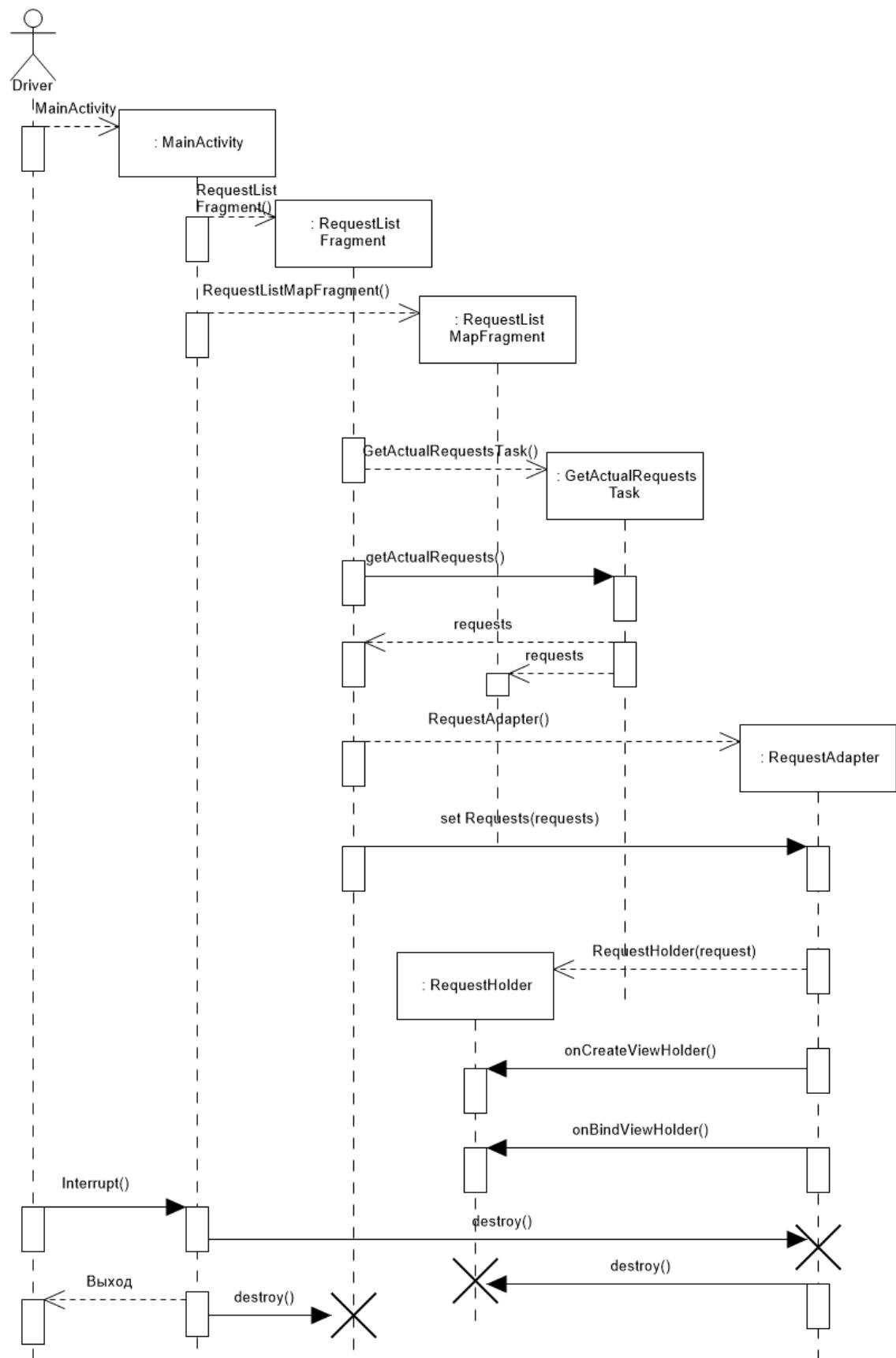


Рис. 5.45. Диаграмма последовательностей действий сценария «Получение списка заявок». Прерывание пользователя

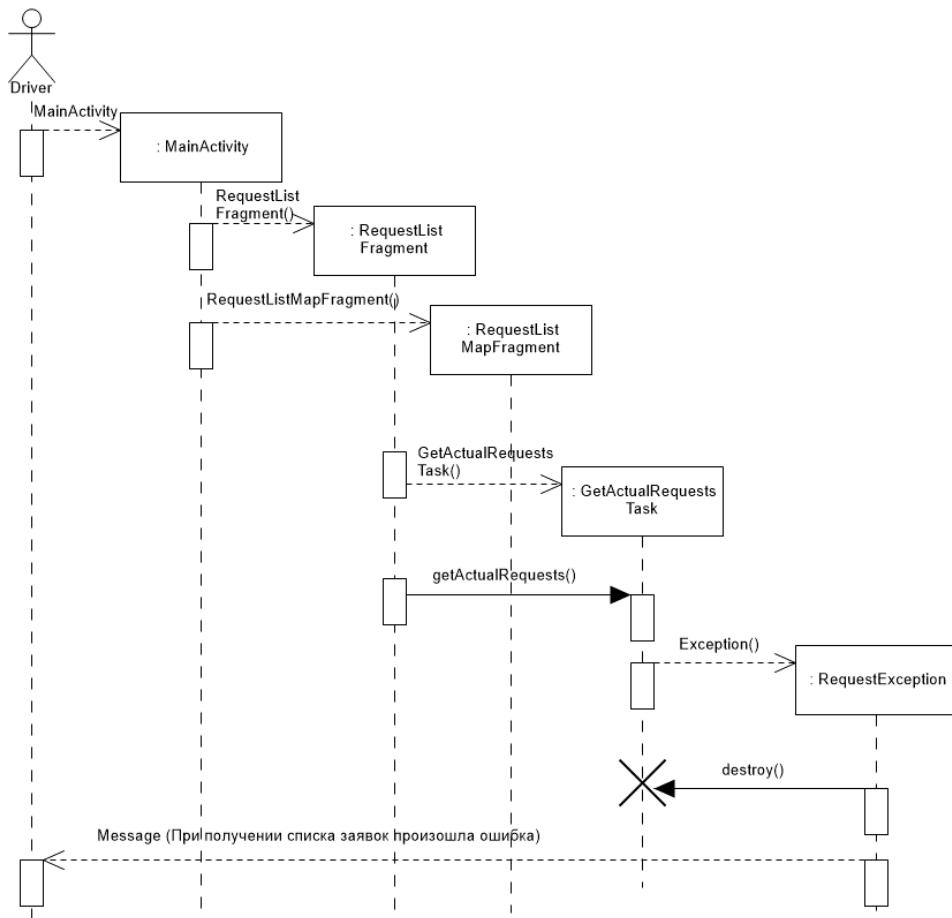


Рис. 5.46. Диаграмма последовательностей действий сценария «Получение списка заявок». Исключительная ситуация

### 5.9.3 Детальное проектирование классов

На этом этапе проектирования выполнено уточнение методов и атрибутов проектируемых классов. Классы, отвечающие за реализацию окон и их фрагментов, содержат графические компоненты и методы для работы с ними. Классы, реализующие задание для получения и загрузки данных от серверной части приложения, содержат методы для реализации данного взаимодействия.

Детальное описание классов представлено в п. 1.7 прил. 4 «Спецификации на модули» для модуля, реализующего проектируемый пакет «Управление списком заявок». Детальная диаграмма классов проектируемого пакета представлена на рис. 5.47.

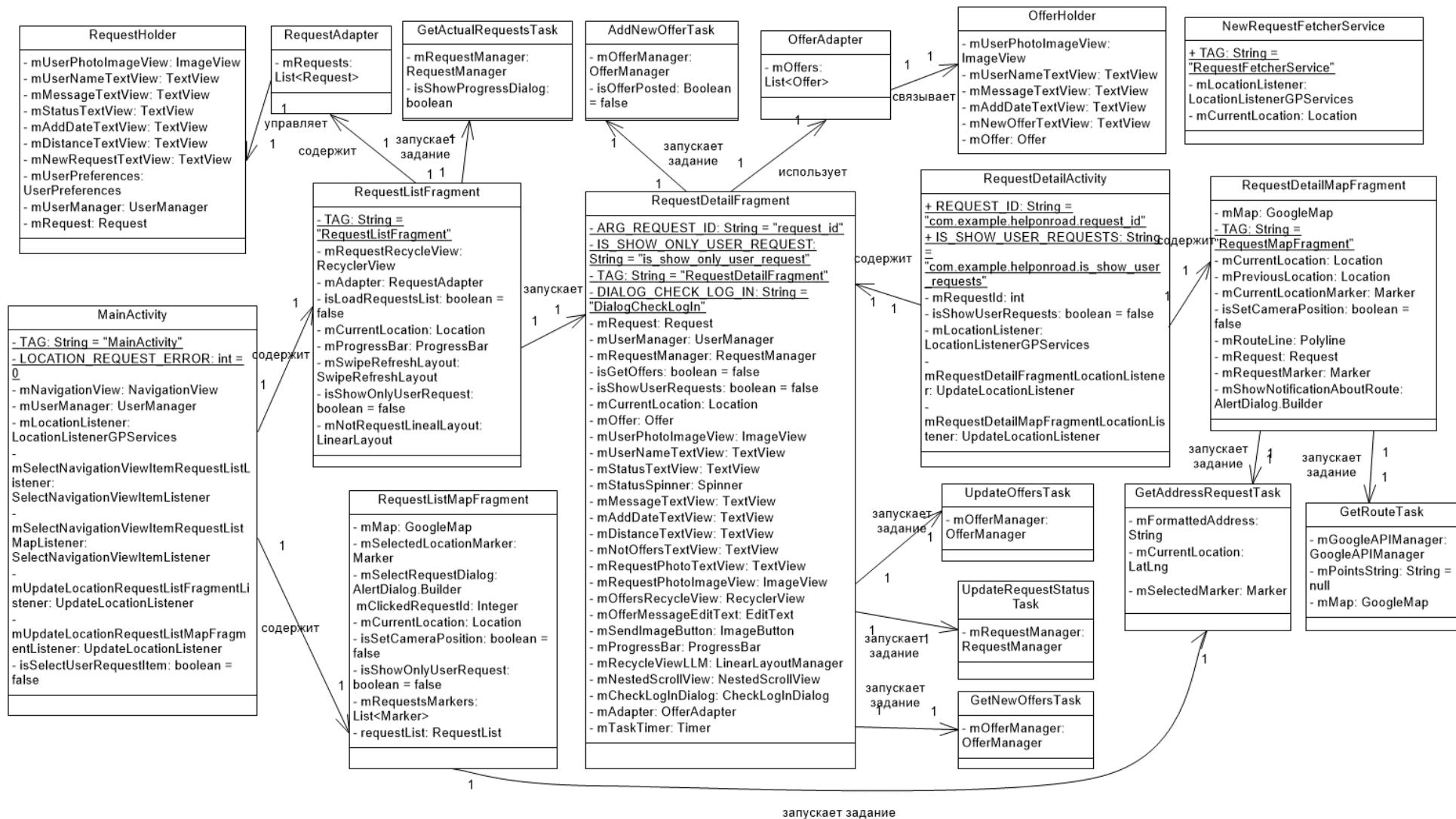


Рис. 5.47. Детальная диаграмма классов пакета «Управление списком заявок»

## 5.10 Проектирование классов пакета «Интерфейс к базе данных MySQL»

### 5.10.1 Построение исходной диаграммы классов

Данный пакет предназначен для реализации классов серверной части приложения, которые содержат методы, выполняющие запросы к базе данных MySQL: на запись сведений в базу о новой заявке на помощь и предложении помощи, на получение списка заявок и предложений помощи из базы, на регистрацию и авторизацию пользователей в системе. Список классов-кандидатов пакета «Управление добавлением заявок» представлен в табл. 5.17.

Таблица 5.17

Классы пакета «Интерфейс к базе данных MySQL»

Класс	Описание
RequestsAPI	Предоставляет набор методов для работы с заявками на помощь
OffersAPI	Предоставляет набор методов для работы с предложениями помощи
UserAPI	Предоставляет набор методов для работы с данными о пользователях
FilesAPI	Предоставляет набор методов для графических файлов заявок на помощь и пользователей системы

Имеются следующие связи между классами. Классы: «UserAPI», «RequestsAPI» - «используют» методы класса «FilesAPI» для добавления файла пользователя и файла заявки на помощь, соответственно.

Исходная диаграмма классов проектируемого пакета представлена на рис. 5.48.

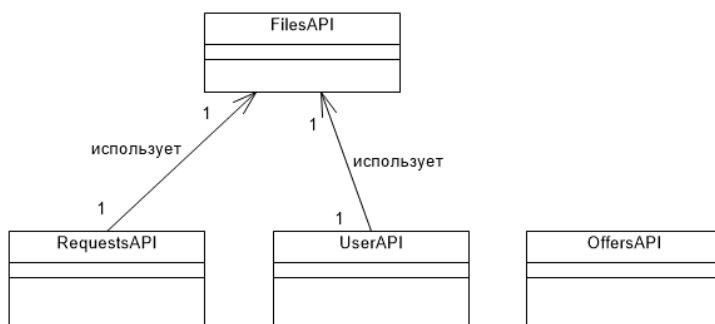


Рис. 5.48. Исходная диаграмма классов пакета «Интерфейс к базе данных MySQL»

### 5.10.2 Построение диаграмм последовательностей действий

На данном этапе построены диаграммы последовательностей действий, которые показывают последовательность обмена сообщениями объектов классов клиентской части приложения: «UserManager», «RequestManager» - пакета «Управление объектами модели» («ManagementObjectOfModel») и объектов проектируемых классов: «UserAPI», «RequestAPI», «FilesAPI» - пакета «Интерфейс к базе данных MySQL» серверной части приложения в процессе регистрации пользователя и добавления новой заявки в систему.

Сценарий заключается в следующем: пользователь приложения на мобильном устройстве вводит логин, пароль, имя, выбирает фотографию для регистрации в системе. Клиентская часть приложения, посредством объекта класса «UserManager», отправляет POST HTTP-запрос серверной части приложения на регистрацию нового пользователя. Серверная часть приложения принимает и обрабатывает полученный запрос, в процессе обработки которого, посредством объекта «UserManager», делает запрос к базе данных на добавление нового пользователя и сохранение его данных (имя, логин, пароль); посредством объекта «FileManager», выполняет сохранение файла фотографии изображения пользователя в директории на сервере. Создав нового пользователя, серверная часть отправляет ответ объекту клиентской части «UserManager» о регистрации нового пользователя в системе.

Аналогичным образом, с использованием объекта «RequestManager», осуществляется вторая часть рассматриваемого сценария - добавление новой заявки на помощь в систему.

Диаграмма последовательности действий в процессе регистрации пользователя и добавления новой заявки в систему представлена на рис. 5.49.

В случае возникновения исключительной ситуации серверная часть приложения отправляет сообщение клиентской части о возникшей ошибке в процессе регистрации или добавлении новой заявки в систему.

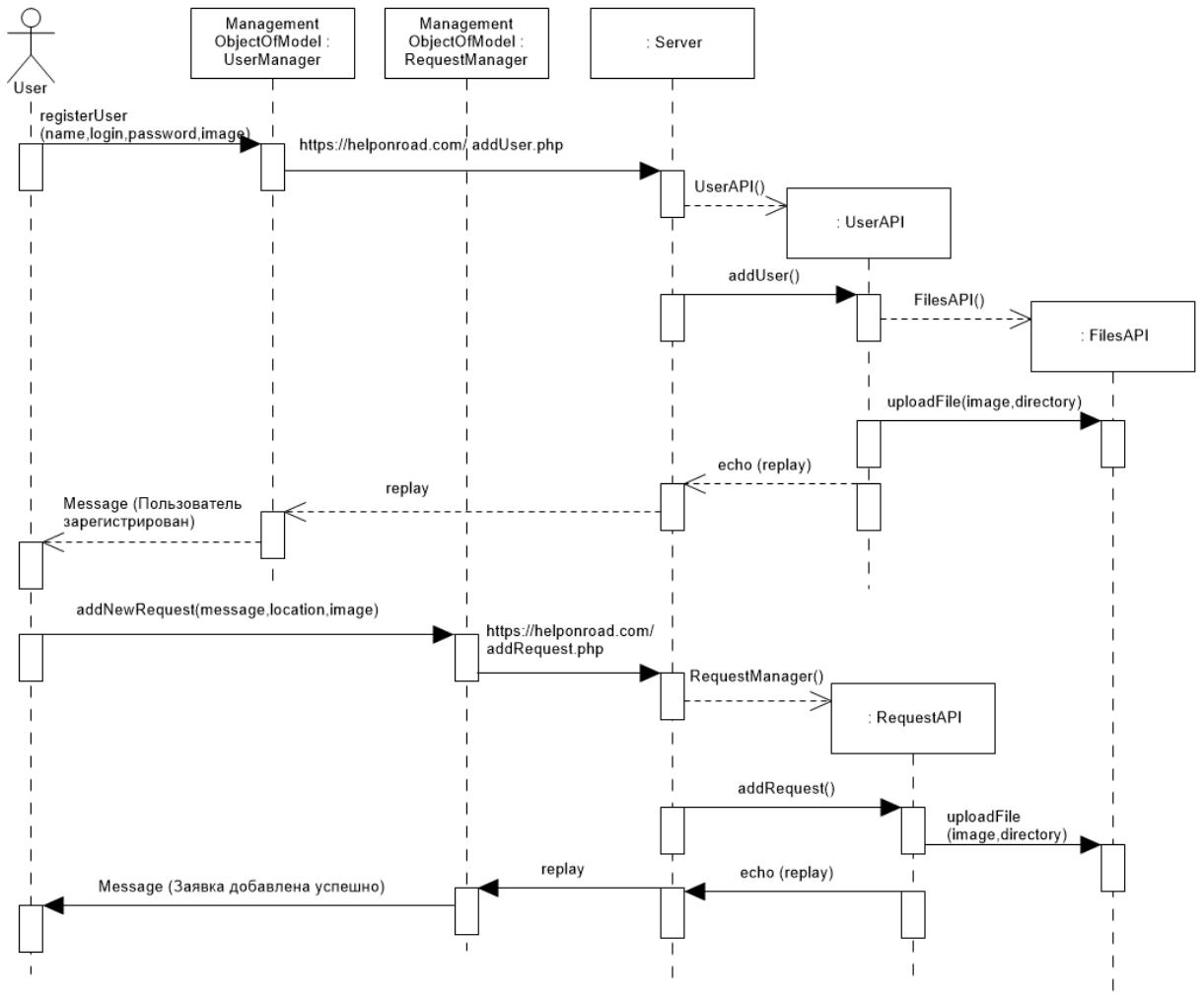


Рис. 5.49. Диаграмма последовательностей действий сценария «Регистрации пользователя и добавления новой заявки».

### 5.10.3 Детальное проектирование классов

На данном этапе проектирования выполнено уточнение методов проектируемых классов.

Класс «FilesAPI» содержит метод для сохранения файла в указанную директорию.

Класс RequestAPI содержит методы для добавления, получения списка и изменения статуса заявки.

Класс UserAPI содержит методы для регистрации, авторизации, изменения имени, пароля пользователя и добавления, удаления изображения пользователя.

Класс OfferAPI содержит методы для добавления и получения списка предложений помощи.

Детальное описание классов представлено в п. 2. прил. 4 «Спецификации на модули» для модуля «ServerClasses» компонента «ServerPartApp».

Детальная диаграмма классов проектируемого пакета представлена на рис. 5.50.

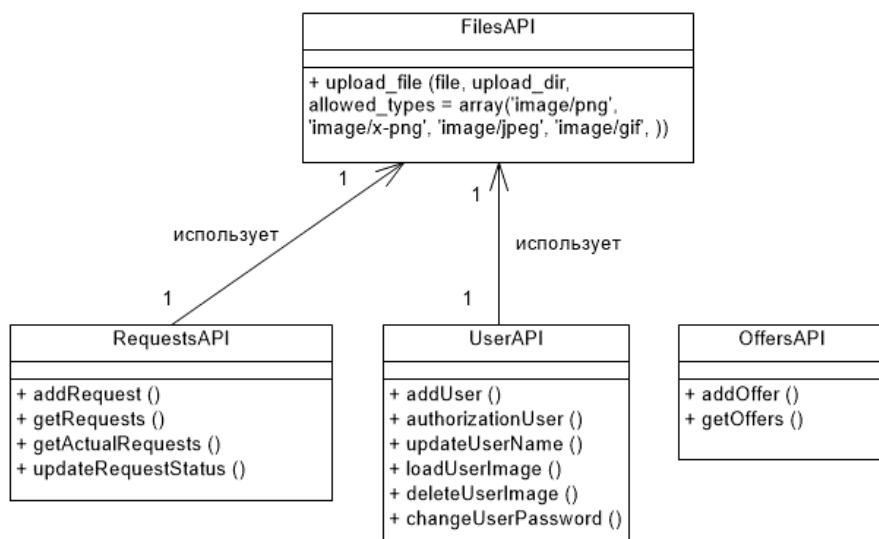


Рис. 5.50. Детальная диаграмма классов пакета «Интерфейс к базе данных MySQL»

### 5.11 Проектирование пакета «База данных MySQL»

Указанный пакет содержит базу данных, которая размещается на удаленном сервере под управлением СУБД MySQL.

На основе логического проектирования базы данных серверной части приложения (см. п. 4.4.1) выполнено ее физическое проектирование для СУБД MySQL. В табл. 5.18 - 5.21 представлены таблицы базы данных с указанием имен полей, имен атрибутов, типов данных и предлагаемых ограничений целостности. Название таблиц приводится на английском языке, с указанием в скобках названия на русском языке. ER-диаграмма базы данных MySQL серверной части приложения представлена на рис. 5.51.

Таблица 5.18

## Users (Пользователи)

Имя атрибута	Имя поля	Тип данных	Ограничения
Идентификатор пользователя	id	INTEGER(11)	Первичный ключ, значение уникально
Логин	login	VARCHAR(50)	Значение уникально
Пароль	password	VARCHAR(50)	Значение не должно быть пустым
Имя	name	VARCHAR(30)	Значение не должно быть пустым
Фото	image	TEXT	По умолчанию пустое значение
Каталог	directory	TEXT	Значение не должно быть пустым
Дата добавления	create_date	TIMESTAMP	Значение не должно быть пустым, по умолчанию CURRENT_TIMESTAMP – текущая дата и время
Дата изменения	update_date	TIMESTAMP	По умолчанию пустое значение

Таблица 5.19

## Requests (Заявки)

Имя атрибута	Имя поля	Тип данных	Ограничения
Идентификатор заявки	id	INTEGER(11)	Первичный ключ, значение уникально
Сообщение	message	VARCHAR(255)	Значение не должно быть пустым
Широта	latitude	DOUBLE	Значение не должно быть пустым
Долгота	longitude	DOUBLE	Значение не должно быть пустым
Дата добавления	create_date	TIMESTAMP	Значение не должно быть пустым, по умолчанию CURRENT_TIMESTAMP – текущая дата и время
Дата изменения	update_date	TIMESTAMP	По умолчанию пустое значение
Идентификатор пользователя	id_user	INTEGER(11)	Внешний ключ. Значение не должно быть пустым
Идентификатор статуса	id_status	INTEGER(11)	Внешний ключ. Значение не должно быть пустым. По умолчанию равно 0

Таблица 5.20

## Offers (Предложения)

Имя атрибута	Имя поля	Тип данных	Ограничения
1	2	3	4
Идентификатор предложения	id	INTEGER(11)	Первичный ключ, значение уникально
Сообщение	message	VARCHAR(255)	Значение не должно быть пустым
Дата добавления	create_date	TIMESTAMP	Значение не должно быть пустым, по умолчанию CURRENT_TIMESTAMP – текущая дата и время
Дата изменения	update_date	TIMESTAMP	По умолчанию пустое значение

Продолжение табл. 5.20

1	2	3	4
Идентификатор пользователя	id_user	INTEGER(11)	Внешний ключ. Значение не должно быть пустым
Идентификатор заявки	id_request	INTEGER(11)	Внешний ключ. Значение не должно быть пустым

Таблица 5.21

## Statuses (Статусы)

Имя атрибута	Имя поля	Тип данных	Ограничения
Идентификатор статуса	id_status	INTEGER(11)	Первичный ключ, значение уникально
Значение	value	VARCHAR(100)	Значение не должно быть пустым

## 5.12 Проектирование пакета «База данных SQLite»

Указанный пакет содержит базу данных SQLite, которая создается при первом запуске клиентской части приложения на мобильном устройстве.

На основе логического проектирования базы данных клиентской части приложения (см. п. 4.4.2) выполнено ее физическое проектирование.

В табл. 5.22 - 5.24 представлены таблицы базы данных с указанием имен полей, имен атрибутов, типов данных и предлагаемых ограничений целостности. Название таблиц приводится на английском языке, с указанием в скобках названия на русском языке.

Таблица 5.22

## Users (Пользователи)

Имя атрибута	Имя поля	Тип данных	Ограничения
Идентификатор пользователя	id	INTEGER	Первичный ключ, значение уникально
Имя	name	TEXT	Значение не должно быть пустым
Наличие изображения	image	BOOLEAN	Значение не должно быть пустым. По умолчанию равно 0
Дата добавления	create_date	TEXT	Значение не должно быть пустым
Дата изменения	update_date	TEXT	По умолчанию пустое значение
Дата загрузки	upload_date	TEXT	Значение не должно быть пустым

Таблица 5.23

## Requests (Заявки)

Имя атрибута	Имя поля	Тип данных	Ограничения
Идентификатор заявки	id	INTEGER	Первичный ключ, значение уникально
Сообщение	message	TEXT	Значение не должно быть пустым
Широта	latitude	REAL	Значение не должно быть пустым
Долгота	longitude	REAL	Значение не должно быть пустым
Наличие изображения	is_image	BOOLEAN	Значение не должно быть пустым. По умолчанию равно 0
Является новой	is_new	BOOLEAN	Значение не должно быть пустым. По умолчанию равно 1
Получено уведомление	is_get_notify	BOOLEAN	Значение не должно быть пустым. По умолчанию равно 0
Статус	status	INTEGER	Значение не должно быть пустым. Принимает значения: 0,1,2. По умолчанию равно 0
Дата добавления	create_date	TEXT	Значение не должно быть пустым
Дата изменения	update_date	TEXT	По умолчанию пустое значение
Дата загрузки	upload_date	TEXT	Значение не должно быть пустым
Идентификатор пользователя	id_user	INTEGER	Внешний ключ. Значение не должно быть пустым

Таблица 5.24

## Offers (Предложения)

Имя атрибута	Имя поля	Тип данных	Ограничения
Идентификатор предложения	id	INTEGER	Первичный ключ, значение уникально
Сообщение	message	TEXT	Значение не должно быть пустым
Является новым	is_new	BOOLEAN	Значение не должно быть пустым. По умолчанию равно 1
Получено уведомление	is_get_notify	BOOLEAN	Значение не должно быть пустым. По умолчанию равно 0
Дата добавления	create_date	TEXT	Значение не должно быть пустым
Дата изменения	update_date	TEXT	По умолчанию пустое значение
Дата загрузки		TEXT	Значение не должно быть пустым
Идентификатор пользователя	id_user	INTEGER	Внешний ключ. Значение не должно быть пустым
Идентификатор заявки	id_request	INTEGER	Внешний ключ. Значение не должно быть пустым

ER-диаграмма базы данных SQLite клиентской части приложения представлена на рис. 5.52.

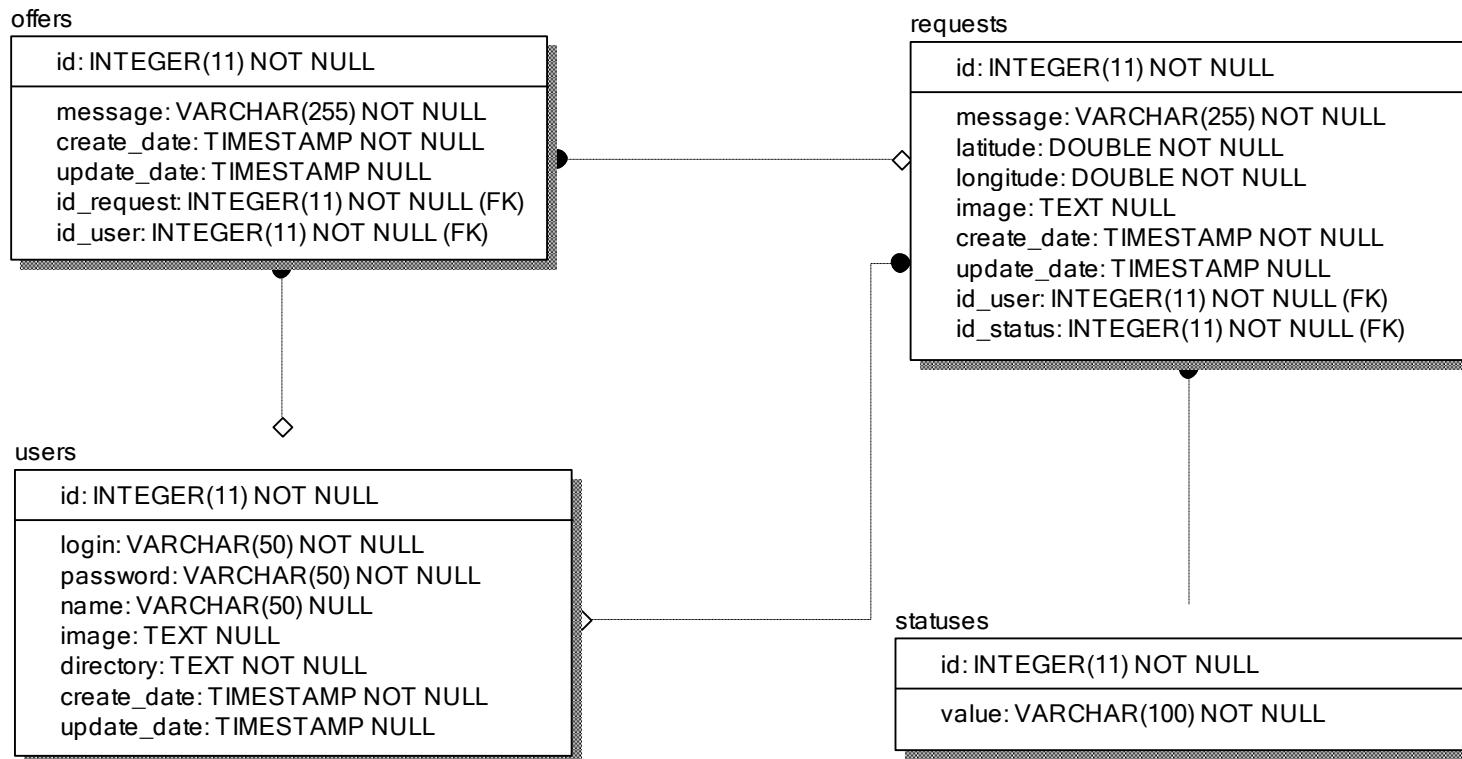


Рис. 5.51. ER-диаграмма базы данных серверной части приложения

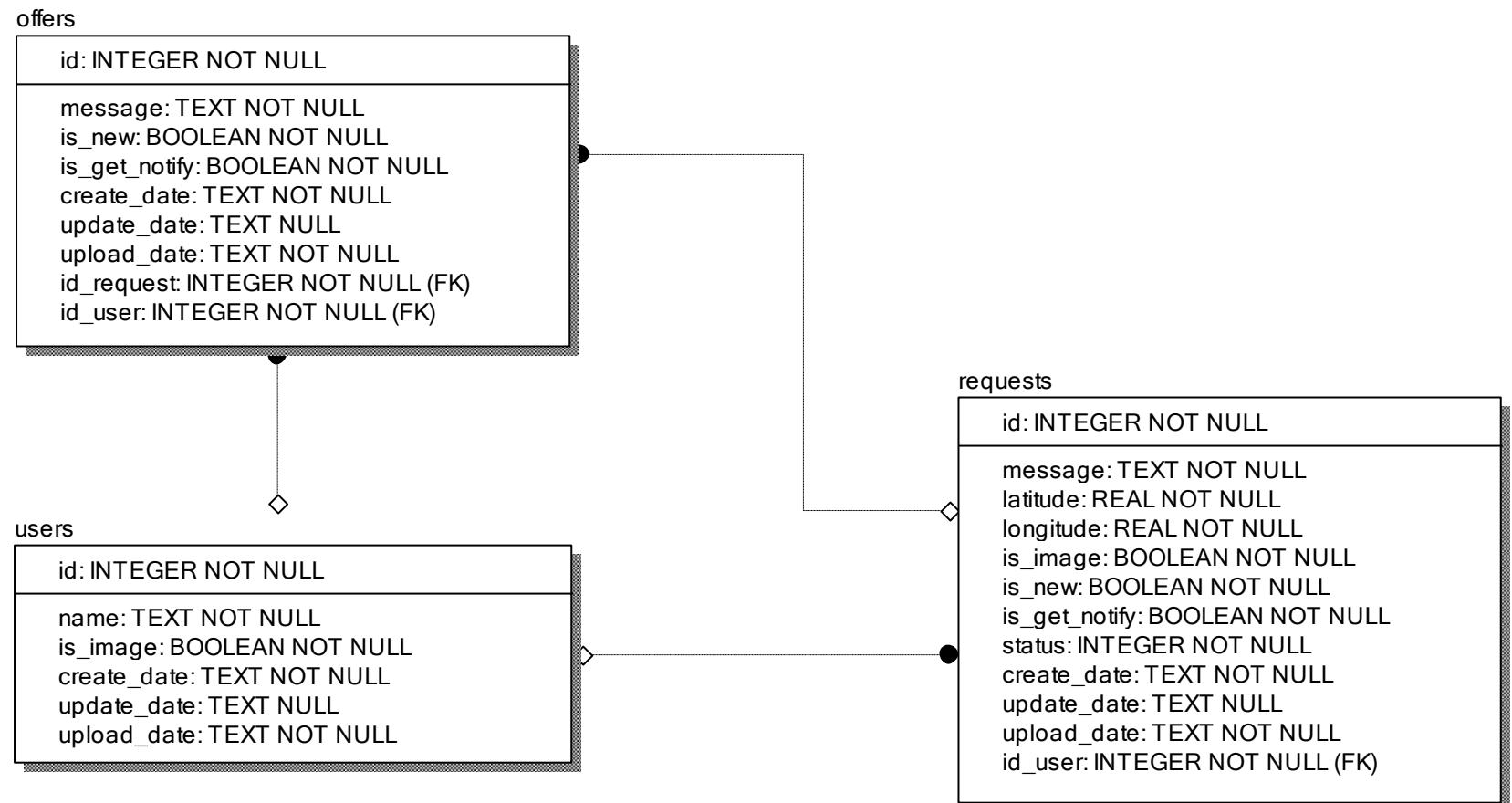


Рис. 5.52. ER-диаграмма базы данных клиентской части приложения

### 5.13 Построение диаграмм компонентов

Диаграммы компонентов показывают, как выглядит программное обеспечение на физическом уровне, т. е. из каких частей оно состоит и как эти части связаны между собой [5].

Диаграмма компонентов разрабатываемого ПО представлена на рис. 5.53.

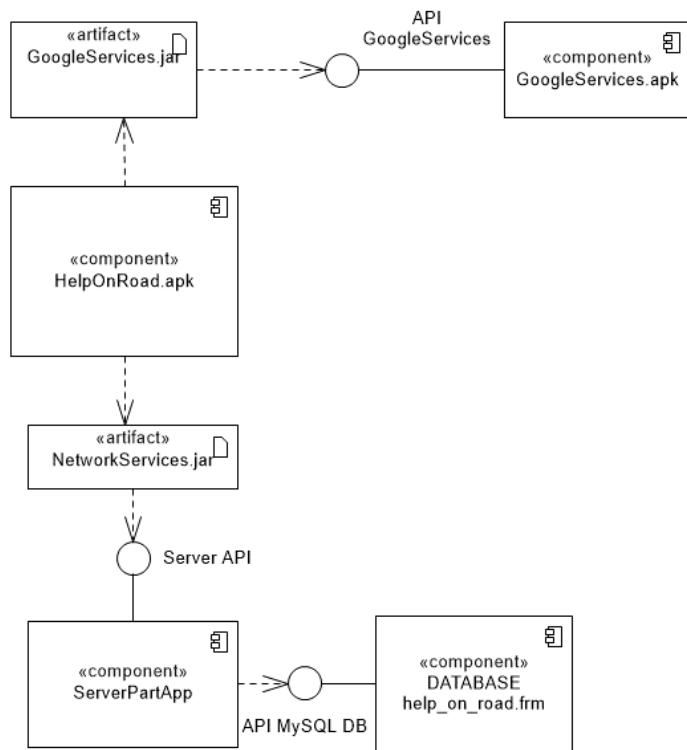


Рис. 5.53. Диаграмма компонентов ПО для мобильных устройств «Помощь на дороге»

Спецификации на компоненты представлены в таблице 5.25.

Таблица 5.25

Спецификации на компоненты

Название	Описание	Назначение	Входные/выходные данные
1	2	3	4
HelpOnRoad.apk	Исполняемый файл мобильного приложения, выполняющийся на устройстве под управлением ОС	Выполнение клиентских функций приложения: отображение списка заявок на помощь, добавление новой	Входные данные: данные о заявке на помощь, данные сообщения о готовности оказать помощь. Выходные данные: список заявок на оказание помощи,

Продолжение табл. 5.25

1	2	3	4
	Android 4.1. (Jelly Bean) и выше	заявки в систему, добавление сообщений-предложений помощи, отображение списка заявок на карте Google, добавление данных о пользователе устройства	данные о пользователе устройства
help_on_road.frm	Файл базы данных «Помощь на дороге» приложения СУБД MySQL	Хранение, добавление, удаление, обновление данных заявок на помощь, предложений помощи, пользователей - на удаленном сервере	Данные о заявках на помощь: имя, логин, пароль, фото водителя, местоположение, сообщения о готовности оказать помощь
ServerPartApp	Серверная часть приложения - набор скриптов на языке PHP для взаимодействия с БД MySQL help_on_road	Выполнение SQL-запросов к базе данных «Помощь на дороге» приложения СУБД MySQL, обработка поступающей информации о заявках водителей на помощь	Входные данные: сведения о заявках помощи в виде POST-параметров запроса. Выходные данные: данные о заявках в формате JSON
ServerAPI	Интерфейс для взаимодействия с размещенной на сервере базой данных MySQL «Помощь на дороге» (скрипты на языке PHP)	Выполнение SQL-запросов к базе данных «Помощь на дороге» приложения СУБД MySQL, обработка поступающей информации о заявках водителей на помощь	Входные данные: сведения о заявках на помощь в виде POST-параметров запроса. Выходные данные: данные о заявках в формате JSON
GoogleServices.apk	Приложение GooglePlayServices	Предоставляет: Google Maps API для работы с картами в приложениях, Fused Location API для работы с местоположением устройства, GCM Task Service для работы с уведомлениями	Входные данные: местоположение пользователя устройства, список заявок Выходные данные: карта Google земной поверхности, данные о маршрутах, отметки на карте, уведомления
GoogleServices.jar	Библиотека мобильного приложения для работы приложением GoogleServices	Предоставляет классы для работы с сервисами Google для определения местоположения, получения карты Google, получения уведомлений о новых заявках	Входные данные: местоположение пользователя устройства. Выходные данные: карта Google земной поверхности, данные о маршрутах, отметки на карте
Network.jar	Библиотека для выполнения сетевых операций по протоколу HTTP (HTTPS)	Отправляет HTTP-запросы серверу	Входные данные: данные запроса: url, GET, POST-параметры, передаваемые данные (тело запроса). Выходные данные: массив байтов

Компонент ServerPartApp состоит из набора модулей-скриптов на языке PHP, спецификации на которые представлены в п. 2. прил. 4. Текст программы модулей - в прил. 2. Модульная схема компонента ServerPartApp - на рис. 5.54.

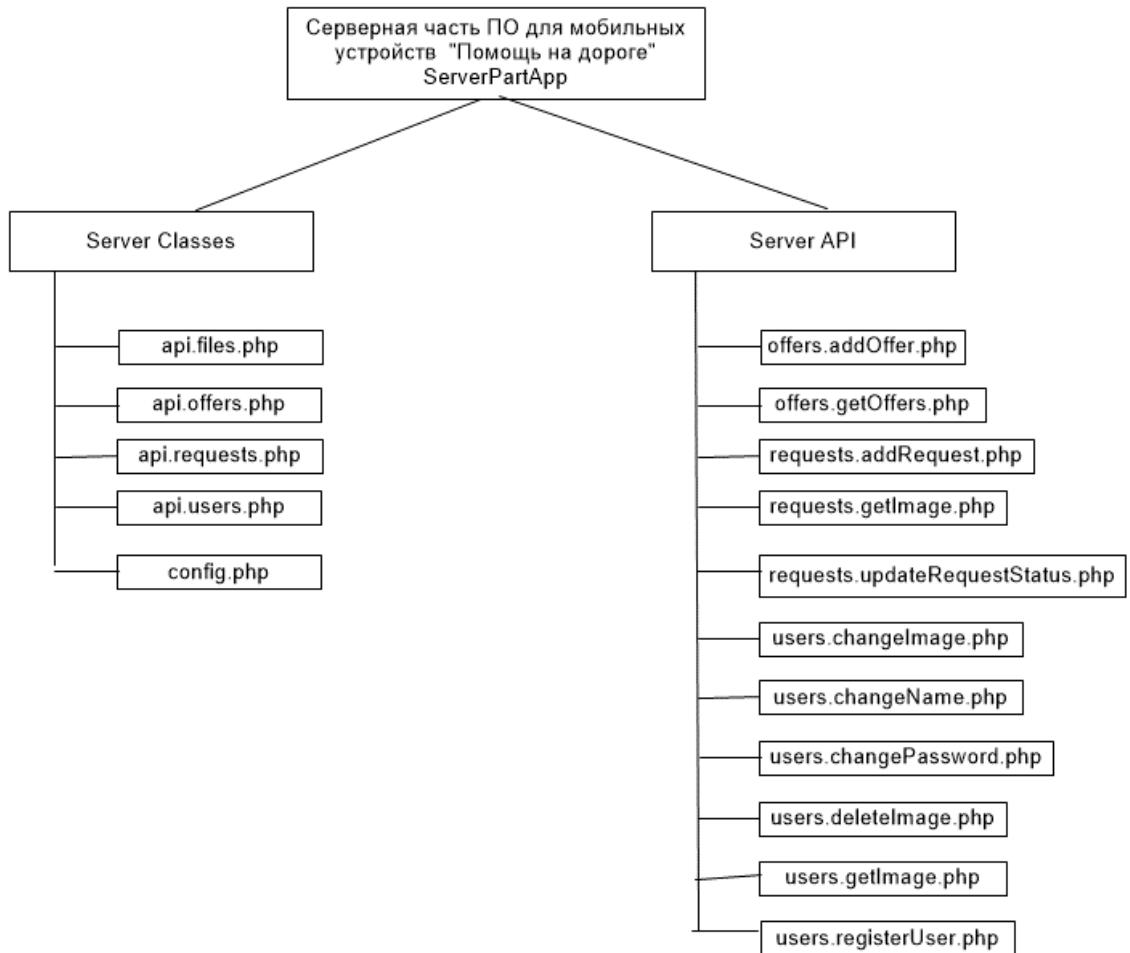


Рис. 5.54. Модульная схема компонента «ServerPartApp»

Компонент HelpOnRoad.apk является исполняемым пакетным файлом и состоит из компонуемых модулей, спецификации на которые представлены в п. 1 прил. 4. Текст программы модулей - в прил. 2. Модульная схема компонента HelpOnRoad.apk - на рис. 5.55.

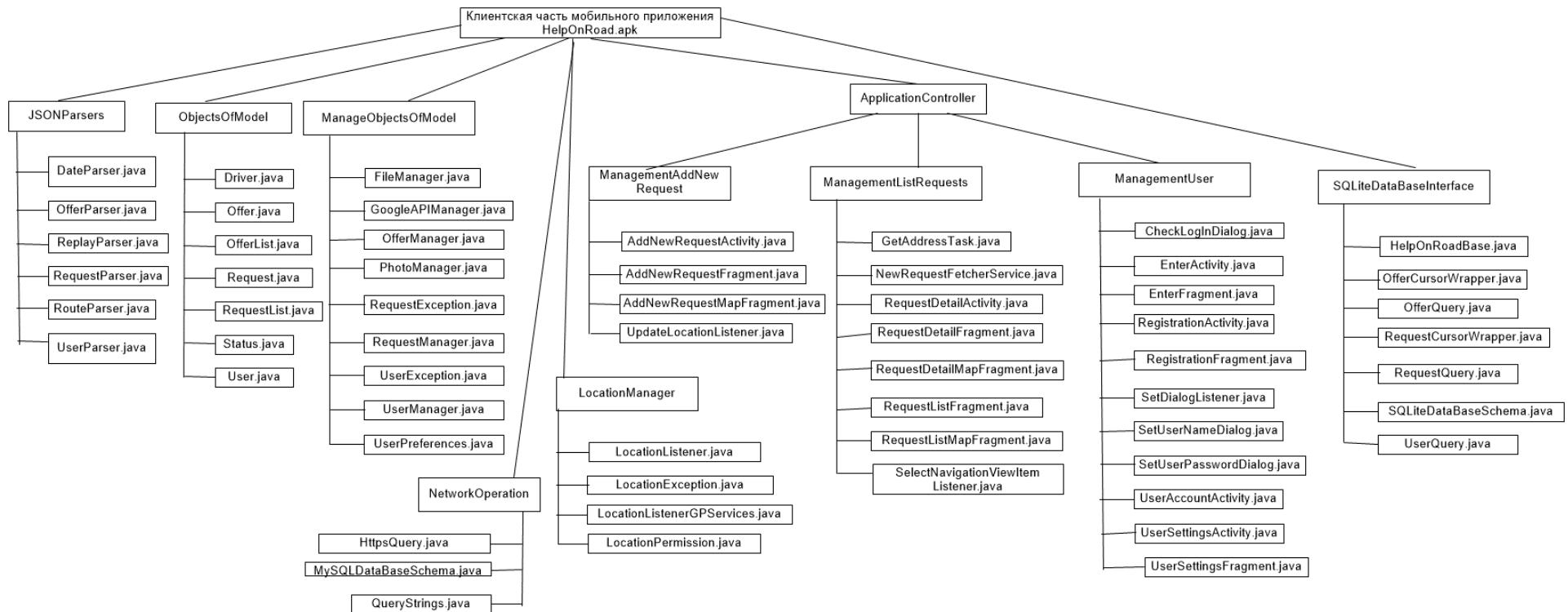


Рис. 5.55. Модульная схема компонента «HelpOnRoad.apk»

### 5.14 Построение диаграмм размещения

Диаграмма размещения отражает физические взаимосвязи между программными и аппаратными компонентами проектируемой системы [5].

Диаграмма размещения разрабатываемой системы представлена на рис. 5.56.

Глобальная сеть интернет связывает сервер под управлением web-сервера Apache, на котором размещается база данных приложения (help\_on\_road.frm), и мобильные устройства пользователей, на которых установлено и запущено мобильное приложение HelpOnRoad.apk. Взаимодействие мобильного приложения и сервера осуществляется средствами сетевого протокола прикладного уровня HTTPS.

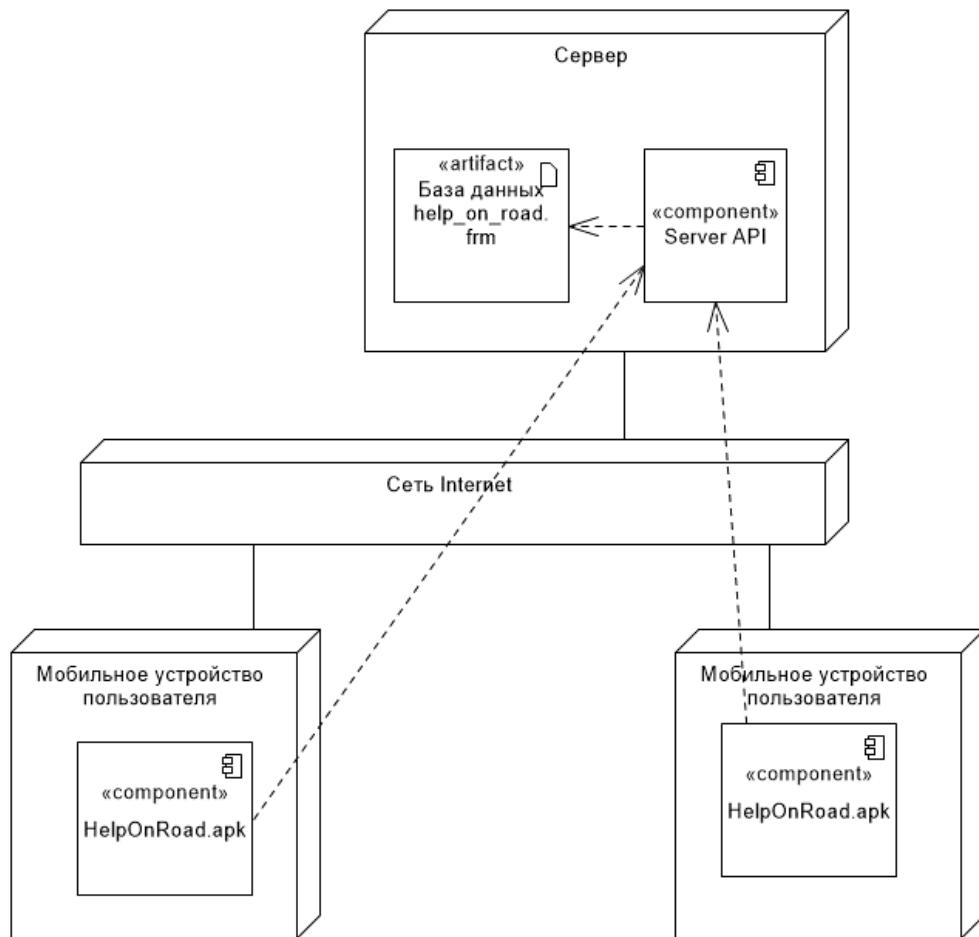


Рис. 5.56. Диаграмма размещения компонентов ПО для мобильных устройств «Помощь на дороге»

## 6 Проектирование интерфейса пользователя

### 6.1 Построение графа диалога

Граф диалога - ориентированный взвешенный граф, каждой вершине которого соответствует определенное состояние диалога, характеризующееся набором доступных пользователю действий. Дуги, исходящие из вершин, показывают возможные изменения состояний при выполнении пользователем указанных действий [5].

Интерфейс клиентской части мобильного приложения «Помощь на дороге» содержит диалоговые окна, которые соответствуют вершинам графа диалога (рис. 6.1), описание которых представлено в табл. 6.1.

Таблица 6.1

#### Вершины графа диалога

Вершина графа	Описание
Заявки в системе: список	Вкладка окна «Список заявок» для просмотра сведений об актуальных заявках на помощь. Содержит список заявок
Заявки в системе: карта	Вкладка окна для просмотра сведений об актуальных заявках на помощь. Содержит карту Google с маркерами заявок
Заявка: описание	Вкладка окна «Заявка» для просмотра сведений детальной заявки, которая содержит описание заявки и список предложений помощи для заявки
Заявка: маршрут	Вкладка окна «Заявка» для просмотра сведений детальной заявки, которая содержит карту с маршрутом между местоположением пользователя и заявки
Создание заявки: описание проблемы	Вкладка окна «Создание заявки» для добавления заявки на помощь, которая обеспечивает ввод сообщения заявки и выбор фотографии неисправности
Создание заявки: местоположение	Вкладка окна «Создание заявки» для добавления новой заявки, которая предназначена для выбора местоположения заявки на карте Google
Вход	Окно для авторизации пользователя в приложении
Регистрация	Окно для регистрации нового пользователя в приложении
Главное меню	Окно для навигации в приложении
Настройки	Окно для редактирования настроек получения уведомлений
Системные настройки приложения	Окно стандартного приложения системы Android для задания настроек приложения «Помощь на дороге»
Учетная запись	Окно для работы с учетной записью авторизованного пользователя приложения
Изменение имени пользователя	Диалоговое окно для изменения имени пользователя
Изменение пароля пользователя	Диалоговое окно для изменения пароля пользователя
Камера	Окно стандартного приложения системы Android для работы с камерой
Галерея	Окно галереи устройства

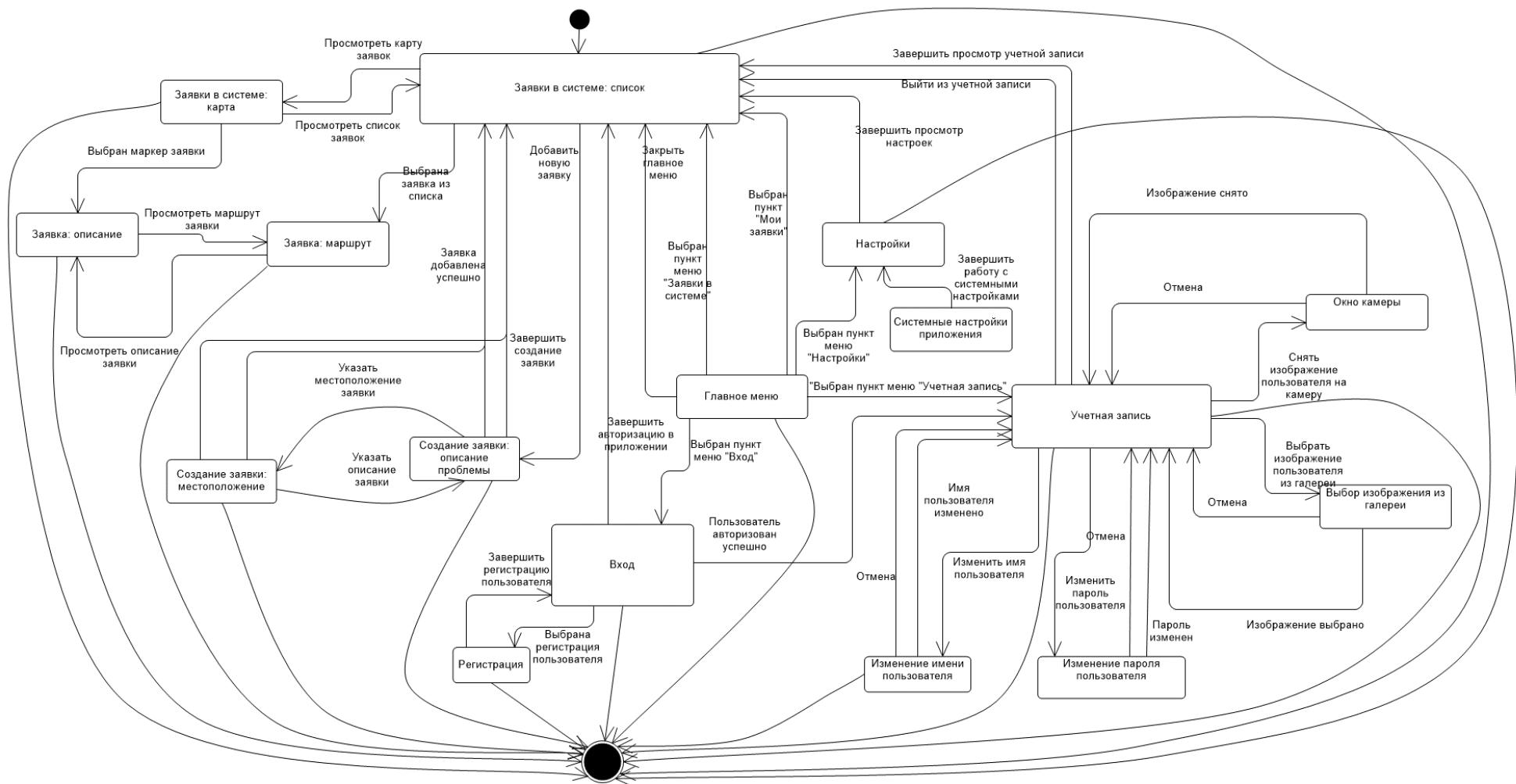


Рис.6.1. Граф диалога мобильного приложения «Помощь на дороге»

## 6.2 Разработка форм ввода-вывода информации

Интерфейс мобильного приложения является интерфейсом со свободной навигацией, где осуществляется визуальная обратная часть с пользователем и возможность прямого манипулирования информацией на экране [5].

На рис. 6.2 представлено главное окно «Заявки в системе», которое содержит вкладки: «Список» - для отображения списка заявок на помощь, «Карта» - для отображения маркеров заявок (рис. 6.2).

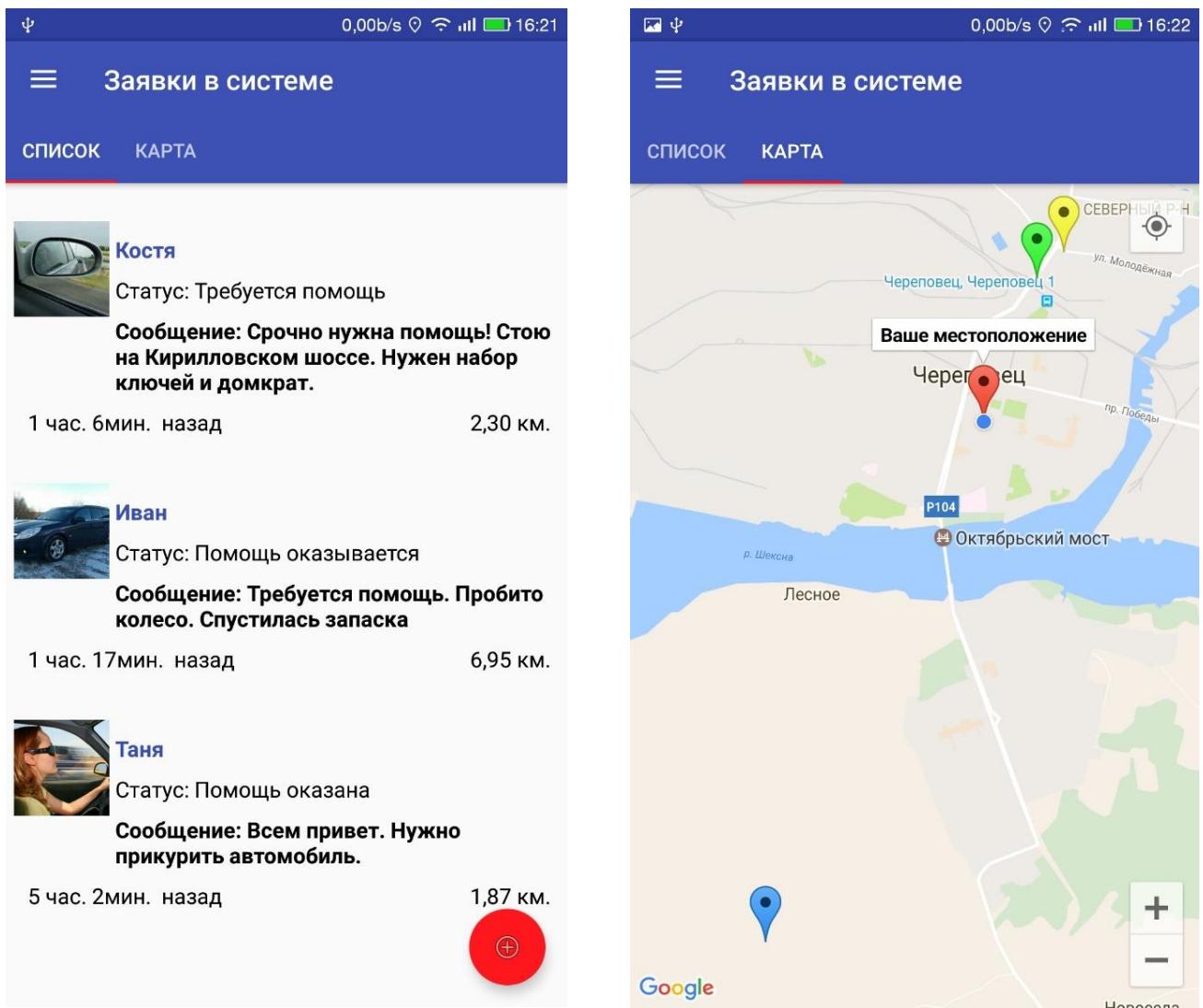


Рис. 6.2. Окно «Заявки в системе»

При нажатии на кнопку «+» окна «Заявки в системе» осуществляется переход к окну «Создание заявки» (рис. 6.3), которое содержит вкладки: «Описание проблемы» и «Местоположение» - для добавления новой заявки на помощь.

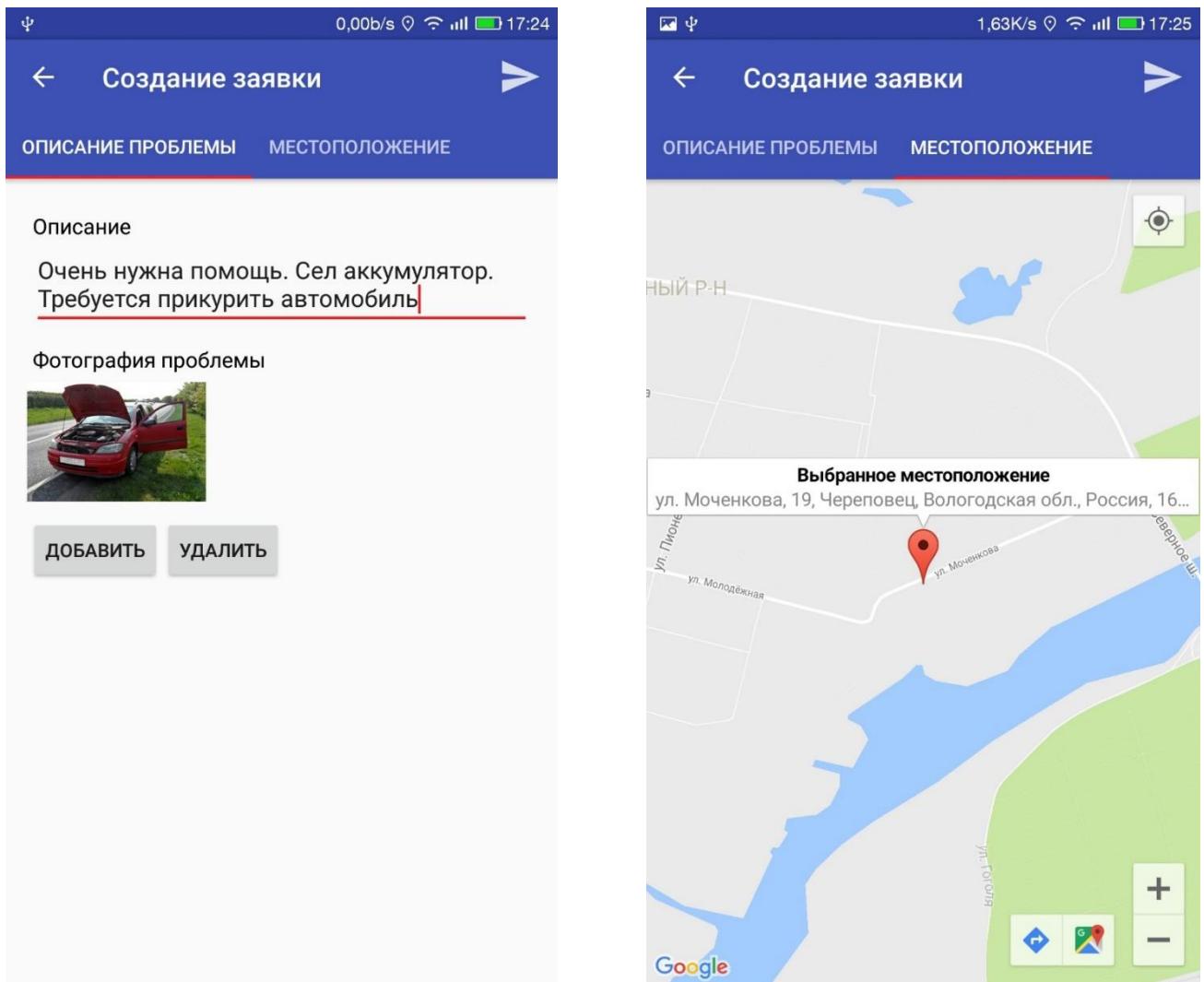


Рис. 6.3. Окно «Создание заявки»

В окне «Заявки в системе» (рис. 6.2) пользователь может выбрать: заявку из списка заявок или маркер заявки на карте - для просмотра деталей заявки в окне «Заявка» (рис. 6.4), содержащее вкладки «Описание» и «Маршрут».

Вкладка «Описание» содержит описание заявки и список сообщений - переписку пользователей с водителем, которому требуется помощь.

Вкладка «Маршрут» содержит карту, на которой отображается маршрут до выбранной заявки.

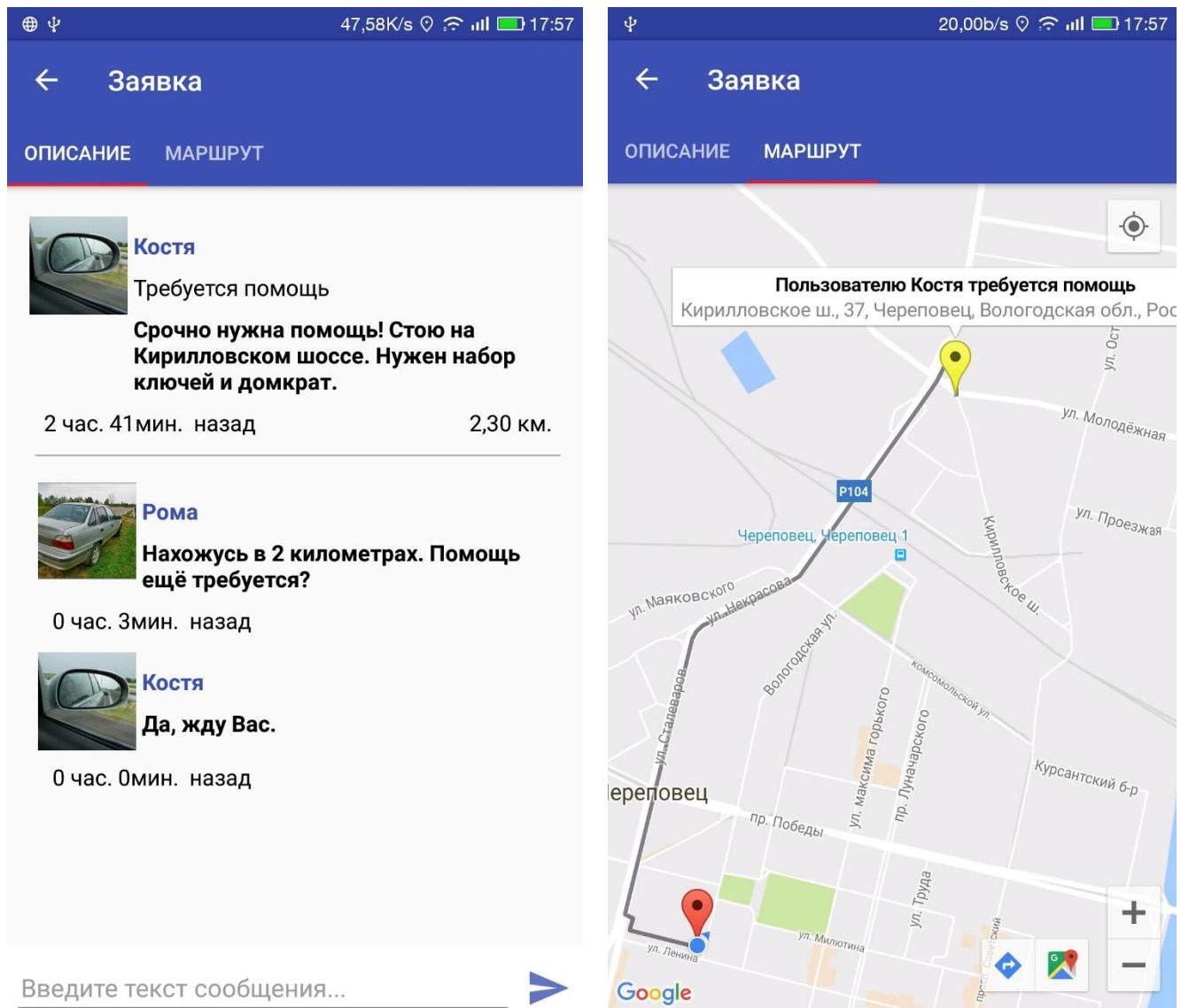


Рис. 6.3. Окно «Заявка»

При левом свайпе в окне «Заявки в системе» (рис. 6.2) осуществляется переход в главное меню приложения. На рис. 6.4 показаны пункты главного меню для неавторизованного пользователя приложения (левая часть рисунка) и для авторизованного пользователя (правая часть рисунка).

При выборе в главном меню пункта «Войти» осуществляется переход в окно «Войти» для авторизации в приложении (левая часть рис. 6.5), в котором незарегистрированные пользователи нажатием на кнопку «Зарегистрироваться» переходят в окно «Регистрация» для регистрации в приложении (правая часть рис. 6.5).

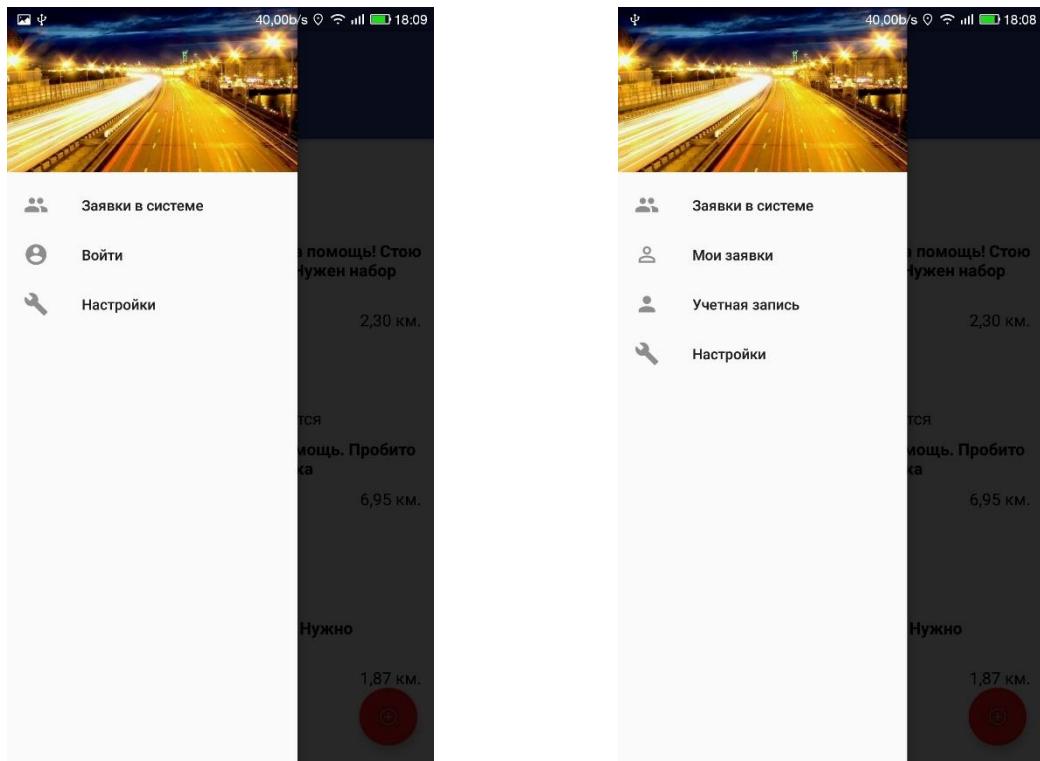


Рис. 6.4. Главное меню для неавторизованного и авторизованного пользователя

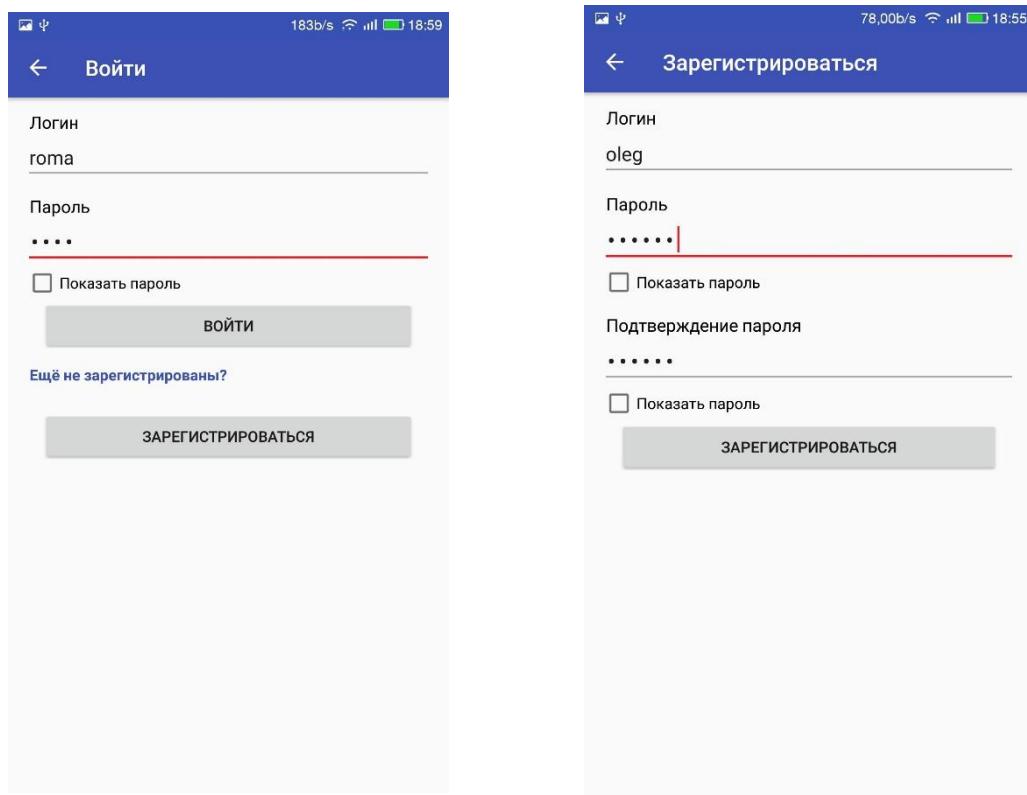


Рис. 6.5. Окна «Войти» и «Зарегистрироваться»

При выборе в главном меню (рис. 6.4) пункта «Настройки» осуществляется переход в окно для просмотра настроек приложения (рис. 6.6, окно «Настройки»).

При выборе в главном меню для авторизованного пользователя (правая часть рис. 6.4) пункта «Учетная запись» осуществляется переход в окно «Учетная запись» (рис. 6.6, окно «Учетная запись») для просмотра сведений аккаунта пользователя приложения.

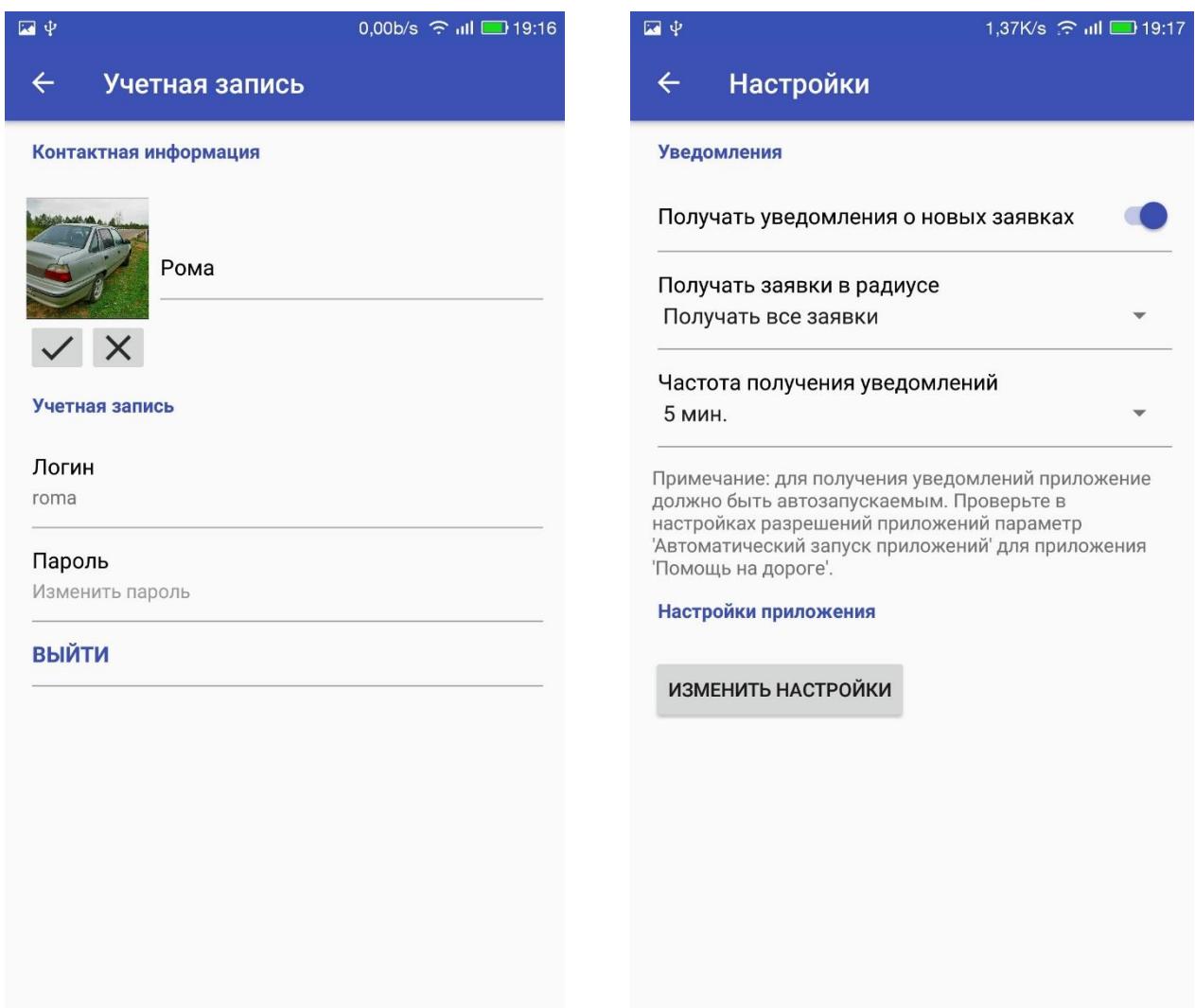


Рис. 6.6. Окна «Учетная запись» и «Настройки»

Подробное описание окон и последовательностей действий по работе с ними представлено в руководстве пользователя (прил. 5).

## 7 Выбор стратегии тестирования, разработка тестов, программа и методика испытаний

### 7.1 Объект испытаний

Объектом тестирования является программное обеспечение для мобильных устройств «Помощь на дороге» под управлением операционной системы Android.

### 7.2 Цель испытаний

Тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий две различные цели [2, 22]:

- продемонстрировать заказчику, что программное обеспечение соответствует требованиям, которые заявлены в техническом задании;
- выявить ситуации, в которых поведение программы является неправильным, нежелательным или несоответствующим спецификации;
- произвести проверку работы модулей и всей программы на различных наборах входных данных.

### 7.3 Требования к информационному, аппаратно-программному обеспечению и документации

#### 7.3.1 Требования к функциональным характеристикам

На основании анализа требований к программному обеспечению составлено техническое задание на разработку (см. прил. 1) и разработаны спецификации (см. п. 4).

Программное обеспечение должно полностью удовлетворять функциональным и эксплуатационным требованиям, заявленным в техническом задании (см. п. 4.1, 4.3 прил. 1):

- обеспечить добавление пользователем новой заявки на помощь: ввод и редактирование данных заявки пользователем, отправку запроса на

- сохранение данных заявки в базу, размещенную на сервере, с получением текстового сообщения, подтверждающего сохранение заявки;
- регистрировать пользователя в системе: обеспечивать ввод пользователем логина и пароля с клавиатуры, отправку запроса серверной части на сохранение введенных данных в базу с получением текстового сообщения, подтверждающего регистрацию пользователя;
  - авторизовать пользователя в системе по логину и паролю;
  - отправлять запрос на сохранение данных предложения помощи в базу, размещенную на сервере (посредством сети интернет), с получением текстового сообщения, подтверждающего сохранение данных в базе;
  - загружать из базы и отображать список заявок на оказание помощи в диалоговом окне устройства;
  - загружать из базы и отображать местоположения заявок в виде отметок на карте Google;
  - отображать сведения, список сообщений выбранной пользователем заявки;
  - строить маршрут для выбранной пользователем заявки из списка;
  - обеспечить добавление пользователем сообщения к выбранной заявке;
  - получать уведомления о заявках в выбранном пользователем радиусе получения заявок;
  - отображать последние добавленные заявки авторизованного пользователя.

### 7.3.2 Требования к надежности

Программное обеспечение должно удовлетворять всем заявленным в техническом задании требованиям к надежности (см. п. 4.2 прил. 1).

В процессе работы программного обеспечения не допустима ситуация полного отказа и не допустима ситуация искажения обрабатываемой информации в процессе работы приложения.

### 7.3.3 Требования к аппаратному и информационному обеспечению

Программа должна функционировать на устройствах с минимальными аппаратными требованиями, заявленными в техническом задании на разработку (см. п. 4.4 прил. 1).

Программное обеспечение должно быть совместимо между версиями операционной системы Android. Минимальная версия системы - Android 4.1 (JellyBean). Установка на устройства с версией Android ниже минимальной недопустима.

### 7.3.4 Требования к программной документации

В состав программной документации входит (см. п. 5 прил. 1): расчетно-пояснительная записка с приложениями (Приложение 1 - Техническое задание, Приложение 2 – Схемы и диаграммы, Приложение 3 - Спецификация, Приложение 4 - Текст программы, Приложение 5 - Руководство пользователя).

## 7.4 Состав, порядок и методы испытаний

Программное обеспечение тестируется на устройствах с версией операционной системы и минимальными техническими требованиями, заявленными в техническом задании (см. п. 4.4 прил. 1).

- сенсорный экран, использующимся для ввода, вывода и отображения информации;
- поддержка Wi-Fi, 2G/3G/4G модуля;
- камера 3,2 Мпикс и выше;
- процессор 1ГГц и выше;
- память 512 Мб ROM/ 256 Мб RAM.

### 7.4.1 Состав испытаний

Порядок проведения испытаний:

- проверка запуска программы на тестируемом устройстве;
- тестирование заявленных в техническом задании функциональных требований на полноту и логику выполнения;
- тестирование интерфейса пользователя на удобство и понятность использования;
- тестирование надежности;
- выявление ошибок на каждом из этапов;
- отладка программного обеспечения;
- контрольное тестирование.

#### 7.4.2 Методы испытаний

Различают два подхода к формированию тестов: структурный и функциональный. Каждый из указанных подходов имеет свои особенности и области применения [5].

Структурный подход базируется на том, что известна структура тестируемого программного обеспечения, в том числе его алгоритмы («стеклянный ящик»). В этом случае тесты строят так, чтобы проверить правильность реализации заданной логики в коде программы.

Функциональный подход основывается на том, что структура программного обеспечения неизвестна («черный ящик»). В этом случае тесты строят, опираясь на функциональные спецификации. Этот подход называют также подходом, управляемым данными, так как при его использовании тесты строят на базе различных способов декомпозиции множества данных.

Наборы тестов, полученные в соответствии с методами этих подходов, обычно объединяют, обеспечивая всестороннее тестирование программного обеспечения.

Также на ранних стадиях разработки программного применяют методы ручного тестирования. Ручной контроль как правило позволяет обнаружить 30-70% ошибок. Исходные данные для таких проверок: техническое задание,

спецификации, диаграммы классов, диаграммы компонентов, схемы отдельных компонентов, а на более поздних этапах - алгоритмы и тексты программ [5].

Для тестирования клиентской части приложения выбран метод ручного тестирования, для тестирования правильности выполнения скриптов серверной части приложения выбран метод граничных значений функционального тестирования [5].

## 7.5 Результаты проведения испытаний

На основании указанных подходов функционального тестирования сформированы тестовые задания (прил. 6, табл. П6.1 - П6.13) для проверки правильности работы серверной части приложения. В табл. П6.14 прил. 6 «Тестирование модулей» представлены результаты тестирования модулей клиентской и серверной частей приложения.

В табл. П6.15 прил. 6 «Системное тестирование» представлены результаты тестирования соответствия программного обеспечения для мобильных устройств «Помощь на дороге» требованиям технического задания.

Все выявленные в процессе тестирования неточности, ошибки работы, несоответствия спецификациям и требованиям технического задания программного обеспечения для мобильных устройств «Помощь на дороге» устранены.

## ЗАКЛЮЧЕНИЕ

Результатом выпускной квалификационной работы является разработанное клиент-серверное программное обеспечение для мобильных устройств на платформе Android «Помощь на дороге» для жителей г. Череповца и Череповецкого района.

Разработанное мобильное приложение «Помощь на дороге» упрощает процесс отправки сообщений водителями, которым требуется помочь, другим участникам дорожного движения, обеспечивает обратную связь водителя с пользователями приложения, которые готовы оказать требуемую помощь.

В результате выполнения выпускной квалификационной работы все поставленные цели и задачи на работу выполнены полностью.

В ходе выполнения этапов выпускной квалификационной работы (анализ предметной области, анализ требований и выбор модели ЖЦ ПО, выбор технологий и инструментальных средств для разработки, разработка спецификаций, проектирование ПО, разработка и тестирования ПО, подготовка программной документации) освоены следующие профессиональные компетенции:

- Владение навыками использования различных технологий разработки программного обеспечения (ПК-3);
- Владение концепциями и атрибутами качества программного обеспечения (надежности, безопасности, удобства использования), в том числе, роли людей, процессов, методов, инструментов и технологий обеспечения качества (ПК-4);
- Владение классическими концепциями и моделями менеджмента в управлении проектами (ПК-6);
- Способность к формализации в своей предметной области с учетом ограничений используемых методов исследования (ПК-12);
- Готовность к использованию методов и инструментальных средств исследования объектов профессиональной деятельности (ПК-13)

- Готовность обосновать принимаемые проектные решения, осуществлять постановку и выполнение экспериментов по проверке их корректности и эффективности (ПК-14);
- Способность готовить презентации, оформлять научно-технические отчеты по результатам выполненной работы, публиковать результаты исследований в виде статей и докладов на научно-технических конференциях (ПК-15);
- Способность формализовать предметную область программного проекта и разработать спецификации для компонентов программного продукта (ПК-16);
- Способность выполнить начальную оценку степени трудности, рисков, затрат и сформировать рабочий график (ПК-17);
- Способность готовить коммерческие предложения с вариантами решения задания для выполнения ВКР (ПК-18).

## СПИСОК ЛИТЕРАТУРЫ

1. Вендроу, А.М. Проектирование программного обеспечения экономических информационных систем: учебник/А.М. Вендроу -2-е. изд. перераб. и доп. -М.: Финансы и статистика, 2006. -544 с.: ил.
2. Ершов, Е.В Методика и организация самостоятельной работы студентов [Текст]/ Е.В. Ершов и др.: учебно-методическое пособие – Череповец, ЧГУ, 2012 – 208 стр.
3. Карпова, Т.С. Базы данных: модели, разработка, реализация [Текст]/Т.С. Карпова - СПб.: Питер, 2002. - 304 с.: ил.
4. Лаврищева, Е.М. Software Engineering компьютерных систем. Парадигмы, технологии и CASE-средства программирования/Е.М. Лаврищева-К.: Наук. думка,2013. -383 с.
5. Иванова, Г.С. Технология программирования: учебник для вузов.-М.:Изво МГТУ им Н.Э. Баумана,2002. -320 с.:ил. (Серия Информатика в техническом университете).
6. Харди, Б Android. Программирование для профессионалов. 2-е изд. – СПб.: Питер, 2016. -240 стр.: ил. – (Серия «Для профессионалов）.

### Электронные ресурсы

7. Google Play. Road Helper (помощь на дороге) [Электронный ресурс]: интернет – ресурс – Режим доступа: <https://play.google.com/store/apps/details?id=by.shophunter.roadhunterapp&hl=ru> - (Дата обращения: 05.05.17).
8. PHP, Apache и MySQL. Как происходит процесс их взаимодействия? // [Электронный ресурс]: интернет- ресурс - Режим доступа: <https://coder-booster.ru/learning/php-beginners/php-apache-mysql-relation> – (Дата обращения: 05.05.17).
9. PHP [Электронный ресурс]: // интернет-ресурс – Режим доступа: <http://php.net> - (Дата обращения: 05.05.17).

10. Software Ideas Modeller [Электронный ресурс]: интернет-ресурс – Режим доступа: <https://www.softwareideas.net/> - (Дата обращения: 05.05.17).
11. Академия Microsoft: Методы и средства инженерии программного обеспечения [Электронный ресурс]: // ИНТУИТ. Национальный открытый университет. – Режим доступа: [http://www.intuit.ru/studies/courses/2190/237/lecture/6120?page=2\\_](http://www.intuit.ru/studies/courses/2190/237/lecture/6120?page=2_) - (Дата обращения: 05.05.2017).
12. Википедия. Свободная энциклопедия [Электронный ресурс]: интернет-ресурс – Режим доступа: <https://ru.wikipedia.org/wiki/> - (Дата обращения: 05.05.2017).
13. Вычисление расстояния и начального азимута между двумя точками на сфере // GIS LAB. Географические информационные системы и дистанционное зондирование. [Электронный ресурс]: интернет-ресурс - Режим доступа: <http://gis-lab.info/qa/great-circles.html> – (Дата обращения: 05.05.2017).
14. ДТП Череповца [Электронный ресурс]: //Социальная сеть «ВКонтакте» - Режим доступа: <https://vk.com/dtp.cherepovets> - (Дата обращения: 05.05.17).
15. Кузнецов, С.Д. Основы современных баз данных [Электронный ресурс]: интернет-ресурс - Режим доступа: [http://citforum.ru/database/osbd/ contents.shtml](http://citforum.ru/database/osbd/contents.shtml) - (Дата обращения: 05.05.2017).
16. Машина сломалась в дороге. Что делать? [Электронный ресурс]: интернет-ресурс – Режим доступа: [https://bycars.ru/journal/polomka-v-doroge\\_-chto-delat\\_293](https://bycars.ru/journal/polomka-v-doroge_-chto-delat_293) - (Дата обращения: 05.05.2017).
17. Нотация и семантика языка UML: Информация. [Электронный ресурс]: // ИНТУИТ. Национальный открытый университет. – Режим доступа: <http://www.intuit.ru/studies/courses/32/32/info> - (дата обращения: 06.05.2017).
18. Основы разработки программного обеспечения на примере языка С [Электронный ресурс]: // ИНТУИТ. Национальный открытый университет.

- Режим доступа: <http://www.intuit.ru/studies/courses/11876/1156/lecture/18252?page=1> - (Дата обращения: 06.05.2017).
- 19.Проектирование информационных систем. [Электронный ресурс]: // ИНТУИТ. Национальный открытый университет. – Режим доступа: <http://www.intuit.ru/studies/courses/2195/55/lecture/1620?page=1> - (дата обращения: 06.05.2017).
- 20.Разработка приложений для смартфонов на ОС Android. [Электронный ресурс]: // ИНТУИТ. Национальный открытый университет. – Режим доступа: <http://www.intuit.ru/studies/courses/12786/1219/info> - (Дата обращения: 06.05.2017).

## ГОСТЫ

- 21.ISO 12207:1995. (ГОСТ Р-1999). ИТ. Процессы жизненного цикла программных средств.
- 22.ГОСТ 34.603-92. ИТ. Виды испытаний автоматизированных систем.

ПРИЛОЖЕНИЕ 1

МИНОБРНАУКИ РОССИИ  
ФГБОУ ВО «Череповецкий государственный университет»

Институт информационных технологий  
Кафедра математического и программного обеспечения ЭВМ

УТВЕРЖДАЮ  
Зав. кафедрой МПО ЭВМ,  
д.т.н., профессор \_\_\_\_\_ Ершов Е.В.  
«\_\_\_» \_\_\_\_\_ 2017 г.

Разработка программного обеспечения для мобильных устройств  
«Помощь на дороге»

Техническое задание на выпускную квалификационную работу

Листов 11

Руководитель  
Гордеев Сергей Васильевич

Исполнитель  
студент гр. 1ПИб-01-41оп  
Кравинский Олег Алексеевич

2017 г.

## 1 Введение

В настоящее время у водителей часто возникает необходимость в быстром устранении внезапно возникшей в процессе дорожного движения поломки транспортного средства.

Во многих случаях другие участники дорожного движения в данной ситуации могут оказать посильную помощь водителю, у которого возникли указанные проблемы.

Основными средствами передачи сообщений о техпомощи в подобной ситуации выступают, в виду их распространенности на современном этапе развития информационных технологий, имеющиеся у большинства участников дорожного движения мобильные устройства (смартфоны и планшеты).

Возникает необходимость в разработке программного обеспечения для мобильных устройств (мобильного приложения), позволяющего водителю при поломке дорожно-транспортного средства попросить техпомощь у других участников дорожного движения.

## 2 Основания для разработки

Основанием для разработки служит задание на выпускную квалификационную работу, которое предложено для выполнения во время прохождения производственной практики предприятием – базой практики МБУ «Центр муниципальных информационных ресурсов и технологий» и согласовано с кафедрой МПО ЭВМ.

## 3 Назначение для разработки

Основное назначение данного программного обеспечения – обмен сообщениями между водителями, которые нуждаются в помощи при внезапной неисправности (поломке) транспортного средства, возникшей в процессе дорожного движения, и участниками дорожного движения, которые готовы

предоставить эту помощь, посредством мобильных устройств на платформе Android.

## 4 Требования к программе

### 4.1 Требования к функциональности

Программное обеспечение имеет клиент-серверную архитектуру и должно обеспечивать выполнение следующих функций.

Клиентская часть (мобильное приложение) выполняет функции:

1) регистрацию пользователя в системе, включающую заполнение пользователем логина и пароля с клавиатуры и отправку запроса серверной части на сохранение введенных данных в базу, посредством сети интернет;

2) авторизацию пользователя в системе по логину и паролю:

2.1) заполнение пользователем контактной информации, включающей:

2.1.1) имя пользователя (строка текста вводится пользователем с клавиатуры, является обязательным значением);

2.1.2) фотографию пользователя (выбирается пользователем существующее фото из галереи устройства или непосредственно снимается на камеру, является необязательным значением);

2.1.3) радиус получения заявок (выбирается пользователем из списка значений: принимает значения от 1 до 100 км или «получать все заявки»);

2.2) отправку введенных данных серверной части, посредством сети интернет;

2.3) сохранение введенных данных в памяти устройства;

3) добавление новой заявки на помощь (для авторизованных пользователей):

3.1) ввод пользователем данных заявки на помощь, включающей:

3.1.1) имя пользователя;

3.1.2) описание неисправности (строка текста вводится пользователем с клавиатуры устройства);

3.1.3) фото неисправности (выбирается пользователем существующее фото из галереи устройства или непосредственно снимается на камеру);

3.1.4) местоположение (определяется автоматически модулем для определения местоположения или выбирается пользователем на карте Google);

3.1.5) дату и время (определяется автоматически системой);

3.2) редактирование данных заявки пользователем в процессе заполнения;

3.3) отправку запроса на сохранение данных заявки в базу, размещенную на сервере с получением текстового сообщения, подтверждающего сохранение заявки (посредством сети интернет);

4) добавление сообщений о готовности оказать помощь (предложения помочь) к выбранной заявке (для авторизованных пользователей):

4.1) ввод пользователем данных предложения помощи, которое включает:

4.1.1) имя водителя (строка текста вводится пользователем с клавиатуры);

4.1.2) сообщение (строка текста вводится пользователем с клавиатуры);

4.1.3) местоположение (определяется автоматически модулем для определения местоположения или выбирается пользователем на карте Google);

4.1.4) дату и время (определяется автоматически устройством) добавления предложения помощи;

4.2) редактирование данных предложения помощи пользователем в процессе заполнения;

4.3) отправку запроса на сохранение данных предложения помощи в базу, размещенную на сервере (посредством сети интернет), с получением текстового сообщения, подтверждающего сохранение данных;

5) получение от серверной части списка заявок на помощь и отображение его в окне устройства;

6) получение от серверной части сведений о местоположении заявок и их отображение в виде отметок на карте Google;

7) выбор пользователем заявки из списка заявок или из отметок на карте;

7.1) определение и отображение маршрута от пользователя устройства до местоположения выбранной заявки на карте Google;

7.2) добавление предложения помощи к выбранной заявке (см. п. 4)

8) получение уведомлений о новых заявках в выбранном радиусе получения заявок (принимает значения от 1 до 100 км или «получать все заявки») и с выбранной частотой получения заявок (принимает значения от 1 до 60 мин.);

9) отображение последних добавленных заявок авторизованного пользователя, с возможностью изменения статуса заявки: «требуется помощь», «помощь оказывается», «помощь оказана»;

10) сохранение полученных данных от серверной части в базе данных SQLite мобильного устройства;

Серверная часть должна обеспечивать выполнение следующих функций:

11) предоставление сведений из базы данных, размещенной на сервере, о заявках пользователей, которые добавлены не позже 24 часов на момент получения, по запросу клиентской части;

12) выполнение сохранения данных заявки, полученной от клиентской части приложения, в базу данных, размещенную на сервере;

13) выполнение изменения статуса заявки по запросу клиентской части приложения;

14) добавление данных предложения помощи к заявке, полученных от клиентской части приложения, в базу данных, размещенную на сервере;

15) получение списка предложений помощи к заявке из базы данных, размещенной на сервере, по запросу клиентской части приложения;

16) сохранение данных нового пользователя, полученных от клиентской части приложения, в базе данных, размещенной на сервере, с проверкой наличия данного пользователя в системе;

17) предоставление данных пользователя по запросу клиентской части приложения.

#### 4.1.1 Исходные данные

Данные заявки на помощь:

- имя пользователя (строка текста);
- описание неисправности (текстовое описание);
- фото неисправности – файл изображения в графических форматах: .jpg, .png (является необязательным для заполнения пользователем);
- местоположение (широта и долгота), выбранные на карте Google или полученные автоматически с модуля для определения местоположения устройства;
- дату и время добавления (определяются автоматически системой).

Данные сообщения о готовности оказать помощь:

- имя пользователя (строка текста);
- текст сообщения (текстовое описание);
- местоположение (долгота и широта), выбранные на карте Google или полученные автоматически модулем для определения местоположения устройства;
- дату и время добавления (определяются системой автоматически).

Данные о пользователе:

- имя пользователя (строка текста);
- логин (строка текста, должен быть уникален в системе, длина от 4 до 50 символов);
- пароль (строка текста, длина от 4 до 50 символов);
- фото пользователя – файл изображения в графических форматах: .jpg, .png. (является необязательным для заполнения пользователем).

## 4.2 Требования к надежности

Предусмотреть:

- контроль вводимой, выводимой информации при вводе пользователем данных по типу данных, длине, допустимым диапазонам значений;
- блокировку вычислений и вывод сообщения пользователю при возникновении исключительной ситуации;

- проверку работы модуля мобильного устройства для осуществления сетевых операций;
- контроль проверки работы модуля мобильного устройства для определения местоположения;
- должен присутствовать контроль тайм-аута подключения к сети при запросе информации с сервера для исключения нештатной ситуации и экстренного завершения программы;
- сохранность данных пользователей в базе;
- недопустима ситуация просмотра личных данных пользователя, не прошедшего процедуру авторизации;
- контроль работы модуля работы с картами Google и вывод соответствующих сообщений об ошибках при возникновении исключительных ситуаций;
- предусмотреть контроль установки приложения на устройства с допустимой версией операционной системы Android (см. п. 4.4);
- предусмотреть шифрование паролей пользователей в базе данных;
- предусмотреть доступ к API серверной части по уникальному ключу.

#### 4.3 Условия эксплуатации

Приложение должно удовлетворять эксплуатационным требованиям:

- 1) пользовательский интерфейс приложения должен быть удобным, интуитивно-понятным и эффективным для использования, должен отображать текущее состояние подключения к сети;
- 2) компоненты пользовательского интерфейса должны быть разработаны с учетом специфики решаемой задачи;
  - 2.1) необходима поддержка компонента интерфейса для отображения списка заявок с возможностью отображения сообщения, имени и фотографии пользователя, времени добавления заявки, местоположения заявки. Компонент должен предусматривать:

2.1.1) быструю загрузку данных заявок из базы и их отображение на экране устройства;

2.1.2) выбор заявки из списка, с возможностью просмотра детальных сведений о заявке в отдельном окне приложения;

2.1.3) переход в режим просмотра заявок на карте, с возможностью выбора заявки и перехода к просмотру детальных сведений о заявке в отдельном окне;

2.2) необходима поддержка компонента для отображения детальных сведений о заявке (имя пользователя, местоположение, текст заявки и текст оставленных сообщений к заявке). Компонент должен предусматривать для выбранной заявки наличие интерфейсных компонентов для:

2.2.1) просмотра отметки заявки на карте Google;

2.2.2) определение маршрута до выбранной заявки;

2.2.3) добавление сообщения о готовности оказания помощи;

2.3) необходима поддержка интерфейсного компонента для отображения карт Google, с возможностью отметки местоположения и масштабирования карты, формирования списка заявок на карте, прорисовку маршрута до выбранной заявки;

2.4) необходима поддержка вкладок для переключения между режимом просмотра заявок в виде списка и режимом просмотра отметок заявок на карте;

2.5) необходима поддержка окна интерфейса для добавления и отображения сведений о пользователе устройства (имя, логин, пароль, фото, радиус получения заявок);

2.6) значения радиуса получения заявок должны быть представлены в виде компонента интерфейса, позволяющего пользователю непосредственно выбирать их из списка допустимых значений;

2.7) необходима поддержка интерфейсного компонента для просмотра уведомлений;

2.7.1) необходима поддержка компонента для отключения и включения уведомлений;

2.8) компоненты пользовательского интерфейса должны быть совместимы между минимально допустимой версией системы – Android 4.1 (Jelly Bean) и последней на текущий момент версией системы – Android 7.1 (Nougat).

#### 4.4 Требования к составу и параметру технических средств

Данное программное обеспечение должно использоваться на мобильных устройствах под управлением операционной системы Android с минимальной версией Android 4.1 (Jelly Bean) и выше со следующими характеристиками:

- сенсорный экран, использующийся для ввода, вывода, отображения данных;
- поддержка Wi-Fi, 2G/3G/4G модуля;
- камера 3,2 Мпикс и выше;
- процессор 1ГГц и выше;
- память 512 Мб ROM/ 256 Мб RAM;
- наличие установленного приложения Google Play Services;

#### 4.5 Требования к транспортированию и хранению

Пакетный файл клиентского приложения с расширением. apk устанавливается на мобильные устройства пользователей, которые удовлетворяют требованиям, представленным в п. 4.4.

База данных и файлы скриптов серверной части приложения размещаются на удаленном сервере под управлением web-сервера Apache.

### 5 Требования к программной документации

Программная документация содержит расчетно-пояснительную записку, техническое задание, спецификации на модули, тест программы, руководство пользователя, наборы тестовых данных и результатов тестирования.

## 6 Стадии и этапы разработки

Описание стадий и этапов разработки ПО представлено в табл. П1.1.

Таблица П1.1

### Стадии и этапы разработки

Наименование этапа разработки ПО	Сроки разработки	Результат выполнения	Отметка о выполнении
1	2	3	4
Изучение предметной области и разработка технического задания	08.02.17-16.02.17	Техническое задание	
Изучение аналогов разрабатываемой системы	17.02.17-20.02.17	Текстовое описание аналогов	
Выбор технологии и инструментальных средств для решения поставленной задачи	20.02.17-23.02.17	Описание выбранной технологии и инструментальных средств для решения задачи	
Разработка спецификаций проектируемого ПО	23.02.17-28.02.17	Диаграммы вариантов использования, контекстная диаграмма классов, диаграммы последовательностей системы с описанием	
Разработка структуры системы: проектирование диаграмм пакетов	01.03.17-03.03.17	Диаграмма пакетов системы с описанием	
Проектирование базы данных серверной части приложения	04.03.17-05.04.17	ER-диаграмма базы данных серверной части, детальное описание диаграммы	
Проектирование пакетов серверной части приложения	06.03.17-13.03.17	Диаграммы классов, диаграммы последовательностей действий для каждого пакета с детальным описанием	
Проектирование базы данных клиентской части приложения	14.03.17-15.03.17	ER-диаграмма базы данных клиентской части	
Проектирование пакетов клиентской части приложения	15.03.17-30.03.17	Диаграммы классов, диаграммы последовательностей действий для каждого пакета с детальным описанием	
Построение диаграмм размещения и компонентов	01.04.17-02.04.17	Диаграммы размещения и компонентов с детальным описанием	
Реализация БД серверной части приложения	03.04.17-06.04.17	Размещенная база данных на сервере	

## Продолжение табл. П1.1

1	2	3	4
Реализация БД клиентской части приложения	07.04.17-08.04.17	Реализованные модули проекта в среде Android Studio	
Реализация модулей для добавления новой заявки в систему	08.04.17-13.04.17	Реализованные модули проекта в среде Android Studio, файлы скриптов с расширением .php	
Реализация модулей для регистрации, авторизации пользователей в системе	14.04.17-17.04.17	Реализованные модули проекта в среде Android Studio, файлы скриптов с расширением .php	
Реализация модулей для просмотра учетной записи пользователя в системе	18.04.17-23.04.17	Реализованные модули проекта в среде Android Studio, файлы скриптов с расширением .php	
Реализация модулей для просмотра списка заявок	23.04.17-28.04.17	Реализованные модули проекта в среде Android Studio, файлы скриптов с расширением .php	
Реализация модулей для просмотра списка заявок на карте	28.04.17-30.04.17	Реализованные модули проекта в среде Android Studio, файлы скриптов с расширением .php	
Реализация модулей для просмотра детальной заявки	01.05.17-06.05.17	Реализованные модули проекта в среде Android Studio, файлы скриптов с расширением .php	
Реализация модулей для построения маршрута заявки	06.05.17-09.05.17	Реализованные модули проекта в среде Android Studio, файлы скриптов с расширением .php	
Реализация модулей для получения уведомлений о новых заявках	10.05.17-13.05.17	Реализованные модули проекта в среде Android Studio, файлы скриптов с расширением .php	
Разработка детальных спецификаций на модули	13.05.17-16.05.17	Детальные спецификации на модули	
Разработка тестовых данных для тестирования	20.04.17-15.05.17	Таблицы с набором тестовых данных	
Тестирование ПО	20.04.17-15.05.17	Результаты тестирования	
Подготовка расчетно-пояснительной записки	15.05.17-27.05.17	Расчетно-пояснительная записка	

## 7 Порядок контроля и приемки

Порядок контроля и приемки производится в соответствии со стадиями и этапами разработки (см. табл. П1.1).

## Текст программы

### Пакет ObjectsOfModel

#### Модуль Driver.java

```

package com.example.helponroad.ObjectsOfModel;
import java.util.Date;
/*
 * Created by Oleg on 13.03.2017.
 * Класс, описывающий сведения о водителе
 */
public class Driver {
    protected String mId;
    protected String mName;//Имя водителя
    protected String mImage;//Фотография водителя
    protected boolean isImage;//есть ли у водителя фотография
    protected Date mCreateDate;//дата создания записи о
    //пользователе на сервере
    protected Date mUpdateDate;//дата обновления записи о
    //пользователе на сервере
    protected Date mUploadDate;//дата загрузки записи о
    //пользователе на устройство

    public String getId() {
        return mId;
    }
    public void setId(String id) {
        mId = id;
    }
    public String getName() {
        return mName;
    }
    public void setName(String name) {
        mName = name;
    }
}
public String getImage() {
    return mImage;
}
public void setImage(String image){
    mImage=image;
}
public void setIsImage(boolean image) {
    isImage=image;
}
public boolean getIsImage() {
    return isImage;
}
public Date getCreateDate() {
    return mCreateDate;
}
public void setCreateDate(Date createDate) {
    mCreateDate = createDate;
}
public Date getUpdateDate() {
    return mUpdateDate;
}
public void setUpdateDate(Date updateDate) {
    mUpdateDate = updateDate;
}
public Date getUploadDate() {
    return mUploadDate;
}
public void setUploadDate(Date uploadDate) {
    mUploadDate = uploadDate;
}
}

```

Рис.П2.1. Текст модуля Driver.java

#### Модуль User.java

```

package com.example.helponroad.ObjectsOfModel;
import com.example.helponroad.NetworkOperations.MySQLDataBaseSchema;
import com.example.helponroad.NetworkOperations.QueryStrings;
import org.json.JSONException;
import org.json.JSONObject;

/*
 * Класс, описывающий пользователя устройства
 * Created by Oleg on 15.04.2017.
 */
public class User extends Driver {

    private String mLogin;
    private String mPassword;

    public String getLogin() {
        return mLogin;
    }
    public void setLogin(String login) {
        mLogin = login;
    }
}
public String getPassword() {
    return mPassword;
}
public void setPassword(String password) {
    mPassword = password;
}
/*
 * Возвращает объект пользователя в формате JSON с
 * указанием в списке параметров возвращаемые поля
 */
public JSONObject getJSONUserData(boolean isId,boolean
    isLogin,boolean isPassword, boolean isName,boolean
    isImage)throws JSONException{
    JSONObject jsonObject = new JSONObject();
    jsonObject.put(QueryStrings.ANDROID_KEY_NAME,QueryString
        s.ANDROID_KEY_VALUE);
    if(isId)
        jsonObject.put(MySQLDataBaseSchema.UsersTable.Cols.ID, mId);
    if(isLogin)
        jsonObject.put(MySQLDataBaseSchema.UsersTable.Cols.LOGIN,m
            Login);
}

```

Рис.П2.2. Текст модуля User.java

## П2.2. Продолжение

```

        if(isPassword)
jsonObject.put(MySQLDataBaseSchema.UsersTable.Cols.PASSWD,
RD,mPassword);
        if(isName)
jsonObject.put(MySQLDataBaseSchema.UsersTable.Cols.NAME,m
Name);
        if(isImage)
jsonObject.put(MySQLDataBaseSchema.UsersTable.Cols.IMAGE,
mImage);
        return jsonObject;
    }
}
public Date getUpdateDate() {
}

```

### Модуль Request.java

```

package com.example.helponroad.ObjectsOfModel;

import com.example.helponroad.
NetworkOperations.MySQLDataBaseSchema;
import com.example.helponroad. NetworkOperations.QueryStrings;
import com.google.android.gms.maps.model.LatLng;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.Date;
/*
 * Created by Олег on 13.03.2017.
 * Класс Request предназначен для описания заявки
 */
public class Request {
    private int mId;//Идентификатор заявки
    private String mMessage;//Текст сообщения заявки
    private boolean isImage;//есть ли у заявки фотография
    private String mImage;//Прикрепленное изображение
    private Driver mDriver;//Сведения о водителе
    private LatLng mLocation;//Сведения о местоположении
    private Status mStatus;//Статус заявки
    private Date mCreateDate;//Дата добавления заявки в
    private Date mUpdateDate;//Дата обновления заявки
    private Date mUploadDate;//Дата загрузки заявки на
    private boolean isNew;//Является ли заявка новой
    private boolean isGetNotify;//Было ли получено

    public int getId() {
        return mId;
    }
    public void setId(int id) {
        mId = id;
    }
    public String getMessage() {
        return mMessage;
    }
    public void setMessage(String message) {
        mMessage = message;
    }
    public boolean getIsImage() {
        return isImage;
    }
    public void setIsImage(boolean image) {
        isImage = image;
    }
    public String getImage() {
        return mImage;
    }
    public void setImage(String image) {
        mImage = image;
    }
    public Driver getDriver() {
}

```

```

        return mUpdateDate;
    }
    public void setUpdateDate(Date updateDate) {
        mUpdateDate = updateDate;
    }
    public Date getUploadDate() {
        return mUploadDate;
    }
    public void setUploadDate(Date uploadDate) {
        mUploadDate = uploadDate;
    }
}

```

```

        return mDriver;
    }
    public void setDriver(Driver driver) {
        mDriver = driver;
    }
    public LatLng getLocation() {
        return mLocation;
    }
    public void setLocation(LatLng location) {
        mLocation = location
    }
    public Status getStatus() {
        return mStatus;
    }
    public void setStatus(Status status) {
        mStatus = status;
    }
    public Date getCreateDate() {
        return mCreateDate;
    }
    public void setCreateDate(Date createDate) {
        mCreateDate = createDate;
    }
    public Date getUpdateDate() {
        return mUpdateDate;
    }
    public void setUpdateDate(Date updateDate) {
        mUpdateDate = updateDate;
    }
    public Date getUploadDate() {
        return mUploadDate;
    }
    public void setUploadDate(Date uploadDate) {
        mUploadDate = uploadDate;
    }
    public boolean isNew() {
        return isNew;
    }
    public void setNew(boolean aNew) {
        isNew = aNew;
    }
    public boolean isGetNotify() {
        return isGetNotify;
    }
    public void setGetNotify(boolean getNotify) {
        isGetNotify = getNotify;
    }
}

```

Рис.П2.3. Текст модуля Request.java

## П2.3. Продолжение

```

public JSONObject getRequestData(boolean isId, boolean
isMessage, boolean isLongitude, boolean isLatitude,
boolean isUserId, boolean isImage, boolean isStatus) throws
JSONException {
    JSONObject jsonObject = new JSONObject();
    jsonObject.put(QueryStrings.ANDROID_KEY_NAME, QueryStrings
.ANDROID_KEY_VALUE);
    if(isId)
        jsonObject.put(MySQLDataBaseSchema.RequestsTable.Cols.ID, mId
);
    if(isMessage)
        jsonObject.put(MySQLDataBaseSchema.RequestsTable.Cols.MESS
AGE, mMessage);
    if(isLongitude)
        jsonObject.put(MySQLDataBaseSchema.RequestsTable.Cols.LONG
ITUDE, mLocation.longitude);
    if(isLatitude)
        jsonObject.put(MySQLDataBaseSchema.RequestsTable.Cols.LATIT
UDE, mLocation.latitude);
    if(isUserId)
        jsonObject.put(MySQLDataBaseSchema.RequestsTable.Cols.USER
_ID, mDriver.getId());
    if(isImage)
        jsonObject.put(MySQLDataBaseSchema.RequestsTable.Cols.IMAG
E, mImage);
    if(isStatus)
        jsonObject.put(MySQLDataBaseSchema.RequestsTable.ColsSTAT
US_ID, mStatus.getStatusValue());
}

return jsonObject;
}

```

### Модуль Offer.java

```

package com.example.helponroad.ObjectsOfModel;
import com.example.helponroad.NetworkOperations.
MySQLDataBaseSchema;
import com.example.helponroad.NetworkOperations.QueryStrings
import org.json.JSONException;
import org.json.JSONObject;
import java.util.Date;

/**
 * Класс, описывающий предложение помочи к заявке
 * Created by Oleg on 06.05.2017.
 */
public class Offer {
    private int mId;//идентификатор предложения
    private String mMessage;//сообщение предложения
    private Date mCreateDate;//дата добавления
    private Date mUpdateDate;//дата обновления
    private Date mUploadDate;//дата загрузки на устройство
    private boolean isNew;//Является ли предложение помочи
    новым (true-да, false-нет)
    private boolean isGetNotify;//Было ли получено уведомление для
    предложения
    private Driver mDriver;//Сведения о водителе, который добавил
    заявку
    private int mRequestId;//идентификатор заявки, к которой
    добавлено предложение
    public int getId() {
        return mId;
    }
    public void setId(int id) {
        mId = id;
    }
    public String getMessage() {
        return mMessage;
    }
    public void setMessage(String message) {
        mMessage = message;
    }
    public Date getCreateDate() {
        return mCreateDate;
    }
    public void setCreateDate(Date createDate) {
        mCreateDate = createDate;
    }
    public Date getUpdateDate() {
        return mUpdateDate;
    }
    public void setUpdateDate(Date updateDate) {
        mUpdateDate = updateDate;
    }
    public Date getUploadDate() {
        return mUploadDate;
    }
    public void setUploadDate(Date uploadDate) {
        mUploadDate = uploadDate;
    }
    public boolean isNew() {
        return isNew;
    }
    public void setNew(boolean aNew) {
        isNew = aNew;
    }
    public boolean isGetNotify() {
        return isGetNotify;
    }
    public void setGetNotify(boolean getNotify) {
        isGetNotify = getNotify;
    }
    public Driver getDriver() {
        return mDriver;
    }
}

```

Рис.П2.4. Текст модуля Offer.java

## Модуль Status.java

```

package com.example.helponroad.ObjectsOfModel;

/*
 * Created by Олег on 27.04.2017.
 */

public class Status {
    private int mStatusValue;

    public int getStatusValue() {
        return mStatusValue;
    }

    public void setStatusValue(int statusValue) {
        this.mStatusValue = statusValue;
    }

    public String getStatusMessage(){
        String rez="";
        switch (mStatusValue){
            case 0:{
                rez="Требуется помощь";
                break;
            }
            case 1:{
                rez="Помощь оказывается";
                break;
            }
            case 2:{
                rez="Помощь оказана";
                break;
            }
        }
        return rez;
    }
}

```

Рис.П2.5. Текст модуля Status.java

## Модуль OfferList.java

```

package com.example.helponroad.ObjectsOfModel;

import android.content.ContentValues;
import android.content.Context;
import com.example.helponroad.ManageObjectsOfModel.OfferManager;
import com.example.helponroad.SQLiteDataBaseInterface.OfferCursorWra
pper;
import com.example.helponroad.SQLiteDataBaseInterface.OfferQuery;
import java.util.ArrayList;
import java.util.List;
/***
 * Синглентный класс для получения списка заявок
 */
public class OfferList {

    private static OfferList sOfferList;
    private OfferManager mOfferManager;
    private OfferQuery mOfferQuery;
    private Context mContext;

    public static OfferList get(Context context){
        if(sOfferList ==null){
            sOfferList =new OfferList(context);
        }
        return sOfferList;
    }

    private OfferList(Context context){
        mOfferManager=new OfferManager();
        mOfferQuery=new OfferQuery(context);
        mContext=context.getApplicationContext();
    }

    public List<Offer> getOffers(int requestId) {
        mOfferQuery.openConnection();
        List<Offer> mOffers = new ArrayList<>();
        OfferCursorWrapper cursor =
        mOfferQuery.getOffers(requestId);

        try {
            cursor.moveToFirst();
            while (!cursor.isAfterLast()) {
                mOffers.add(cursor.getOffer());
                cursor.moveToNext();
            }
        } finally{
            cursor.close();
            mOfferQuery.closeConnection();
        }
        return mOffers;
    }

    public void setNotNewOffer(Offer offer){
        try {
            mOfferQuery.openConnection();

            ContentValues
offerContentValues=OfferQuery.getOfferValues(offer,true,false,false,
false,
false,false,true,false);
            mOfferQuery.updateOffer(offerContentValues);
        } finally {
            mOfferQuery.closeConnection();
        }
    }
}

```

Рис.П2.6. Текст модуля OfferList.java

## Модуль RequestList.java

```

package com.example.helponroad.ObjectsOfModel;

import android.content.ContentValues;
import android.content.Context;
import com.example.helponroad.ManageObjectsOfModel.RequestManager;
import com.example.helponroad.ManageObjectsOfModel.UserManager;
import com.example.helponroad.ManageObjectsOfModel.UserPreferenceS;
import com.example.helponroad.SQLiteDataBaseInterface.RequestCursorWrapper;
import com.example.helponroad.SQLiteDataBaseInterface.RequestQuery;
import com.google.android.gms.maps.model.LatLng;
import java.util.ArrayList;
import java.util.List;

/**
 * Синглентный класс для получения данных о заявках (список
 * заявок, заявка по идентификатору)
 * Created by Олег on 27.04.2017.
 */

public class RequestList {

    private static RequestList sRequestList;
    private RequestManager mRequestManager;
    private RequestQuery mRequestQuery;
    private Context mContext;

    public static RequestList get(Context context){
        if(sRequestList==null){
            sRequestList=new RequestList(context);
        }
        return sRequestList;
    }

    private RequestList(Context context){
        mRequestManager=new RequestManager();
        mRequestQuery=new RequestQuery(context);
        mContext=context.getApplicationContext();
    }

    /**
     * Получаем список заявок из базы SQLite
     * Вычисляем расстояние от пользователя до заявки по
     * формуле гаверсинусов,
     * если это расстояние не превышает радиус получения
     * заявок, то добавляем заявку в список
     * в случае, если радиус получения заявок равен -1
     * (получать все заявки),
     * то возвращаем все имеющиеся актуальные заявки из базы
     * @param userLocation-текущее местоположение
     * пользователя
     * @return -список заявок
     */
    public List<Request> getRequests(LatLng userLocation){

        UserPreferences userPreferences=new
        UserPreferences(mContext);
        int
radiusNotification=userPreferences.getStoredRadiusNotifications()
//радиус получения
//заявок
        List<Request> mRequests=new ArrayList<>();

        mRequestQuery.openConnection();

        RequestCursorWrapper
cursor=mRequestQuery.getRequests();
try{
    cursor.moveToFirst();
    while (!cursor.isAfterLast()){
        Request request=cursor.getRequest();

        //если не задано получать все заявки и
        //местоположение пользователя!=null то фильтруем их по
        //расстоянию
        if(radiusNotification!=-1 && userLocation!=null) {

            //Вычисляем расстояние от пользователя до заявки
            double
distance=RequestManager.calculateDistance(request,
userLocation);

            //Фильтруем заявки по расстоянию до
            //пользователя
            if(distance/1000<=radiusNotification){
                mRequests.add(request);
            }
        } else{//в противном случае возвращаем все
        //актуальные заявки
                mRequests.add(request);
            }
        cursor.moveToNext();
    }
finally {
    cursor.close();
    mRequestQuery.closeConnection();
}

        return mRequests;
    }

    public List<Request> getRequestsAppUser(){

        UserPreferences userPreferences=new
        UserPreferences(mContext);
        UserManager mUserManager=new
        UserManager(userPreferences);

        List<Request> mRequests=new ArrayList<>();

        mRequestQuery.openConnection();

        RequestCursorWrapper
cursor=mRequestQuery.getRequestsAppUser(mUserManager.getU
ser());
try{
    cursor.moveToFirst();
    while (!cursor.isAfterLast()){
        Request request=cursor.getRequest();
        mRequests.add(request);
        cursor.moveToNext();
    }
finally {
    cursor.close();
    mRequestQuery.closeConnection();
}

        return mRequests;
    }
}

```

Рис.П2.7. Текст модуля RequestList.java

## П2.7. Продолжение

```

public List<Request> getNewRequest(LatLang userLocation){

    UserPreferences userPreferences=new
    UserPreferences(mContext);
    int
radiusNotification=userPreferences.getStoredRadiusNotifications()
://радиус получения заявок

    List<Request> mRequests=new ArrayList<>();

    mRequestQuery.openConnection();

    RequestCursorWrapper
cursor=mRequestQuery.getNewRequests(new
UserManager(userPreferences)
    .getUser());

    try{
        cursor.moveToFirst();
        while (!cursor.isAfterLast()){
            Request request=cursor.getRequest();

            //если не задано получать все заявки и
местоположение пользователя!=null
                // то фильтруем их по расстоянию
            if(radiusNotification!=-1 && userLocation!=null) {
                //Вычисляем расстояние от пользователя до заявки
                double
distance=RequestManager.calculateDistance(request,
userLocation);
                //Фильтруем заявки по расстоянию до
пользователя
                if(distance/1000<=radiusNotification){
                    mRequests.add(request);
                }
            }else{//в противном случае возвращаем все
актуальные заявки
                mRequests.add(request);
            }

            cursor.moveToNext();
        }
    finally {
        cursor.close();
        mRequestQuery.closeConnection();
    }

    return mRequests;
}

/**
 * Возвращает сведения о заявке с выбранным
идентификатором
 * @param requestId- идентификатор заявки
 * @return данные заявки Request
 */
public Request getRequest(int requestId){

    mRequestQuery.openConnection();

    RequestCursorWrapper
cursor=mRequestQuery.getRequest(requestId);

    try{
        if (cursor.getCount() == 0) {
            return null;
        }

        cursor.moveToFirst();

        return cursor.getRequest();

    }finally {
        cursor.close();
        mRequestQuery.closeConnection();
    }
}

public void setNotNewRequest(Request request){

    try {
        mRequestQuery.openConnection();

        ContentValues offerContentValues=
RequestQuery.getRequestValues(request,true,false,
false,false,
false,false,false,false,false,false,true,false);

        mRequestQuery.updateRequest(offerContentValues);
    }finally {
        mRequestQuery.closeConnection();
    }
}

public void setGetNotify(Request request){

    try {
        mRequestQuery.openConnection();

        ContentValues offerContentValues=
RequestQuery.getRequestValues(request,true,false,
false,false,
false,false,false,false,false,false,false,true);

        mRequestQuery.updateRequest(offerContentValues);
    }finally {
        mRequestQuery.closeConnection();
    }
}

public void setStatus(Request request){

    try {
        mRequestQuery.openConnection();

        ContentValues offerContentValues=
RequestQuery.getRequestValues(request,true,false,
false,false,
false,false,false,false,false,true,false,false);

        mRequestQuery.updateRequest(offerContentValues);
    }finally {
        mRequestQuery.closeConnection();
    }
}

```

## Модуль LocationListenerGPSServices.java

```

package com.example.helponroad.LocationManager;
import android.content.Context;
import android.location.Location;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.util.Log;
import android.view.Gravity;
import android.widget.Toast;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GooglePlayServicesNotAvailableException;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import static java.lang.Integer.MAX_VALUE;

/**
 * Класс для получения данных местоположения, используя
Fused Location Provider
 * Created by Oler on 22.03.2017.
 */

public class LocationListenerGPSServices implements
GoogleApiClient.ConnectionCallbacks,
GoogleApiClient.OnConnectionFailedListener,LocationListener {

    public static final int MILLISECONDS_PER_SECOND = 1000;

    // количество секунд для обновления
    private static final int UPDATE_INTERVAL_IN_SECONDS =
10;

    //количество секунд, для обновления в активном режиме
    private static final int FAST_CEILING_IN_SECONDS = 10;

    // интервал обновления в миллисекундах
    private static final int
UPDATE_INTERVAL_IN_MILLISECONDS =
        MILLISECONDS_PER_SECOND *
UPDATE_INTERVAL_IN_SECONDS;

    // Интервал обновления, когда приложения видно
    private static final int
FAST_INTERVAL_CEILING_IN_MILLISECONDS =
        MILLISECONDS_PER_SECOND *
FAST_CEILING_IN_SECONDS;

    private Location mLocation;
    private GoogleApiClient mGoogleApiClient;
    private Context mContext;
    private int mInterval;
    private int mFastUpdateInterval;
    private int mNumUpdates= Integer.MAX_VALUE;
    private LocatingListener mLocatingListener;//слушатель
    события изменения местоположения
    private final static String TAG="LocationListener";

    public LocationListenerGPSServices(Context context,int
interval,int fastUpdateInterval) {
        mContext = context;
        this.mInterval =interval;
        this.mFastUpdateInterval =fastUpdateInterval;
    }

    /**
     * Метод для получения текущего местоположения
     * @return текущее местоположение
     */
    public Location getCurrentLocation() {
        return mLocation;
    }

    /**
     * Метод для задания текущего местоположения
     * @param location -местоположение
     */
    public void setCurrentLocation(Location location) {
        mLocation = location;
    }
}

public LocationListenerGPSServices(Context context,int
interval,int fastUpdateInterval,int numUpdates) {
    mContext = context;
    this.mInterval =interval;
    this.mFastUpdateInterval =fastUpdateInterval;
    this.mNumUpdates= numUpdates;
}

@Override
public void onConnected(@Nullable Bundle bundle){
    try {
        if(mGoogleApiClient.isConnected()){
            enableLocationUpdates(mInterval,
mFastUpdateInterval,mNumUpdates);
        }
    } catch (LocationException ex){
        Toast
toast=Toast.makeText(mContext,ex.getMessage(),Toast.LENGTH_
LONG);
toast.setGravity(Gravity.CENTER, 0, 0);
toast.show();
    }
}

@Override
public void onConnectionSuspended(int i) {
}

@Override
public void onConnectionFailed(@NonNull ConnectionResult
connectionResult) {
}

@Override
public void onLocationChanged(Location location) {
    mLocation = location;
    mLocatingListener.onUpdateLocation(mLocation);
    Log.i(TAG,"Текущее
местоположение:"+location.toString());
}

/**
 *get-метод для получения значения текущего
местоположения
 * @return текущее местоположение
 */
public Location getCurrentLocation() {
    return mLocation;
}

/**
 *set-метод для задания значения текущего местоположения
 * @param location -местоположение
 */
public void setCurrentLocation(Location location) {
    mLocation = location;
}

```

Рис.П2.8 Текст модуля LocationListenerGPSServices.java

```

/**
 * Устанавливает слушателя на событие находления
 * местоположения пользователя
 *
 * @param locationRunnable слушатель
 */
public void setLocatingListener(LocatingListener
locationRunnable) {
    this.mLocatingListener = locationRunnable;
}

/**
 * Включает определения местоположения
 * Проверяет доступность GoogleApiClient, настраивает
GoogleApiClient для работы с API Location
 * Services и подключает его
 * @throws GooglePlayServicesNotAvailableException
*/
private void enableLocationUpdates(int updateInterval,int
fastUpdateInterval,int numUpdates)
    throws LocationException {
    LocationRequest locationRequest=
locationRequestBuilder(LocationRequest.PRIORITY_HIGH_ACC
URACY,updateInterval,
        fastUpdateInterval,numUpdates);
    LocationPermission locationPermission=new
LocationPermission(mContext);

if(locationPermission.checkAccessCoarseLocationPermission()
    && locationPermission.checkAccessFineLocationPermission()) {
    LocationServices.FusedLocationApi.requestLocationUpdates(mGo
ogleApiClient,locationRequest,this);
} else {
    throw new LocationException("Отсутствует разрешение
для определения местоположения."+
        "\n В меню приложения перейдите в пункт
        'Настройки' (нажмите кнопку 'Изменить настройки'). В
        разделе 'Разрешения' установите разрешение для
        определения местоположения");
}
}

/**
 * Выключает обновления местоположения
 */
public void stopLocating(){
    if(mGoogleApiClient!=null) {
        if (mGoogleApiClient.isConnected()) {
            LocationServices.FusedLocationApi.removeLocationUpdates(mG
oogleApiClient, this);
            mGoogleApiClient.disconnect();
            Log.i(TAG,"Stop locating");
        }
    }
}

/**
 * Метод-обертка для построения запроса на определение
местоположения
 * @param priority-как следует поступать в ситуации выбора
между расходом заряда
 * @param interval-как часто должны обновляться данные о
местоположении
 * @param fastestInterval-как часто должны обновляться
данные о местоположении(при использовании
 *      Google-сервисов)
 * @return возвращает объект LocationRequest(запрос на
определение местоположения)
*/
private LocationRequest locationRequestBuilder(int priority,int
interval,int fastestInterval,int numUpdates){
    LocationRequest locationRequest=LocationRequest.create();
    locationRequest.setPriority(priority);
    locationRequest.setInterval(interval);
    locationRequest.setFastestInterval(fastestInterval);
}

locationRequest.setNumUpdates(numUpdates);

return locationRequest;
}

/**
 * Запускает определение местоположения, путем создания
запроса LocationRequest и передачи его
 * GoogleApiClient
 * Вызывается в методе onConnected переопределяемого
интерфейса GoogleApiClient.ConnectionCallbacks
 * @throws LocationException
*/
private void enableLocationUpdates(int updateInterval,int
fastUpdateInterval,int numUpdates)
    throws LocationException {
    LocationRequest locationRequest=
locationRequestBuilder(LocationRequest.PRIORITY_HIGH_ACC
URACY,updateInterval,
        fastUpdateInterval,numUpdates);
    LocationPermission locationPermission=new
LocationPermission(mContext);

if(locationPermission.checkAccessCoarseLocationPermission()
    && locationPermission.checkAccessFineLocationPermission()) {
    LocationServices.FusedLocationApi.requestLocationUpdates(mGo
ogleApiClient,locationRequest,this);
} else {
    throw new LocationException("Отсутствует разрешение
для определения местоположения."+
        "\n В меню приложения перейдите в пункт
        'Настройки' (нажмите кнопку 'Изменить настройки'). В
        разделе 'Разрешения' установите разрешение для
        определения местоположения");
}
}
}

```

## Модуль LocationPermission.java

```

package com.example helponroad.LocationManager;

import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.support.v4.app.ActivityCompat;
import android.util.Log;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GooglePlayServicesUtil;

/**
 * Класс, описывающий разрешения для работы с сетью
 * Created by Oleg on 01.04.2017.
 */
public class LocationPermission {

    private final Context mContext;
    private final static String TAG="LocationPermission";

    public LocationPermission(final Context context) {
        mContext = context;
    }

    /**
     * Проверка в Манифесте приложения разрешение на
     * определение точного местоположения
     *
     * @return возвращает true в случае успеха
     */
    public boolean checkAccessFineLocationPermission(){
        boolean result=true;

        if(ActivityCompat.checkSelfPermission(
            mContext,
            Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED){
            Log.e(TAG, "Отсутствует разрешение
ACCESS_FINE_LOCATION");
            result=false;
        }
    }

    /**
     * Проверка в Манифесте приложения разрешение на
     * определение приближенного местоположения
     *
     * @return возвращает true в случае успеха
     */
    public boolean checkAccessCoarseLocationPermission(){
        boolean result=true;
        if(ActivityCompat.checkSelfPermission(
            mContext,
            Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED){
            Log.e(TAG, "Отсутствует разрешение
ACCESS_COARSE_LOCATION");
            result=false;
        }
    }

    /**
     * Проверяет, можно ли подключиться к типу определения
     * местоположения
     *
     * @return возвращает true в случае успеха
     */
    public boolean googlePlayServicesConnected(){
        //проверим, можем ли мы использовать сервисы
        final int resultCode =
        GooglePlayServicesUtil.isGooglePlayServicesAvailable(mContext);
        return ConnectionResult.SUCCESS == resultCode;
    }
}

```

Рис.П2.9. Текст модуля LocationPermission.java

## Модуль LocatingListener.java

```

package com.example helponroad.LocationManager;

import android.location.Location;

/**
 * Определяет интерфейс обратного вызова (слушателя
 * изменения местоположения)
 * Created by Oleg on 01.04.2017.
 */
public interface LocatingListener {
    /**
     * Вызывается, когда местоположение пользователя
     * определяется
     *
     * @param location местоположение пользователя
     */
    public void onUpdateLocation(Location location);
}

```

Рис.П2.10. Текст модуля LocatingListener.java

## Модуль LocationException.java

```
package com.example.helponroad.LocationManager;
public class LocationException extends Exception{
    public LocationException() {
        super();
    }
    public LocationException(String message) {
        super(message);
    }
}
```

Рис.П2.11. Текст модуля LocationException.java

## Пакет SQLiteDataBaseInterface

### Модуль HelpOnRoadBase.java

```
package com.example.helponroad.SQLiteDataBaseInterface;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import static
com.example.helponroad.SQLiteDataBaseInterface.SQLiteDataBa
seSchema.*;
/*
 * Класс для работы HelpOnRoadBase с локальной базой
 данных SQLite приложения
 */
public class HelpOnRoadBase extends SQLiteOpenHelper {
    private static final int VERSION=1;
    private static final String DATABASE_NAME="HelpOnRoadBase.db";
    public HelpOnRoadBase(Context context){
        super(context,DATABASE_NAME,null,VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table " + UsersTable.NAME + "(" +
                UsersTable.Cols.ID+ " integer primary key
                autoincrement, " +
                UsersTable.Cols.NAME + " text NOT NULL, " +
                UsersTable.Cols.IS_IMAGE+ " boolean NOT NULL, " +
                UsersTable.Cols.IMAGE + " text, " +
                UsersTable.Cols.CREATE_DATE+ " text NOT NULL,
                " +
                UsersTable.Cols.UPDATE_DATE+ " text, " +
                UsersTable.Cols.UPLOAD_DATE+ " text NOT NULL "
                +
                ")"
        );
        db.execSQL("create table " + RequestsTable.NAME+ "(" +
                RequestsTable.Cols.ID+ " integer primary key, " +
                RequestsTable.Cols.MESSAGE + " text NOT NULL, "
                +
                RequestsTable.Cols.LATITUDE + " real NOT NULL, "
                +
                RequestsTable.Cols.LONGITUDE+ " real NOT NULL,
                "
                +
                RequestsTable.Cols.IS_IMAGE+ " boolean NOT NULL, " +
                RequestsTable.Cols.IMAGE+ " text, " +
                RequestsTable.Cols.CREATE_DATE+ " text NOT
                NULL, " +
                RequestsTable.Cols.UPDATE_DATE+ " text, " +
                RequestsTable.Cols.UPLOAD_DATE+ " text NOT NULL
                "
                +
                RequestsTable.Cols.STATUS+ " text NOT NULL
                CHECK (" +RequestsTable.Cols.STATUS+ " IN (0,1,2), " +
                RequestsTable.Cols.IS_NEW+ " boolean NOT NULL
                DEFAULT 1, " +
                RequestsTable.Cols.IS_GET_NOTIFY+ " boolean NOT
                NULL DEFAULT 0, " +
                RequestsTable.Cols.USER_ID+ " integer, " +
                "foreign key (" +RequestsTable.Cols.USER_ID+"")+
                " references "+UsersTable.NAME+"
                (" +UsersTable.Cols.ID+ " )"+ "
                ");
        db.execSQL("create table " + OffersTable.NAME+ "(" +
                OffersTable.Cols.ID+ " integer primary key, " +
                OffersTable.Cols.MESSAGE + " text NOT NULL, " +
                OffersTable.Cols.CREATE_DATE+ " text NOT NULL,
                "
                +
                OffersTable.Cols.UPDATE_DATE+ " text, " +
                OffersTable.Cols.UPLOAD_DATE+ " text NOT NULL,
                "
                +
                OffersTable.Cols.IS_NEW+ " boolean NOT NULL
                DEFAULT 1, " +
                OffersTable.Cols.IS_GET_NOTIFY+ " boolean NOT
                NULL DEFAULT 0, " +
                OffersTable.Cols.USER_ID+ " integer, " +
                OffersTable.Cols.REQUEST_ID+ " integer, " +
                "foreign key (" +OffersTable.Cols.USER_ID+"")+
                " references "+UsersTable.NAME+"
                (" +OffersTable.Cols.ID+ " ), " +
                "foreign key (" +OffersTable.Cols.REQUEST_ID+"")+
                " references "+RequestsTable.NAME+"
                (" +RequestsTable.Cols.ID+ " ) ON DELETE CASCADE"+
                ")"
        );
    }
    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i,
    int i1) {
    }
}
```

Рис.П2.12. Текст модуля HelpOnRoadBase.java

## Модуль OfferCursorWrapper.java

```

package com.example.helponroad.SQLiteDataBaseInterface;

import android.database.Cursor;
import android.database.CursorWrapper;
import com.example.helponroad.ObjectsOfModel.Driver;
import com.example.helponroad.ObjectsOfModel.Offer;
import com.example.helponroad.JSONParsers.DateParser;
import com.example.helponroad.SQLiteDataBaseInterface.SQLiteDataBaseSchema.OffersTable;
import com.example.helponroad.SQLiteDataBaseInterface.SQLiteDataBaseSchema.UsersTable;
import java.util.Date;

/**
 * Обертка для курсора, который получает данные о
 * предложениях помощи
 * Created by Олег on 06.05.2017.
 */

public class OfferCursorWrapper extends CursorWrapper {

    public OfferCursorWrapper(Cursor cursor) {
        super(cursor);
    }

    //Заполняем объект предложение
    public Offer getOffer() {

        int id=getInt(getColumnIndex(OffersTable.Cols.ID));
        String message=getString(getColumnIndex(OffersTable.Cols.MESSAGE));
        int offerRequestId=getInt(getColumnIndex(OffersTable.Cols.REQUEST_ID));
        boolean isNew=getInt(getColumnIndex(OffersTable.Cols.IS_NEW))==1;
        boolean isGetNotify=getInt(getColumnIndex(OffersTable.Cols.IS_GET_NOTIFY))==1;

        //Даты загрузки обновления
        Date offerCreateDate=
            DateParser.parseStringToDate(getString(getColumnIndex(OffersTable.NAME+"_"+OffersTable.Cols.CREATE_DATE)));
        String stringOfferUpdateDate=getString(getColumnIndex(OffersTable.NAME+"_"+OffersTable.Cols.UPDATE_DATE));
        Date offerUpdateDate=null;
        if(stringOfferUpdateDate!=null) {
            offerUpdateDate =
                DateParser.parseStringToDate(stringOfferUpdateDate);
        }
    }
}

```

```

DateParser.parseStringToDate(stringOfferUpdateDate);
}
Date offerUploadDate=

DateParser.parseStringToDate(getString(getColumnIndex(OffersTable.NAME+"_"+OffersTable.Cols.UPLOAD_DATE)));
//Заполняем данные о водителе
Driver driver=new Driver();

driver.setId(getString(getColumnIndex(OffersTable.Cols.USER_ID)));
driver.setName(getString(getColumnIndex(UsersTable.Cols.NAME)));
driver.setIsImage(getInt(getColumnIndex(UsersTable.Cols.IS_IMAGE))==1);
//Даты загрузки данных о пользователе
Date
driverCreateDate=DateParser.parseStringToDate(getString(getColumnIndex(UsersTable.NAME+"_"+UsersTable.Cols.CREATE_DATE)));
driver.setCreateDate(driverCreateDate);
String
stringDriverUpdateDate=getString(getColumnIndex(UsersTable.NAME+"_"+UsersTable.Cols.UPDATE_DATE));
if(stringDriverUpdateDate!=null) {
    driver.setUpdateDate(DateParser.parseStringToDate(stringDriverUpdateDate));
}
Date
driverUploadDate=DateParser.parseStringToDate(getString(getColumnIndex(UsersTable.NAME+"_"+UsersTable.Cols.UPLOAD_DATE)));
driver.setUploadDate(driverUploadDate);
//Заполняем данные о заявке
Offer offer=new Offer();
offer.setId(id);
offer.setMessage(message);
offer.setRequestId(offerRequestId);
offer.setNew(isNew);
offer.setGetNotify(isGetNotify);
offer.setCreateDate(offerCreateDate);
offer.setUpdateDate(offerUpdateDate);
offer.setUploadDate(offerUploadDate);
offer.setDriver(driver);

return offer;
}
}

```

Рис.П2.13. Текст модуля OfferCursorWrapper.java

## Модуль OfferQuery.java

```

package com.example.helponroad.SQLiteDataBaseInterface;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import com.example.helponroad.ObjectsOfModel.Offer;
import com.example.helponroad.JSONParsers.DateParser;

```

```

import
com.example.helponroad.SQLiteDataBaseInterface.SQLiteDataBaseSchema.OffersTable;
import
com.example.helponroad.SQLiteDataBaseInterface.SQLiteDataBaseSchema.UsersTable;

```

Рис.П2.14. Текст модуля OfferQuery.java

## П2.14. Продолжение

```


    /**
     * Класс для запросов к таблице Offers
     * Created by Oleg on 06.05.2017.
     */
}

public class OfferQuery {

    private static final String TAG = "OfferQuery";

    private HelpOnRoadBase mHelpOnRoadBase;
    private SQLiteDatabase mDatabase;
    private Context mContext;

    public OfferQuery(Context context) {
        mContext = context;
        mHelpOnRoadBase = new HelpOnRoadBase(mContext);
    }

    public void openConnection() {
        if(mContext!=null) {
            mDatabase = mHelpOnRoadBase.getWritableDatabase();
        }
    }

    public void closeConnection() {
        if(mDatabase!=null) {
            mDatabase.close();
        }
    }

    public boolean checkConnection() {
        return mDatabase != null && mDatabase.isOpen();
    }

    public static ContentValues getOfferValues(Offer offer, boolean
isId, boolean isMessage, boolean isCreateDate, boolean
isUpdateDate,boolean isUploadDate, Boolean isUserId, boolean
isRequestId, boolean isNew,boolean isGetNotify) {
        ContentValues values = new ContentValues();
        if (isId) values.put(OffersTable.Cols.ID, offer.getId());
        if (isMessage) values.put(OffersTable.Cols.MESSAGE,
offer.getMessage());
        if (isCreateDate)
values.put(OffersTable.Cols.CREATE_DATE,
        DateParser.parseDateToString(offer.getCreateDate()));
        if (isUpdateDate) {
            if (offer.getUpdateDate() != null) {
                values.put(OffersTable.Cols.UPDATE_DATE,
DateParser.parseDateToString(offer.getUpdateDate()));
            }
        }
        if (isUploadDate)
values.put(OffersTable.Cols.UPLOAD_DATE,
        DateParser.parseDateToString(offer.getUploadDate()));
        if (isUserId) values.put(OffersTable.Cols.USER_ID,
offer.getDriver().getId());
        if (isRequestId) values.put(OffersTable.Cols.REQUEST_ID,
offer.getRequestId());
        if (isNew) values.put(OffersTable.Cols.IS_NEW,
offer.isNew()?1:0);
        if (isGetNotify)
values.put(OffersTable.Cols.IS_GET_NOTIFY,
offer.isGetNotify()?1:0);

        return values;
    }

    /**
     * Добавляем данные предложения помощи в базу
     *
     * @param offerContentValue-добавляемые данные
     * предложения помохи (имя поля записи, значение поля записи)
     */
    public void addOffer(ContentValues offerContentValue) {
        if (checkConnection()) {
            mDatabase.insertWithOnConflict(OffersTable.NAME,
null, offerContentValue, SQLiteDatabase.CONFLICT_IGNORE);
        }
    }

    /**
     * Обновляем данные заявки в базе
     *
     * @param requestContentValue-обновляемые данные (имя
     * поля, значение поля записи)
     */
    public void updateOffer(ContentValues requestContentValue) {
        if (checkConnection()) {
            String idString = requestContentValue.getAsString("id");
            ContentValues values = requestContentValue;
            mDatabase.update(OffersTable.NAME, values,
OffersTable.Cols.ID + "=?", new String[]{idString});
        } else{
            Log.e(TAG,"Требуется открыть подключение к базе
данных");
            throw new SQLException("Требуется открыть
подключение к базе данных");
        }
    }

    /**
     * Получение списка предложений по заявке с
     * идентификатором id из базы из таблиц Offers, Users
     * @param requestId - идентификатор заявки
     * @return - курсор OfferCursorWrapper , который содержит
     * данные предложений помоши
     */
    public OfferCursorWrapper getOffers(int requestId){
        if (checkConnection()) {
            String query = "SELECT " + OffersTable.NAME + "."
+ OffersTable.Cols.ID + " AS " + OffersTable.Cols.ID + ", "
+ OffersTable.NAME + "."
+ OffersTable.Cols.MESSAGE + " AS "
+ OffersTable.Cols.MESSAGE + ", "
+ OffersTable.NAME + "."
+ OffersTable.Cols.CREATE_DATE + " AS "
+ OffersTable.NAME+"_" + OffersTable.Cols.CREATE_DATE + ",
"
+
OffersTable.NAME + "."
+ OffersTable.Cols.UPDATE_DATE + " AS "
+ OffersTable.NAME+"_" + OffersTable.Cols.UPDATE_DATE + ",
"
+
OffersTable.NAME + "."
+ OffersTable.Cols.UPLOAD_DATE + " AS "
+ OffersTable.NAME+"_" + OffersTable.Cols.UPLOAD_DATE + ",
"
+
OffersTable.NAME+ "."
+ OffersTable.Cols.IS_NEW +
OffersTable.NAME+ "."
+ OffersTable.Cols.IS_GET_NOTIFY + " AS "
+ OffersTable.Cols.IS_GET_NOTIFY + ", "
+
OffersTable.NAME+ "."
+ OffersTable.Cols.REQUEST_ID + " AS "
+ OffersTable.Cols.REQUEST_ID + ", "
+
UsersTable.NAME + "."
+ UsersTable.Cols.ID + " AS "
+ OffersTable.Cols.USER_ID + ", "
+
UsersTable.NAME + "."
+ UsersTable.Cols.NAME +
" AS "
+ UsersTable.Cols.NAME + ", "
+
UsersTable.NAME + "."
+ UsersTable.Cols.IS_IMAGE + " AS "
+ UsersTable.Cols.IS_IMAGE + ", "
+
UsersTable.NAME + "."
+ UsersTable.Cols.CREATE_DATE + " AS "
+ UsersTable.NAME+"_" + UsersTable.Cols.CREATE_DATE + ",
"
+
UsersTable.NAME + "."
+ UsersTable.Cols.UPDATE_DATE + " AS "
+ UsersTable.NAME+"_" + UsersTable.Cols.UPDATE_DATE + ", "
+
UsersTable.NAME + "."
+ UsersTable.Cols.UPLOAD_DATE + " AS "
+


```

## П2.14. Продолжение

```

UsersTable.NAME+"_" + UsersTable.Cols.UPLOAD_DATE + " "
+
    "FROM " + OffersTable.NAME + ", " +
UsersTable.NAME + " " +
    "WHERE " + OffersTable.NAME + ". " +
OffersTable.Cols.USER_ID +"=" + UsersTable.NAME + ". " +
UsersTable.Cols.ID+
        " AND "+OffersTable.Cols.REQUEST_ID +"=?";
Cursor cursor = mDatabase.rawQuery(query, new
String[]{Integer.toString(requestId)});
Log.i(TAG, "Запрос: " + query);
return new OfferCursorWrapper(cursor);
}else{
    Log.e(TAG,"Требуется открыть подключение к базе
данных");
    throw new SQLException("Требуется открыть
подключение к базе данных");
}
}

//Возвращает идентификатор последнего предложения для
данной заявки
public int getLastOfferId(int requestId){
    int maxId = -1;
    if(checkConnection()) {
        String query = "SELECT MAX(" + OffersTable.Cols.ID +
")" +
            " FROM " + OffersTable.NAME +
            " WHERE " + OffersTable.Cols.REQUEST_ID +
"=?";
        Cursor cursor = mDatabase.rawQuery(query, new
String[]{Integer.toString(requestId)} );
        if(cursor != null) {
            if(cursor.moveToFirst()){
                maxId = cursor.getInt(0);
            }
            cursor.close();
        }else{
            Log.e(TAG, "Требуется открыть подключение к базе
данных");
        }
        return maxId;
    }
}
}

```

### Модуль RequestCursorWrapper.java

```

package com.example.helponroad.SQLiteDataBaseInterface;

import android.database.Cursor;
import android.database.CursorWrapper;
import com.example.helponroad.ObjectsOfModel.Driver;
import com.example.helponroad.ObjectsOfModel.Request;
import com.example.helponroad.ObjectsOfModel.Status;
import com.example.helponroad.JSONParsers.DateParser;
import com.google.android.gms.maps.model.LatLng;
import java.util.Date;
import static
com.example.helponroad.SQLiteDataBaseInterface.SQLiteDataBa
seSchema.*;
/***
 * Обертка для курсора, который извлекает данные о заявке
 */
public class RequestCursorWrapper extends CursorWrapper {
    public RequestCursorWrapper(Cursor cursor) {
        super(cursor);
    }

    //Заполняет объект заявка
    public Request getRequest(){
        int id=getInt(getColumnIndex(RequestsTable.Cols.ID));
        String
message=getString(getColumnIndex(RequestsTable.Cols.MESSA
GE));
        Double
longitude=getDouble(getColumnIndex(RequestsTable.Cols.LONG
ITUDE));
        Double
latitude=getDouble(getColumnIndex(RequestsTable.Cols.LATITU
DE));
        Status status=new Status();

        status.setStatusValue(getInt(getColumnIndex(RequestsTable.Cols.
STATUS)));
        boolean
isImage=getInt(getColumnIndex(RequestsTable.NAME+"_"+Req
uestsTable.Cols.IS_IMAGE))==1;
        Date requestCreateDate=
DateParser.parseStringToDate(getString(getColumnIndex(Request
sTable.NAME+"_"+RequestsTable.Cols.CREATE_DATE)));
        String
StringRequestUpdateDate=getString(getColumnIndex(RequestsTa
ble.NAME+"_"+RequestsTable.Cols.UPDATE_DATE));
        Date requestUpdateDate=null;
        if(stringRequestUpdateDate!=null) {
            requestUpdateDate =
DateParser.parseStringToDate(stringRequestUpdateDate);
        }
        Date requestUploadDate=
DateParser.parseStringToDate(getString(getColumnIndex(Request
sTable.NAME+"_"+RequestsTable.Cols.UPLOAD_DATE)));
        boolean
isNew=getInt(getColumnIndex(RequestsTable.Cols.IS_NEW))==1;
        boolean
isGetNotify=getInt(getColumnIndex(RequestsTable.Cols.IS_GET
_NOTIFY))==1;
        //Заполняем данные о водителе
        Driver driver=new Driver();

        driver.setId(getString(getColumnIndex(RequestsTable.Cols.USER
_ID)));
        driver.setName(getString(getColumnIndex(UsersTable.Cols.NAM
E)));
        driver.setImage(getInt(getColumnIndex(UsersTable.NAME+"_"
+UsersTable.Cols.IS_IMAGE))==1);
        Date
driverCreateDate=DateParser.parseStringToDate(getString(getCol
umnIndex(UsersTable.NAME+"_"+UsersTable.Cols.CREATE_DATE)));
        driver.setCreateDate(driverCreateDate);
        String
stringDriverUpdateDate=getString(getColumnIndex(UsersTable.N
AME+"_"+UsersTable.Cols.UPDATE_DATE));
        if(stringDriverUpdateDate!=null) {

            driver.setUpdateDate(DateParser.parseStringToDate(stringDriverU
pdateDate));
        }
    }
}

```

Рис.П2.15. Текст модуля RequestCursorWrapper.java

## П2.15. Продолжение

```

Date
driverUploadDate=DateParser.parseStringToDate(getString(getCol
umnIndex(UsersTable.NAME+"_"+UsersTable.Cols.UPLOAD_DATE)));
    driver.setUploadDate(driverUploadDate);
//Заполняем данные о заявке
Request request=new Request();
request.setId(id);
request.setMessage(message);
request.setLocation(new LatLng(latitude,longitude));

```

```

request.setStatus(status);
request.setIsImage(isImage);
request.setCreateDate(requestCreateDate);
request.setUpdateDate(requestUpdateDate);
request.setUploadDate(requestUploadDate);
request.setDriver(driver);
request.setNew(isNew);
request.setGetNotify(isGetNotify);
return request;
}

```

### Модуль SQLiteDataBaseSchema.java

```

package com.example.helponroad.SQLiteDataBaseInterface;

/**
 * Схема локальной базы данных SQLite
 */
public class SQLiteDataBaseSchema {
    /**Структура таблицы для хранения данных пользователей
    (водителей)*/
    public static final class UsersTable {
        public static final String NAME = "users";
        public static final class Cols {
            public static final String ID= "id";//первичный ключ
            (системное поле)
            public static final String NAME = "name";//имя
            public static final String IS_IMAGE =
            "is_image";//наличие изображения
            public static final String IMAGE = "image";//изображение
            (NULL)
            public static final String
CREATE_DATE="create_date";//дата создания на сервере
            public static final String
UPDATE_DATE="update_date";//дата обновления на сервере
            public static final String
UPLOAD_DATE="upload_date";//дата загрузки на устройство
        }
    }

    /**Структура таблицы для хранения данных заявок о
    помощи*/
    public static final class RequestsTable {
        public static final String NAME = "requests";
        public static final class Cols {
            public static final String ID = "id";//первичный ключ
            public static final String MESSAGE =
            "message";//сообщение
            public static final String LONGITUDE =
            "longitude";//местоположение:широта
            public static final String LATITUDE =
            "latitude";//местоположение:долгота
            public static final String IS_IMAGE =
            "is_image";//наличие изображения
            public static final String IMAGE = "image";//ссылка на
            изображение (NULL)
            public static final String

```

```

CREATE_DATE="create_date";//дата создания на сервере
            public static final String
UPDATE_DATE="update_date";//дата изменения на
сервере(NULL)
            public static final String
UPLOAD_DATE="upload_date";//дата загрузки на устройство
            public static final String STATUS =
            "status";//идентификатор статуса
            public static final String IS_NEW="is_new";//является ли
            заявка новой
            public static final String
IS_GET_NOTIFY="is_get_notify";//получено ли уведомление
            для заявки
            public static final String USER_ID =
            "id_user";//идентификатор пользователя
        }
    }

    /**Структура таблицы для хранения данных предложений
    помощи*/
    public static final class OffersTable {
        public static final String NAME = "offers";
        public static final class Cols {
            public static final String ID = "id";//первичный ключ
            (системное поле)
            public static final String MESSAGE =
            "message";//сообщение
            public static final String CREATE_DATE =
            "create_date";//дата создания на сервере
            public static final String UPDATE_DATE =
            "update_date";//дата изменения на сервере (NULL)
            public static final String
UPLOAD_DATE="upload_date";//дата загрузки на устройство
            public static final String IS_NEW="is_new";//является ли
            предложение новым
            public static final String
IS_GET_NOTIFY="is_get_notify";//получено ли уведомление
            для предложения
            public static final String REQUEST_ID =
            "id_request";//идентификатор заявки
            public static final String USER_ID =
            "id_user";//идентификатор пользователя
        }
    }
}

```

Рис.П2.16. Текст модуля SQLiteDataBaseSchema.java

## Модуль RequestQuery.java

```

package com.example.helponroad.SQLiteDataBaseInterface;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import com.example.helponroad.ObjectsOfModel.Driver;
import com.example.helponroad.ObjectsOfModel.Request;
import com.example.helponroad.JSONParsers.DateParser;
import java.util.Date;
import static
com.example.helponroad.SQLiteDataBaseInterface.SQLiteDataBase
seSchema.*;

/**
 * Запросы для работы с таблицей Requests
 * Created by Oleg on 29.04.2017.
 */
public class RequestQuery {

    private static final String TAG = "RequestQuery";
    private HelpOnRoadBase mHelpOnRoadBase;
    private SQLiteDatabase mDatabase;
    private Context mContext;

    public RequestQuery(Context context) {
        mContext = context;
        mHelpOnRoadBase = new HelpOnRoadBase(mContext);
    }

    public void openConnection() {
        mDatabase = mHelpOnRoadBase.getWritableDatabase();
    }

    public void closeConnection() {
        if (mDatabase != null && mDatabase.isOpen()) {
            mDatabase.close();
        }
    }

    public boolean checkConnection() {
        return mDatabase != null && mDatabase.isOpen();
    }

    public static ContentValues getRequestValues(Request request,
        boolean isId, boolean isMessage, boolean isLatitude, boolean
        isLongitude, boolean isIsImage, Boolean isImage, boolean
        isCreateDate, boolean isUpdateDate,
        boolean isUploadDate, boolean isStatus, boolean isUserId, boolean
        isNew,boolean isGetNotify) {
        ContentValues values = new ContentValues();
        if (isId) values.put(RequestsTable.Cols.ID, request.getId());
        if (isMessage) values.put(RequestsTable.Cols.MESSAGE,
            request.getMessage());
        if (isLatitude) values.put(RequestsTable.Cols.LATITUDE,
            request.getLocation().latitude);
        if (isLongitude)
            values.put(RequestsTable.Cols.LONGITUDE,
            request.getLocation().longitude);
        if (isIsImage) values.put(RequestsTable.Cols.IS_IMAGE,
            request.getIsImage()?!:0);
        if (isImage) values.put(RequestsTable.Cols.IMAGE,
            request.getImage());
        if (isCreateDate)
            values.put(RequestsTable.Cols.CREATE_DATE,
                DateParser.parseDateToString(request.getCreateDate()));
        if (isUpdateDate) {
            if (request.getUpdateDate() != null) {
                values.put(RequestsTable.Cols.UPDATE_DATE,
                    dateParser.parseDateToString(request.getUpdateDate()));
            }
        }
        if (isUploadDate)
            values.put(RequestsTable.Cols.UPLOAD_DATE,
                DateParser.parseDateToString(request.getUploadDate()));
        if (isStatus) values.put(RequestsTable.Cols.STATUS,
            request.getStatus().getStatusValue());
        if (isUserId) values.put(RequestsTable.Cols.USER_ID,
            request.getDriver().getId());
        if (isNew) values.put(RequestsTable.Cols.IS_NEW,
            request.isNew()?!:0);
        if (isGetNotify)
            values.put(RequestsTable.Cols.IS_GET_NOTIFY,
            request.isGetNotify()?!:0);
        return values;
    }

    /**
     * Добавляем данные заявки в базу
     *
     * @param requestContentValue-добавляемые данные (имя
     * поля, значение поля записи)
     */
    public void addRequest(ContentValues requestContentValue) {
        if (checkConnection()) {
            ContentValues values = requestContentValue;
            mDatabase.insert(RequestsTable.NAME, null, values);
        } else{
            Log.e(TAG,"Требуется открыть подключение к базе
данных");
            throw new SQLException("Требуется открыть
подключение к базе данных");
        }
    }

    /**
     * Удаляем неактуальные заявки из базы
     */
    public void deleteNotActualRequest(){
        if (checkConnection()) {
            String whereClauses = "(strftime('%s',?) - strftime('%s', " +
RequestsTable.Cols.CREATE_DATE + " )) > 86400";
            String currentDate = DateParser.parseDateToString(new
Date());
            int numberRows=mDatabase.delete(RequestsTable.NAME,
whereClauses, new String[]{currentDate});
            Log.i(TAG,"Удалено "+numberRows+" заявок");
            //Удаляем пользователей неактуальных заявок
        } else{
            Log.e(TAG,"Требуется открыть подключение к базе
данных");
            throw new SQLException("Требуется открыть
подключение к базе данных");
        }
    }

    /**
     * Обновляем данные заявки в базе
     *
     * @param requestContentValue-обновляемые данные (имя
     * поля, значение поля записи)
     */
    public void updateRequest(ContentValues requestContentValue) {
        if (checkConnection()) {
            String idString = requestContentValue.getAsString("id");
            values.put(RequestsTable.Cols.ID, idString);
            mDatabase.update(RequestsTable.NAME, values,
            RequestsTable.Cols.ID + " = ?",
            new String[]{idString});
        }
    }
}

```

Рис.П2.17. Текст модуля RequestQuery.java

## П2.17. Продолжение

## П2.17. Продолжение

```

        UsersTable.NAME + "." + UsersTable.Cols.ID + " AS "
    " + RequestsTable.Cols.USER_ID + "," +
        UsersTable.NAME + "." + UsersTable.Cols.NAME +
    " AS " + UsersTable.Cols.NAME + "," +
        UsersTable.NAME + "." +
    UsersTable.Cols.IS_IMAGE + " AS " + UsersTable.NAME + "_"
    + UsersTable.Cols.IS_IMAGE + "," +
        UsersTable.NAME + "." +
    UsersTable.Cols.CREATE_DATE + " AS "
    +UsersTable.NAME+"_" + UsersTable.Cols.CREATE_DATE +",
    "+

        UsersTable.NAME + "." +
    UsersTable.Cols.UPDATE_DATE + " AS "
    +UsersTable.NAME+"_" + UsersTable.Cols.UPDATE_DATE +",
    "+

        UsersTable.NAME + "." +
    UsersTable.Cols.UPLOAD_DATE + " AS "
    +"UsersTable.NAME+"_" + UsersTable.Cols.UPLOAD_DATE +
    "+

        "FROM " + RequestsTable.NAME + ", "
    UsersTable.NAME + " "
    "WHERE " + RequestsTable.NAME + "."
    RequestsTable.Cols.USER_ID + "=" + UsersTable.NAME + "."
    +UsersTable.Cols.ID+" "
    "AND ("+RequestsTable.NAME + "."
    RequestsTable.Cols.USER_ID + "<>" + driver.getId() + " OR
    "+driver.getId() + " IS NULL )"
    "AND (strftime('%s','+
DateParser.parseDateToString(new Date())")-
strftime('%s',"+RequestsTable.NAME + "."
    RequestsTable.Cols.CREATE_DATE +"))<=86400 "
    "AND "+RequestsTable.NAME + "."
    RequestsTable.Cols.IS_NEW + "=1 "
    "AND "+RequestsTable.NAME + "."
    RequestsTable.Cols.IS_GET_NOTIFY + "=0 "
    "AND "+RequestsTable.NAME + "."
    RequestsTable.Cols.STATUS + "<>3";//Не выбираем заявки для
которых оказана помощь

    Cursor cursor = mDatabase.rawQuery(query, new
String[]{});

    Log.i(TAG, "Запрос: " + query);
    return new RequestCursorWrapper(cursor);
} else{
    Log.e(TAG,"Требуется открыть подключение к базе
данных");
    throw new SQLException("Требуется открыть
подключение к базе данных");
}

/**
 * Получаем из базы новые заявки, которые не добавлены
текущим пользователем приложения
 * @param driver - пользователь приложения
 * @return - курсор, который содержит сведения о новых
заявках
 */
public RequestCursorWrapper getRequestsAppUser(Driver
driver) {
    if (checkConnection()){
        String query = "SELECT " + RequestsTable.NAME + "."
    + RequestsTable.Cols.ID + " AS " + RequestsTable.Cols.ID + ", "
    + RequestsTable.NAME + "."
    RequestsTable.Cols.MESSAGE + " AS " +
    RequestsTable.Cols.MESSAGE + ", "
    + RequestsTable.NAME + "."
    RequestsTable.Cols.LATITUDE + " AS " +
    RequestsTable.Cols.LATITUDE + ", "
    + RequestsTable.NAME + "."
    RequestsTable.Cols.LONGITUDE + " AS " +
    RequestsTable.Cols.LONGITUDE + ", "
    + RequestsTable.NAME + "."
    RequestsTable.Cols.STATUS + " AS "
    + RequestsTable.Cols.STATUS + ", "
    + RequestsTable.Cols.STATUS + ", "
    + RequestsTable.Cols.NAME + "."
    RequestsTable.Cols.IS_IMAGE + " AS " +
    RequestsTable.Cols.IS_IMAGE + ", "
    + RequestsTable.NAME + "."
    RequestsTable.Cols.UPDATE_DATE + " AS "
    +RequestsTable.Cols.UPDATE_DATE + ", "
    + RequestsTable.Cols.CREATE_DATE + " AS "
    +RequestsTable.Cols.CREATE_DATE + ", "
    + RequestsTable.Cols.IS_NEW + " AS "
    + RequestsTable.Cols.IS_NEW + ", "
    + RequestsTable.NAME + "."
    RequestsTable.Cols.IS_GET_NOTIFY + " AS "
    + RequestsTable.Cols.IS_GET_NOTIFY + ", "
    + UsersTable.NAME + "."
    UsersTable.Cols.USER_ID + ", "
    + UsersTable.NAME + "."
    UsersTable.Cols.NAME + "."
    UsersTable.Cols.IS_IMAGE + " AS " + UsersTable.NAME + "_"
    + UsersTable.Cols.IS_IMAGE + ", "
    + UsersTable.NAME + "."
    UsersTable.Cols.CREATE_DATE + " AS "
    +UsersTable.NAME+"_" + UsersTable.Cols.CREATE_DATE +",
    "+

        UsersTable.NAME + "."
    UsersTable.Cols.UPDATE_DATE + " AS "
    +UsersTable.NAME+"_" + UsersTable.Cols.UPDATE_DATE +",
    "+

        UsersTable.NAME + "."
    UsersTable.Cols.UPLOAD_DATE + " AS "
    +"UsersTable.NAME+"_" + UsersTable.Cols.UPLOAD_DATE +
    "+

        "FROM " + RequestsTable.NAME + ", "
    UsersTable.NAME + " "
    "WHERE " + RequestsTable.NAME + "."
    RequestsTable.Cols.USER_ID + "=" + UsersTable.NAME + "."
    +UsersTable.Cols.ID+" "
    "AND "+RequestsTable.NAME + "."
    RequestsTable.Cols.USER_ID + "=" + driver.getId()+
    " ORDER BY "+RequestsTable.Cols.ID+ " DESC";
    Cursor cursor = mDatabase.rawQuery(query, new
String[]{});

    Log.i(TAG, "Запрос: " + query);
    return new RequestCursorWrapper(cursor);
} else{
    Log.e(TAG,"Требуется открыть подключение к базе
данных");
    throw new SQLException("Требуется открыть
подключение к базе данных");
}

/**
 * Получение сведений о заявке с идентификатором id из
базы из таблиц Requests, Users
 * @param requestId - идентификатор заявки
 * @return - курсор RequestCursorWrapper, которые
содержит данные заявки
 */
public RequestCursorWrapper getRequest(int requestId){
    if (checkConnection()){
        String query = "SELECT " + RequestsTable.NAME + "."
    + RequestsTable.Cols.ID + " AS " + RequestsTable.Cols.ID + ", "
    + RequestsTable.NAME + "."
    RequestsTable.Cols.MESSAGE + " AS " +
    RequestsTable.Cols.MESSAGE + ", "
    + RequestsTable.NAME + "."
    RequestsTable.Cols.LATITUDE + " AS " +
    RequestsTable.Cols.LATITUDE + ", "
    + RequestsTable.NAME + "."
    RequestsTable.Cols.LONGITUDE + " AS " +
    RequestsTable.Cols.LONGITUDE + ", "
    + RequestsTable.NAME + "."
    RequestsTable.Cols.STATUS + " AS "
    + RequestsTable.Cols.STATUS + ", "
    + RequestsTable.Cols.STATUS + ", "
    + RequestsTable.Cols.NAME + "."
    RequestsTable.Cols.IS_IMAGE + " AS " +
    RequestsTable.Cols.IS_IMAGE + ", "
    + RequestsTable.NAME + "."
    RequestsTable.Cols.UPDATE_DATE + " AS "
    +RequestsTable.Cols.UPDATE_DATE + ", "
    + RequestsTable.Cols.CREATE_DATE + " AS "
    +RequestsTable.Cols.CREATE_DATE + ", "
    + RequestsTable.Cols.IS_NEW + " AS "
    + RequestsTable.Cols.IS_NEW + ", "
    + RequestsTable.NAME + "."
    RequestsTable.Cols.IS_GET_NOTIFY + " AS "
    + RequestsTable.Cols.IS_GET_NOTIFY + ", "
    + UsersTable.NAME + "."
    UsersTable.Cols.USER_ID + ", "
    + UsersTable.NAME + "."
    UsersTable.Cols.NAME + "."
    UsersTable.Cols.IS_IMAGE + " AS " + UsersTable.NAME + "_"
    + UsersTable.Cols.IS_IMAGE + ", "
    + UsersTable.NAME + "."
    UsersTable.Cols.CREATE_DATE + " AS "
    +UsersTable.NAME+"_" + UsersTable.Cols.CREATE_DATE +",
    "+

        UsersTable.NAME + "."
    UsersTable.Cols.UPDATE_DATE + " AS "
    +UsersTable.NAME+"_" + UsersTable.Cols.UPDATE_DATE +",
    "+

        UsersTable.NAME + "."
    UsersTable.Cols.UPLOAD_DATE + " AS "
    +"UsersTable.NAME+"_" + UsersTable.Cols.UPLOAD_DATE +
    "+

        "FROM " + RequestsTable.NAME + ", "
    UsersTable.NAME + " "
    "WHERE " + RequestsTable.NAME + "."
    RequestsTable.Cols.USER_ID + "=" + UsersTable.NAME + "."
    +UsersTable.Cols.ID+" "
    "AND "+RequestsTable.NAME + "."
    RequestsTable.Cols.USER_ID + "=" + driver.getId()+
    " ORDER BY "+RequestsTable.Cols.ID+ " DESC";
    Cursor cursor = mDatabase.rawQuery(query, new
String[]{});

    Log.i(TAG, "Запрос: " + query);
    return new RequestCursorWrapper(cursor);
}

```

## П2.17. Продолжение

```

RequestsTable.Cols.LATITUDE + ", " +
    RequestsTable.NAME + ". " +
RequestsTable.Cols.LONGITUDE + " AS " +
RequestsTable.Cols.LONGITUDE + ", " +
    RequestsTable.NAME + ". " +
RequestsTable.Cols.STATUS + " AS " +
RequestsTable.Cols.STATUS + ", " +
    RequestsTable.NAME + ". " +
RequestsTable.Cols.IS_IMAGE + " AS " + RequestsTable.NAME
+ "_" + RequestsTable.Cols.IS_IMAGE + ", " +
    RequestsTable.NAME + ". " +
RequestsTable.Cols.CREATE_DATE + " AS "
+"+RequestsTable.NAME+_+
RequestsTable.Cols.CREATE_DATE + ", " +
    RequestsTable.NAME + ". " +
RequestsTable.Cols.UPDATE_DATE + " AS "
+"+RequestsTable.NAME+_+
RequestsTable.Cols.UPDATE_DATE + ", " +
    RequestsTable.NAME + ". " +
RequestsTable.Cols.UPLOAD_DATE + " AS "
+"+RequestsTable.NAME+_+
RequestsTable.Cols.UPLOAD_DATE + ", " +
    RequestsTable.NAME + ". " +
RequestsTable.Cols.IS_NEW + " AS " +
RequestsTable.Cols.IS_NEW + ", " +
    RequestsTable.NAME + ". " +
RequestsTable.Cols.IS_GET_NOTIFY + " AS " +
RequestsTable.Cols.IS_GET_NOTIFY + ", " +
    UsersTable.NAME + ". " + UsersTable.Cols.ID + " AS "
" + RequestsTable.Cols.USER_ID + ", " +
    UsersTable.NAME + ". " + UsersTable.Cols.NAME +
" AS " + UsersTable.Cols.NAME + ", " +
    UsersTable.NAME + ". "

```

```

UsersTable.Cols.IS_IMAGE + " AS " + UsersTable.NAME + "_"
+ UsersTable.Cols.IS_IMAGE + ", " +
    UsersTable.NAME + ". " +
UsersTable.Cols.CREATE_DATE + " AS "
+UsersTable.NAME+_ + UsersTable.Cols.CREATE_DATE + ",
" +
    UsersTable.NAME + ". " +
UsersTable.Cols.UPDATE_DATE + " AS "
+UsersTable.NAME+_ + UsersTable.Cols.UPDATE_DATE +",
" +
    UsersTable.NAME + ". " +
UsersTable.Cols.UPLOAD_DATE + " AS "
+UsersTable.NAME+_ + UsersTable.Cols.UPLOAD_DATE +
" " +
    "FROM " + RequestsTable.NAME + ", " +
UsersTable.NAME + " " +
"WHERE " + RequestsTable.NAME + ". " +
RequestsTable.Cols.USER_ID + "=" + UsersTable.NAME + ". " +
UsersTable.Cols.ID+
    " AND
"+RequestsTable.NAME+". "+RequestsTable.Cols.ID+"=?";

```

```

        Cursor cursor = mDatabase.rawQuery(query, new
String[] { Integer.toString(requestId) });
        Log.i(TAG, "Запрос: " + query);
        return new RequestCursorWrapper(cursor);
    } else {
        Log.e(TAG, "Требуется открыть подключение к базе
данных");
        throw new SQLException("Требуется открыть
подключение к базе данных");
    }
}

```

## Модуль UserQuery.java

```

package com.example.helponroad.SQLiteDataBaseInterface;

import android.content.ContentValues;
import android.content.Context;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;
import com.example.helponroad.ObjectsOfModel.Driver;
import com.example.helponroad.JSONParsers.DateParser;
import static
com.example.helponroad.SQLiteDataBaseInterface.SQLiteDataBa
seSchema.*;

```

```

/**
 * Запросы к таблице о пользователях
 */

```

```

public class UserQuery {

    private static final String TAG="UserQuery";
    private HelpOnRoadBase mHelpOnRoadBase;
    private SQLiteDatabase mDatabase;
    private Context mContext;

    public UserQuery(Context context) {
        mContext = context;
        mHelpOnRoadBase = new HelpOnRoadBase(mContext);
    }

    public void openConnection() {
        mDatabase = mHelpOnRoadBase.getWritableDatabase();
    }

    public void closeConnection() {
        if (mDatabase != null && mDatabase.isOpen()) {
            mDatabase.close();
        }
    }

    public boolean checkConnection() {
        return mDatabase != null && mDatabase.isOpen();
    }

    public static ContentValues getUserValues(Driver driver,
        boolean isId, boolean isName, boolean isImage, boolean
        isImage,boolean isCreateDate, boolean isUpdateDate, boolean
        isUploadDate){
        ContentValues values=new ContentValues();
        if(isId)values.put(UsersTable.Cols.ID,driver.getId());
        if(isName)values.put(UsersTable.Cols.NAME,driver.getName());
        if(isImage)values.put(UsersTable.Cols.IS_IMAGE,driver.getIsI
        mage()?1:0);
        if(isImage)values.put(UsersTable.Cols.IMAGE,driver.getImage());
        if(isCreateDate)values.put(UsersTable.Cols.CREATE_DATE,
        DateParser.parseDateToString(driver.getCreateDate()));
        if(isUpdateDate){
            if(driver.getUpdateDate()!=null) {
                values.put(UsersTable.Cols.UPDATE_DATE,

```

Рис.П2.18. Текст модуля UserQuery.java

## П2.18. Продолжение

```

DateParser.parseDateToString(driver.getUpdateDate()));
        }
    }
    if(isUploadDate)values.put(UsersTable.Cols.UPLOAD_DATE,
DateParser.parseDateToString(driver.getUploadDate()));
    return values;
}

/**
 * Добавляем данные пользователя в базу
 * @param userContentValue-добавляемые данные (имя поля,
значение поля записи)
 */
public void addUser(ContentValues userContentValue){
    if(checkConnection()) {
        ContentValues values = userContentValue;
        mDatabase.insert(UsersTable.NAME, null, values);
    }else{
        Log.e(TAG,"Требуется открыть подключение к базе
данных");
        throw new SQLException("Требуется открыть
подключение к базе данных");
    }
}

/**
 * Обновляем данные пользователя в базе
 * @param userContentValue-обновляемые данные (имя поля,
значение поля записи)
 */
public void updateUser(ContentValues userContentValue) {
    if (checkConnection()){
        String idString = userContentValue.getAsString("id");
        ContentValues values = userContentValue;
        mDatabase.update(UsersTable.NAME, values,
RequestsTable.Cols.ID + "=?", new String[]{idString});
    }else{
        Log.e(TAG,"Требуется открыть подключение к базе
данных");
        throw new SQLException("Требуется открыть
подключение к базе данных");
    }
}

/**
 * Добавляем или обновляем данные пользователя
 * @param userContentValue-добавляемые данные
пользователя (имя поля записи, значение поля записи)
 */

```

---

```

    */
    public void addOrUpdateUser(ContentValues
userContentValue) {
        if (checkConnection()){
            int id = (int)
mDatabase.insertWithOnConflict(UsersTable.NAME, null,
userContentValue, SQLiteDatabase.CONFLICT_IGNORE);
            if (id == -1) {
                this.updateUser(userContentValue);
            }
        }else{
            Log.e(TAG,"Требуется открыть подключение к базе
данных");
            throw new SQLException("Требуется открыть
подключение к базе данных");
        }
    }

    /**
     * Удаляем пользователей неактуальных заявок и которые
не добавили предложений к актуальным заявкам
     */
    public void deleteUserNotActualRequestAndOffers(){
        if (checkConnection()) {
            String whereClauses = "NOT EXISTS (SELECT * FROM
"+RequestsTable.NAME+
                    " WHERE
"+UsersTable.NAME+"."+UsersTable.Cols.ID+"="+RequestsTabl
e.NAME+"."+RequestsTable.Cols.USER_ID+"")+
                    " AND NOT EXISTS (SELECT * FROM
"+OffersTable.NAME+
                    " WHERE
"+UsersTable.NAME+"."+UsersTable.Cols.ID+"="+OffersTable.
NAME+"."+OffersTable.Cols.USER_ID+"");
            int numberRows=mDatabase.delete(UsersTable.NAME,
whereClauses, new String[]{} );
            Log.i(TAG,"Удалено "+numberRows+" пользователей");
            //Удаляем пользователей неактуальных заявок
        }else{
            Log.e(TAG,"Требуется открыть подключение к базе
данных");
            throw new SQLException("Требуется открыть
подключение к базе данных");
        }
    }
}

```

## Пакет JSONParsers

## Модуль DateParser.java

```
package com.example.helponroad.JSONParsers;
import android.util.Log;
import java.util.Calendar;
import java.util.Date;
import java.util.TimeZone;
import java.text.SimpleDateFormat;
/***
 * Класс для преобразования дат
 */
public class DateParser {
    private static final String TAG="DateParser";
    private static final String DATE_FORMAT_STRING="yyyy-MM-dd HH:mm:ss";
```

```
private static final String TIME_ZONE="GMT+3:00";
/***
 * Преобразует значение даты в строковое представление
 */
public static String parseDateToString(Date date){
    SimpleDateFormat dateFormat = new
SimpleDateFormat(DATE_FORMAT_STRING);

dateFormat.setTimeZone(TimeZone.getTimeZone(TIME_ZONE))
;
    return dateFormat.format(date);
}
```

Рис.П2.19. Текст модуля DateParser.java

## П2.19. Продолжение

```

        */
        public static void setTimeZone(){
            TimeZone tz=TimeZone.getTimeZone(TIME_ZONE);
            TimeZone.setDefault(tz);
            Log.i(TAG,tz.toString());
        }

        /**
         * Вычисляет разницу между датами endDate, startDate в
         * формате час. мин.
         * @param endDate
         * @param startDate
         * @return
         */
        public static String getDateSub(Date endDate, Date startDate){
            Calendar
            diff=Calendar.getInstance(TimeZone.getTimeZone("GMT"));
            diff.setTimeInMillis(endDate.getTime()-startDate.getTime());
            return diff.get(Calendar.HOUR_OF_DAY)+" час.
            "+diff.get(Calendar.MINUTE)+" мин. ";
        }
    }

    /**
     * Устанавливает временную зону по московскому времени

```

### Модуль OfferParser.java

```

import com.example.helponroad.ObjectsOfModel.Driver;
import com.example.helponroad.ObjectsOfModel.Offer;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 * Класс для распарсивания предложений помощи
 */
public class OfferParser extends ReplyParser {

    public OfferParser(byte[] replay) {
        super(replay);
    }

    /**
     * Возвращает распарсенные данные предложений заявок
     * @return - объект заявка
     */
    public List<Offer> getParsedOffers()throws JSONException {
        JSONObject jsonObject = new JSONObject(mStringReplay);
        JSONArray
        offersJsonObject=jsonObject.getJSONArray("offers");
        List<Offer> offers=new ArrayList<>();
        for(int i=0;i<offersJsonObject.length();i++){
            JSONObject
            offerJSONObject=offersJsonObject.getJSONObject(i);

```

```

Offer offerItem=new Offer();
offerItem.setId(offerJSONObject.getInt("id"));

offerItem.setMessage(offerJSONObject.getString("message"));

offerItem.setRequestId(offerJSONObject.getInt("id_request"));

Driver offerDriver=new Driver();
offerDriver.setId(offerJSONObject.getString("id_user"));
offerItem.setDriver(offerDriver);

offerItem.setCreateDate(DateParser.parseStringToDate(offerJSONObject.getString("create_date")));

String
offerUpdateDate=offerJSONObject.getString("update_date");
if(offerUpdateDate.equals("null")){
    offerItem.setUpdateDate(null);
}else{
    offerItem.setUpdateDate(DateParser.parseStringToDate(offerUpdateDate));
}

offerItem.setUploadDate(new Date());
offers.add(offerItem);
}
return offers;
}
}

```

Рис.П2.20. Текст модуля OfferParser.java

## Модуль RequestParser.java

```

package com.example.helponroad.JSONParsers;
import com.example.helponroad.ObjectsOfModel.Driver;
import com.example.helponroad.ObjectsOfModel.Request;
import com.example.helponroad.ObjectsOfModel.Status;
import com.google.android.gms.maps.model.LatLng;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 * Класс для разбора данных о заявке в формате JSON
 */
public class RequestParser extends ReplyParser {

    public RequestParser(byte[] replay) {
        super(replay);
    }
    /**
     * Возвращает распарсенные данные заявок
     */
    public List<Request> getParsedRequests() throws
    JSONException {
        JSONObject jsonObject = new JSONObject(mStringReplay);
        JSONArray
        requestsJsonObject=jsonObject.getJSONArray("requests");
        List<Request> requests=new ArrayList<>();
        for(int i=0;i<requestsJsonObject.length();i++){
            JSONObject
            requestJsonObject=requestsJsonObject.getJSONObject(i);
            Request requestItem=new Request();
            requestItem.setId(requestJsonObject.getInt("id"));
            requestItem.set

```

requestItem.setMessage(requestJSONObject.getString("message"));  
`; LatLng requestLocation=new LatLng(requestJSONObject.getDouble("latitude"), requestJSONObject.getDouble("longitude")); requestItem.setLocation(requestLocation);`  
`requestItem.setIsImage(requestJSONObject.getInt("image")==1); Status requestStatus=new Status();`  
`requestStatus.setStatusValue(requestJSONObject.getInt("id_status ")); requestItem.setStatus(requestStatus); Driver requestDriver=new Driver();`  
`requestDriver.setId(requestJSONObject.getString("id_user")); requestItem.setDriver(requestDriver);`  
`requestItem.setCreateDate(DateParser.parseStringToDate(requestJSONObject.getString("create_date"))); String requestUpdateDate=requestJSONObject.getString("update_date"); if(requestUpdateDate.equals("null")){ requestItem.setUpdateDate(null); }else{ requestItem.setUpdateDate(DateParser.parseStringToDate(requestUpdateDate)); }`  
`requestItem.setUploadDate(new Date()); requests.add(requestItem); }`  
`return requests;`

Рис.П2.21. Текст модуля RequestParser.java

## Модуль RouteParser.java

```

package com.example.helponroad.JSONParsers;
import android.util.Log;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

/**
 * Класс для разбора данных о маршруте между двумя
 * точками
 */
public class RouteParser {

    private static final String TAG="RouteParser";
    private String mPoints;//массив закодированных точек,
    представляющий сглаженный маршрут между начальной и
    конечной точками
    public String getPoints() {
        return mPoints;
    }
    /**
     * Парсит ответ от сервера Google  маршруте между двумя
     * точками
     */
    public void parseReplyAboutRoute(String replay){
        try {
            JSONObject replayJSONObject = new
            JSONObject(replay);
            //Парсим начальный и конечный адрес
            //startAddress=replayJSONObject.getString("start_address");
            //endAddress=replayJSONObject.getString("end_address");
            //Парсим маршрут
            JSONArray
            routesJsonArray=replayJSONObject.getJSONArray("routes");
            //массив маршрутов
            if(routesJsonArray.length()!=0) {
                JSONObject routeJSONObject =
                routesJsonArray.getJSONObject(0);
                JSONObject polylineJSONObject=routeJSONObject.getJSONObject("overview
                _polyline");//сглаженная линия маршрута
                mPoints=polylineJSONObject.getString("points");//закодированн
                ый массив точек маршрута
            }
        } catch (JSONException ex){
            Log.e(TAG,"Не удалось построить
            маршрут:"+ex.getMessage());
        }
    }
}

```

Рис.П2.22. Текст модуля RouteParser.java

## П2.22. Продолжение

```

    /**
     * Парсит данные об адресе по текущему местоположению
     * @param replay - строка ответа от сервера
     * @return - данные об адресе
     */
    public static String addressParser(String replay){
        String formattedAddress=null;
        try {
            JSONObject replayJSONObject = new
            JSONObject(replay);
            JSONArray
            resultJSONArray=replayJSONObject.getJSONArray("results");
            if(resultJSONArray.length()!=0) {
                JSONObject
                resultItemJSONObject=resultJSONArray.getJSONObject(0);
                formattedAddress=resultItemJSONObject.getString("formatted_ad-
                dress");
            }
        } catch (JSONException ex){
            Log.e(TAG,"Не удалось получить
            адрес:"+ex.getMessage());
        }
        return formattedAddress;
    }
}

```

### Модуль ReplayParser.java

```

package com.example.helponroad.JSONParsers;

import org.json.JSONException;
import org.json.JSONObject;

/**
 * Разбирает статус запроса, который пришел с сервера
 * Created by Олег on 15.04.2017.
 */
public class ReplyParser {

    private static final String TAG="ReplyParser";
    protected byte[] mByteReplay;
    protected String mStringReplay;

    public ReplyParser(byte[] replay) {
        mByteReplay = replay;
        mStringReplay=new String(replay);
    }

    /**
     * Возвращает строковое представление разбираемого
     * потока байтов
     */
    public String stringReplay(){
        return mStringReplay;
    }

    /**
     * Возвращает разбираемый поток байтов
     */
    public byte[] byteReplay(){
        return mByteReplay;
    }

    /**
     * Возвращает статус запроса на сервер
     */
    public boolean getReplaySuccessField()throws JSONException{
        JSONObject jsonObject = new JSONObject(mStringReplay);
        int success=jsonObject.getInt("success");
        return success==1;
    }

    /**
     * Возвращает сообщение от сервера
     */
    public String getReplayMessageField()throws JSONException{
        JSONObject jsonObject = new JSONObject(mStringReplay);
        return jsonObject.getString("message");
    }

    /**
     * Возвращает код ошибки, если статус=1
     */
    public String getReplayErrorCodeField()throws JSONException{
        JSONObject jsonObject = new JSONObject(mStringReplay);
        if(jsonObject.getInt("success")==1){
            return jsonObject.getString("error_code");
        }else{
            return null;
        }
    }

    /**
     * Возвращает сообщение по статусу
     */
    public String getStatusMessage(int status){
        switch (status){
            case 0: return "Не указаны обязательные параметры
запроса";
            case 1: return "Неверное значение ключа";
            case 2: return "При добавлении заявки произошла
внутренняя ошибка";
            case 3: return "Пользователь с данным логином уже
существует";
            case 4: return "Не удалось пройти авторизацию.
Указаны неверные значения логина и пароля";
            case 5: return "Пользователь с данным
идентификатором не найден";
            case 6: return "Ошибка загрузки файла";
            default: return null;
        }
    }
}

```

Рис.П2.23. Текст модуля ReplayParser.java

## Модуль UserParser.java

```

package com.example.helponroad.JSONParsers;
import com.example.helponroad.ObjectsOfModel.Driver;
import com.example.helponroad.ObjectsOfModel.User;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 * Класс для заполнения объекта пользователь, данными,
 * которые пришли от сервера
 * Created by Oleg on 16.04.2017.
 */
public class UserParser extends ReplyParser {

    public UserParser(byte[] replay){
        super(replay);
    }

    /**
     * Возвращает распарсенные параметры пользователя
     * @return - объект пользователя
     * @throws JSONException
     */
    public User getParsedUser(boolean isId,boolean
isLogin,boolean isPassword, boolean isName,boolean
isImage,boolean isCreateDate,boolean isUpdateDate)throws
JSONException {
        JSONObject jsonObject = new JSONObject(mStringReplay);
        JSONObject jsonUser=jsonObject.getJSONObject("user");
        User user=new User();
        if(isId) user.setId(jsonUser.getString("id"));
        if(isLogin) user.setLogin(jsonUser.getString("login"));
        if(isPassword)
user.setPassword(jsonUser.getString("password"));
        if(isName) {
            if(!jsonUser.getString("name").equals("null")) {
                user.setName(jsonUser.getString("name"));
            }
        }
        if(isImage)
    }

    user.setImage(jsonUser.getBoolean("image"));
    if(isCreateDate)
user.setCreateDate(DateParser.parseStringToDate(jsonObject.getString("create_date")));
    if(isUpdateDate)user.setUpdateDate(DateParser.parseStringToDate(jsonObject.getString("update_date")));
    return user;
}

/**
 * Возвращает распарсенные данные о водителях в виде
 * списка водителей
 * @return
 * @throws JSONException
 */
public List<Driver> getParsedDrivers()throws JSONException{
    JSONObject jsonObject = new JSONObject(mStringReplay);
    JSONArray usersJSONObject=jsonObject.getJSONArray("users");
    List<Driver> drivers=new ArrayList<>();
    for(int i=0;i<usersJSONObject.length();i++){
        JSONObject
userJSONObject=usersJSONObject.getJSONObject(i);
        Driver driverItem=new Driver();
        driverItem.setId(userJSONObject.getString("id"));
        driverItem.setName(userJSONObject.getString("name"));
        driverItem.setImage(userJSONObject.getInt("image")==1);
        driverItem.setCreateDate(DateParser.parseStringToDate(userJSONObject.getString("create_date")));
        String
driverUpdateDate=userJSONObject.getString("update_date");
        if(driverUpdateDate.equals("null")){
            driverItem.setUpdateDate(null);
        }else{
            driverItem.setUpdateDate(DateParser.parseStringToDate(driverUpdateDate));
        }
        driverItem.setUploadDate(new Date());
        drivers.add(driverItem);
    }
    return drivers;
}
}

```

Рис.П2.24. Текст модуля UserParser.java

## Пакет ManageObjectsOfModel

### Модуль FileManager.java

```

package com.example.helponroad.ManageObjectsOfModel;
import java.io.File;
import java.io.IOException;
public class FileManager {

    String mFilePath;
    File mFile;

    public FileManager(String filePath) {
        mFilePath = filePath;
        mFile=new File(filePath);
    }
    public String getFilePath() {
        return mFilePath;
    }

    public File getFile(){
        return mFile;
    }
    public void removeFile()throws IOException {
        if (mFile.exists()) {
            String deleteCmd = "rm -r " + mFilePath;
            Runtime runtime = Runtime.getRuntime();
            runtime.exec(deleteCmd);
        } else {
            throw new IOException("Файл не существует");
        }
    }
}

```

Рис.П2.25. Текст модуля FileManager.java

## Модуль GoogleAPIManager.java

```

package com.example helponroad.ManageObjectsOfModel;
import android.net.Uri;
import android.util.Log;
import com.example helponroad.NetworkOperations.HttpsQuery;
import com.google.android.gms.maps.model.LatLng;
import java.io.IOException;

/**
 * Класс для получения данных для построения маршрута
 * Created by Олег on 04.05.2017.
 */
public class GoogleAPIManager {
    private static final String TAG="GoogleAPIManager";
    private static final String ROUTE_URL="https://maps.googleapis.com/maps/api/directions/json";
    private static final String GEOCODE_URL="https://maps.googleapis.com/maps/api/geocode/json";
    /**
     * Получает данные о маршруте между начальной и
     * конечной точками в формате JSON
     * @param originLoc - координаты начальной точки
     * @param destinationLoc - координаты конечной точки
     */
    public String getRoute(LatLng originLoc,LatLng destinationLoc){
        try {
            String url = Uri.parse(ROUTE_URL)
                .buildUpon()
.appendQueryParameter("origin",originLoc.latitude+","+originLoc.longitude)
.appendQueryParameter("destination",
destinationLoc.latitude+","+destinationLoc.longitude)
.appendQueryParameter("sensor", "true")//исходит ли
запрос с датчиком для определения местоположения
.appendQueryParameter("language", "ru")//язык
        }
    }

    /**
     * Полученные данные от сервера Google: "
     */
    public String getAddress(LatLng location){
        try {
            String url = Uri.parse(GEOCODE_URL)
                .buildUpon()
.appendQueryParameter("latlng",location.latitude+","+location.longitude)
.appendQueryParameter("sensor", "true")//исходит ли
запрос с датчиком для определения местоположения
.appendQueryParameter("language", "ru")//язык
                .build().toString();
            String jsonString = HttpsQuery.getUrlString(url);
            Log.i(TAG, "Полученные данные от сервера Google: " +
jsonString);
            return jsonString;
        } catch (IOException ioe) {
            Log.e(TAG, "Ошибка при получении данных для
построения маршрута", ioe);
        }
        return null;
    }
}

```

Рис.П2.26. Текст модуля GoogleAPIManager.java

## Модуль OfferManager.java

```

package com.example helponroad.ManageObjectsOfModel;
import android.content.ContentValues;
import android.content.Context;
import android.util.Log;
import com.example helponroad.NetworkOperations.HttpsQuery;
import com.example helponroad.NetworkOperations.MySQLDataBaseSchema;
import com.example helponroad.NetworkOperations.QueryStrings;
import com.example helponroad.ObjectsOfModel.Offer;
import com.example helponroad.ObjectsOfModel.Request;
import com.example helponroad.JSONParsers.OfferParser;
import com.example helponroad.JSONParsers.ReplyParser;
import com.example helponroad.SQLiteDataBaseInterface.OfferQuery;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.IOException;
import java.util.List;

public class OfferManager {

    private static final String TAG="OfferManager";
    /**
     * Отправляет предложение помощи на сервер
     */
    public void addOffer(Offer offer) throws RequestException {
        try {
            String url=
QueryStrings.API_URL+QueryStrings.ADD_NEW_OFFER_SCRIPT;
            byte[] replay =
HttpsQuery.postQuery(url,offer.getOfferData(false,true,true,true));
            ReplyParser replyParser=new ReplyParser(replay);
            Log.i(TAG,"Предложение помощи отправлено. Ответ
сервера:"+replyParser.stringReplay());
            if (!replyParser.getReplaySuccessField()){
                Log.e(TAG,"Код ошибки:"+
replyParser.getReplayErrorCodeField()+
"Сообщение:"+ replyParser.getReplayMessageField());
                throw new RequestException("Ошибка при добавлении
предложения помощи:"+replyParser.getReplayMessageField());
            }
        }
    }
}

```

Рис.П2.27. Текст модуля OfferManager.java

## П2.27. Продолжение

```

catch (IOException |JSONException ex){
    String exceptionMessage="Произошла внутренняя
ошибка при добавлении сообщения.";
    Log.e(TAG,exceptionMessage+"Текст
ошибки:"+ex.getMessage());
    throw new RequestException(exceptionMessage);
}

/**
 * Получает список предложений помощи к заявке
 */
public void getOffers(Context context,Offer offer) throws
RequestException{
    try {
        String url = QueryStrings.API_URL +
QueryStrings.GET_OFFERS_SCRIPT;
        byte[] replay=HttpsQuery.postQuery(url,offer.getOfferData(false,false,fal
se,true));
        //Разбираем полученные данные о заявках и
записываем их в базу
    }
}

ReplyParser replyParser=new ReplyParser(replay);
Log.i(TAG,"Запрос создан. Ответ
сервера:"+replyParser.stringReplay());
if(!replyParser.getReplaySuccessField()){
    throw new
RequestException("Ошибка:"+replyParser.getReplayMessageFiel
d());
}
//Парсим и сохраняем данные о пользователях
предложений к заявке в локальную базу данных
RequestManager requestManager=new RequestManager();
requestManager.parsedAndSaveUserData(replay,context);
//Парсим и сохраняем данные о новых предложениях
помощи в локальную базу данных
this.parsedAndSaveOffersData(replay,context);
} catch (IOException |JSONException ex){
    String exceptionMessage="Произошла внутренняя
ошибка при загрузке сообщений.";
    Log.e(TAG,exceptionMessage+"Текст
ошибки:"+ex.getMessage());
    throw new RequestException(exceptionMessage);
}
}

```

## Модуль PhotoManager.java

```

package com.example.helponroad.ManageObjectsOfModel;
import android.app.Activity;
import android.content.ActivityNotFoundException;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Point;
import android.net.Uri;
import android.os.Environment;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.widget.ImageView;
import android.widget.Toast;
import com.example.helponroad.NetworkOperations.HttpsQuery;
import com.example.helponroad.R;
import com.squareup.picasso.Callback;
import com.squareup.picasso.NetworkPolicy;
import com.squareup.picasso.OkHttpDownloader;
import com.squareup.picasso.Picasso;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;

/**
 * Класс для обработки изображений
 * Created by Oler on 21.04.2017.
 */
public class PhotoManager {

    private File mFile;//файл
    private static final String TAG="PhotoManager";

    /**
     * Создает новый файл в системе
     */
    public PhotoManager() throws IOException{
        try {
            this.createImageFile();
        }catch (IOException e){
            mFile=null;
        }
    }

    /**
     * Подготавливает место для нового файла в файловой
системе устройства
     */
    private void createImageFile() throws IOException {
        String timeStamp=new
SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
        String imageName="JPEG_"+timeStamp+".jpg";
        File
storageDir=Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);
        File image= new File(storageDir,imageName);
        mFile=image;
    }

    /**
     * Возвращаем созданный файл
     */
    public File getFile() {
        return mFile;
    }

    /**
     * Добавление файла в галерию
     */
    public static void addPhotoIntoGallery(String path, Activity
activity) {
        Intent mediaScanIntent = new
Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
        File f = new File(path);
        Uri contentUri = Uri.fromFile(f);
        mediaScanIntent.setData(contentUri);
        activity.sendBroadcast(mediaScanIntent);
    }
}

```

Рис.П2.28. Текст модуля PhotoManager.java

## П2.28. Продолжение

```

    /**
     * Масштабирование фотографии
     * @param path-путь к изображению в файловой системе
     * устройства
     * @param activity- окно приложения, в котором находится
     * компонент для отображения bitmap
     */
    public static Bitmap getScaledBitmap(String path , Activity
    activity) {
        Point size = new Point();
        activity.getWindowManager().getDefaultDisplay()
            .getSize(size);
        return getScaledBitmap(path, size.x, size.y);
    }

    /**
     * Масштабирование фотографии
     * @param path-путь к изображению в файловой системе
     * устройства
     * @param targetWidth-ширина, на которую выполняется
     * масштабирование
     * @param targetHeight-высота, на которую выполняется
     * масштабирование
     */
    private static Bitmap getScaledBitmap(String path,int
    targetWidth,int targetHeight){
        // получаем размеры изображения
        BitmapFactory.Options bmOptions = new
        BitmapFactory.Options();
        bmOptions.inJustDecodeBounds = true;
        BitmapFactory.decodeFile(path, bmOptions);
        int photoW = bmOptions.outWidth;
        int photoH = bmOptions.outHeight;
        // определяем коэффициент масштабирования
        int scaleFactor = Math.min(photoW/targetWidth,
        photoH/targetHeight);
        // преобразуем фотографию в Bitmap объект
        // соответствующий View
        bmOptions.inJustDecodeBounds = false;
        bmOptions.inSampleSize = scaleFactor;
        bmOptions.inPurgeable = true;
        return BitmapFactory.decodeFile(path, bmOptions);
    }

    /**
     * Кадрирование изображения
     *
     * @param requestCode-идентификационный код запроса
     * @param fragment-фрагмент, вызвавший данный метод
     * @param uri-путь к файлу
     * @param width-ширина файла
     * @param height-высота файла
     */
    public static void performCrop(Fragment fragment,Context
    context, int requestCode, Uri uri, int width, int height)
        throws ActivityNotFoundException {
        try {
            // Намерение для кадрирования. Не все устройства
            // поддерживают его
            Intent cropIntent = new
            Intent("com.android.camera.action.CROP");
            cropIntent.setDataAndType(uri, "image/*");
            cropIntent.putExtra("crop", "true");
            cropIntent.putExtra("aspectX", 1);
            cropIntent.putExtra("aspectY", 1);
            cropIntent.putExtra("outputX", width);
            cropIntent.putExtra("outputY", height);
            cropIntent.putExtra("return-data", true);
            fragment.startActivityForResult(cropIntent,
            requestCode);
        }catch (ActivityNotFoundException ex){
            String errorMessage = "Извините, но ваше устройство
            не поддерживает кадрирование";
            Toast toast = Toast.makeText(context, errorMessage,
            Toast.LENGTH_SHORT);
            toast.show();
        }
    }
}
}

/**
 * Создание файла из Bitmap
 * @param bitmap-из которого создается файл
 * @param filePath-путь к создаваемому файлу
 */
public static File generateFileFromBitmap(Bitmap bitmap,String
filePath)throws IOException {
    File file = new File(filePath);

    //Convert bitmap to byte array
    ByteArrayOutputStream bos = new
    ByteArrayOutputStream();
    bitmap.compress(Bitmap.CompressFormat.JPEG, 100, bos);
    byte[] bitmapdata = bos.toByteArray();
    //write the bytes in file
    FileOutputStream fos = new FileOutputStream(file);
    fos.write(bitmapdata);
    fos.flush();
    fos.close();
    return file;
}

/**
 * Загружает изображение в графический компонент
 * @param target - графический компонент
 * @param URL url-изображения
 */
public static void loadImageIntoTarget(Context activity,
Imageview target, String URL){
    Picasso picasso=new
    Picasso.Builder(activity.getApplicationContext())
        .downloader(new
    OkHttpDownloader(activity.getApplicationContext(),250000000))
        .build();
    //Если сеть доступна, то загружаем из сети
    if(HttpsQuery.isNetworkOnline(activity)) {
        picasso.load(Uri.parse(URL))
            .error(R.drawable.ic_action_save_image)
            .placeholder(R.drawable.ic_action_save_image)
            .into(target);
    }else{ //Если сеть недоступна, то загружаем из кеша
        //приложения
        picasso.load(Uri.parse(URL))
            .networkPolicy(NetworkPolicy.OFFLINE)
            .error(R.drawable.ic_action_save_image)
            .placeholder(R.drawable.ic_action_save_image)
            .into(target);
    }
}

/**
 * Загружает изображение в графический компонент
 * @param target - графический компонент
 * @param URL url-изображения
 */
public static void loadImageIntoTarget(Activity activity,
Imageview target, String URL,int width,int height){
    Picasso picasso=new
    Picasso.Builder(activity.getApplicationContext())
        .downloader(new
    OkHttpDownloader(activity.getApplicationContext(),250000000))
        .build();
    //Если сеть доступна, то загружаем из сети
    if(HttpsQuery.isNetworkOnline(activity)) {
        picasso.load(Uri.parse(URL))
            .resize(width, height)
            .centerCrop()
            .error(R.drawable.ic_action_save_image)
            .placeholder(R.drawable.ic_action_save_image)
            .into(target);
    }else{ //Если сеть недоступна, то загружаем из кеша

```

## П2.28. Продолжение

```

приложения
    Picasso.load(Uri.parse(URL))
        .networkPolicy(NetworkPolicy.OFFLINE)
        .resize(width, height)
        .error(R.drawable.ic_action_save_image)
        .placeholder(R.drawable.ic_action_save_image)
        .centerCrop()
        .into(target);
    }
}
/***
 * Загружает изображение в графический компонент
 * (берет изображение из кеша, если его там нет и сеть
доступна,то загружает из сети)
 * @param target - графический компонент
 * @param URL url-изображения
 * @param isLoadFromNetwork-установка флага=true
предусматривает обязательную загрузку изображения из сети
 */
public static void loadImageIntoTarget(final Context
activity,final ImageView target,final String URL,final int
width,final int height,final boolean isLoadFromNetwork){
    Picasso picasso=new
Picasso.Builder(activity.getApplicationContext())
    .downloader(new
OkHttpDownloader(activity.getApplicationContext(),250000000))
    .build();
    if(isLoadFromNetwork){
        //Если сеть доступна, то загружаем из сети
        if (HttpsQuery.isNetworkOnline(activity)) {
            picasso.load(Uri.parse(URL))
                .resize(width, height)
                .centerCrop()
                .into(target);
            Log.i(TAG, "Load from network");
        }
    }else {
        picasso.load(Uri.parse(URL))
            .networkPolicy(NetworkPolicy.OFFLINE)
            .resize(width, height)
            .centerCrop()
            .into(target, new Callback() {
                @Override
                public void onSuccess() {//Удалось загрузить из
кеша
                }
                @Override
                public void onError() {//не удалось загрузить из
кеша
                    Picasso picasso=new
Picasso.Builder(activity.getApplicationContext())
                    .downloader(new
OkHttpDownloader(activity.getApplicationContext(),250000000))
                    .build();
                    //Если сеть доступна, то загружаем из сети
                    if (HttpsQuery.isNetworkOnline(activity)) {
                        picasso.load(Uri.parse(URL))
                            .resize(width, height)
                            .centerCrop()
                    }
                }
            });
    }
}
*/
 * Загружает изображение в графический компонент
 * (берет изображение из кеша, если его там нет и сеть
доступна,то загружает из сети)
 * @param target - графический компонент
 * @param URL url-изображения
 * @param isLoadFromNetwork-установка флага=true
предусматривает обязательную загрузку изображения из сети
 */
public static void loadImageIntoTarget(final Context
activity,final ImageView target,final String URL, final boolean
isLoadFromNetwork){
    Picasso picasso=new
Picasso.Builder(activity.getApplicationContext())
    .downloader(new
OkHttpDownloader(activity.getApplicationContext(),250000000))
    .build();
    if(isLoadFromNetwork){
        //Если сеть доступна, то загружаем из сети
        if (HttpsQuery.isNetworkOnline(activity)) {
            picasso.load(Uri.parse(URL))
                .into(target);
        }
    }else {
        picasso.load(Uri.parse(URL))
            .networkPolicy(NetworkPolicy.OFFLINE)
            .into(target, new Callback() {
                @Override
                public void onSuccess() {//Удалось загрузить из
кеша
                }
                @Override
                public void onError() {//не удалось загрузить из
кеша
                    Picasso picasso=new
Picasso.Builder(activity.getApplicationContext())
                    .downloader(new
OkHttpDownloader(activity.getApplicationContext(),250000000))
                    .build();
                    //Если сеть доступна, то загружаем из сети
                    if (HttpsQuery.isNetworkOnline(activity)) {
                        picasso.load(Uri.parse(URL))
                            .into(target);
                    }
                }
            });
    }
}

```

## Модуль RequestManager.java

```

package com.example.helponroad.ManageObjectsOfModel;
import android.content.ContentValues;
import android.content.Context;
import android.util.Log;
import
com.example.helponroad.NetworkOperations.MySQLDataBaseSc
hema;

```

```

import com.example.helponroad.ObjectsOfModel.Driver;
import com.example.helponroad.ObjectsOfModel.RequestList;
import com.example.helponroad.JSONParsers.DateParser;
import com.example.helponroad.JSONParsers.RequestParser;
import com.example.helponroad.NetworkOperations.HttpsQuery;

```

Рис.П2.29. Текст модуля RequestManager.java

## П2.29. Продолжение

```

import
com.example.helponroad.NetworkOperations.QueryStrings;
import com.example.helponroad.JSONParsers.UserParser;
import
com.example.helponroad.SQLiteDataBaseInterface.RequestQuery;
import com.example.helponroad.JSONParsers.ReplyParser;
import com.example.helponroad.ObjectsOfModel.Request;
import
com.example.helponroad.SQLiteDataBaseInterface.UserQuery;
import
com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.File;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;

< /**
 * Класс для работы с заявками
 * Created by Oler on 13.03.2017.
 */
public class RequestManager {

    private static final String TAG="RequestManager";
    /**
     * Отправляет заявку на сервер
     * если requestImageFile не существует, то отправляется на
     сервер только текст и местоположение заявки
     * если requestImageFile существует, то отправляется на
     сервер текст, местоположение заявки,
     * графический файл requestImageFile
     * @throws RequestException
     */
    public void addRequest(Request request)throws
RequestException {
        File requestImageFile;
        try {
            String
url=QueryStrings.API_URL+QueryStrings.ADD_NEW_REQUESTS_SCRIPT;
            byte[] replay=null;
            //Если передано изображение для отправки
            if(request.getImage() != null) {
                requestImageFile = new File(request.getImage());
                //Если файл изображения существует
                if (requestImageFile.exists()) {
                    replay = HttpsQuery.postQuery(url,
                        request.getRequestData(false,true, true, true, true,
false,false), requestImageFile);
                    FileManager fileManager=new
FileManager(requestImageFile.getPath());
                    fileManager.removeFile();
                }
            } else {
                replay = HttpsQuery.postQuery(url,
                request.getRequestData(false,true, true, true, true, true,
false,false));
            }
            ReplyParser replyParser=new ReplyParser(replay);
            Log.i(TAG,"Заявка отправлена. Ответ
сервера:"+replyParser.stringReplay());
            if (!replyParser.getReplaySuccessField()){
                Log.e(TAG,"Код ошибки:"+
replyParser.getReplayErrorCodeField()+
                "Сообщение:"+
                replyParser.getReplayMessageField());
                throw new RequestException("Ошибка при добавлении
заявки:"+replyParser.getReplayMessageField());
            }
        }
        catch (IOException |JSONException ex){
            String exceptionMessage="Произошла внутренняя
ошибка при добавлении новой заявки.";
            Log.e(TAG,exceptionMessage+"Текст
ошибки:"+ex.getMessage());
            throw new RequestException(exceptionMessage);
        }
    }

    /**
     * Изменяет статус заявки
     * @param context -контекст приложения
     * @param request заявка
     * @throws RequestException
     */
    public void updateRequestStatus(Context context, Request
request)throws RequestException{
        try {
            String url = QueryStrings.API_URL +
QueryStrings.UPDATE_REQUEST_STATUS_SCRIPT;
            JSONObject
requestParameters=request.getRequestData(true,false,false,false,fal
se,false,true);
            byte[]
replay=HttpsQuery.postQuery(url,requestParameters);
            Log.i(TAG,"Размер ответа:"+replay.length);
            //Разбираем полученные данные о заявках и
записываем их в базу
            ReplyParser replyParser=new ReplyParser(replay);
            Log.i(TAG,"Запрос создан. Ответ
сервера:"+replyParser.stringReplay());
            if(!replyParser.getReplaySuccessField()){
                throw new
RequestException("Ошибка:"+replyParser.getReplayMessageFiel
d());
            }
            //Делаем изменения в локальной базе данных
            RequestList requestList=RequestList.get(context);
            requestList.setStatus(request);
        }
        catch (IOException |JSONException ex){
            String exceptionMessage="Произошла внутренняя
ошибка при изменении статуса заявки.";
            Log.e(TAG,exceptionMessage+"Текст
ошибки:"+ex.getMessage());
            throw new RequestException(exceptionMessage);
        }
    }

    /**
     * Генерирует имя файла заявки в закрытом хранилище
     * файлов приложения
     */
    public String generateNameRequestFile(Context mContext){
        String timeStamp=new
SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
        String fileName="REQUEST_"+timeStamp+".jpg";
        File fileDir=mContext.getFilesDir();
        if (fileDir == null) {
            return null;
        }
        return fileDir+"/"+fileName;
    }

    /**
     * Получает список последних заявок пользователя
     */
    public void getActualRequest(Context context)throws
RequestException{
        try {
            String url = QueryStrings.API_URL +
QueryStrings.GET_ACTUAL_REQUESTS_SCRIPT;
            JSONObject requestParameters=new JSONObject();
            requestParameters.put(QueryStrings.ANDROID_KEY_NAME,Qu
eryStrings.ANDROID_KEY_VALUE);
            byte[]
replay=HttpsQuery.postQuery(url,requestParameters);
            Log.i(TAG,"Размер ответа:"+replay.length);
            //Разбираем полученные данные о заявках и
        }
        catch (IOException |JSONException ex){
            String exceptionMessage="Произошла внутренняя
ошибка при добавлении новой заявки.";
            Log.e(TAG,exceptionMessage+"Текст
ошибки:"+ex.getMessage());
            throw new RequestException(exceptionMessage);
        }
    }
}

```

## П2.29. Продолжение

```

записываем их в базу

ReplyParser replyParser=new ReplyParser(replay);
    Log.i(TAG,"Запрос создан. Ответ
сервера:"+replyParser.stringReplay());
    if(!replyParser.getReplaySuccessField()){
        throw new
RequestException("Ошибка:"+replyParser.getReplayMessageFiel
d());
    }

    //Парсим и сохраняем данные о пользователях новых
заявок в локальную базу данных
    this.parsedAndSaveUserData(replay,context);
    //Парсим и сохраняем данные о новых заявках в
локальную базу данных
    this.parsedAndSaveRequestsData(replay,context);
} catch (IOException |JSONException ex){
    String exceptionMessage="Произошла внутренняя
ошибка при загрузке заявок.";
    Log.e(TAG,exceptionMessage+"Текст
ошибки:"+ex.getMessage());
    throw new RequestException(exceptionMessage);
}

/**
 * Получает список актуальных (новых актуальных,
измененных актуальных и пользователей новых
 * актуальных заявок,измененных пользователей
актуальных заявок)
 */
public void getActualAndChangedRequest(Context
context)throws RequestException{
    try {
        UserPreferences userPreferences=new
UserPreferences(context);
        String url = QueryStrings.API_URL +
QueryStrings.GET_ACTUAL_AND_CHANGED_REQUESTS_S
CRIPT;
        JSONObject requestParameters=new JSONObject();
        requestParameters.put(QueryStrings.ANDROID_KEY_NAME,Qu
eryStrings.ANDROID_KEY_VALUE);

        byte[]
replay=HttpsQuery.postQuery(url,requestParameters);

        //Разбираем полученные данные о заявках и
записываем их в базу
        ReplyParser replyParser=new ReplyParser(replay);

        Log.i(TAG,"Запрос создан. Ответ
сервера:"+replyParser.stringReplay());

        if(!replyParser.getReplaySuccessField()){
            throw new
RequestException("Ошибка:"+replyParser.getReplayMessageFiel
d());
        }

        //Парсим и сохраняем данные о пользователях новых
заявок в локальную базу данных
        this.parsedAndSaveUserData(replay,context);

        //Парсим и сохраняем данные о новых заявках в
локальную базу данных
        this.parsedAndSaveRequestsData(replay,context);
    } catch (IOException |JSONException ex){
        String exceptionMessage="Произошла внутренняя
ошибка при загрузке заявок.";
        Log.e(TAG,exceptionMessage+"Текст
ошибки:"+ex.getMessage());
        throw new RequestException(exceptionMessage);
    }
}

/**
 * Парсит и добавляет данные о заявках в локальную базу
данных
 * @param requestData-распарсиваемый поток байтов
 */
public void parsedAndSaveRequestsData(byte []
requestData,Context context) throws JSONException{
    RequestParser requestParser=new
RequestParser(requestData);
    //Разбираем полученные данные о заявках и записываем
их в базу
    List<Request>
requestList=requestParser.getParsedRequests();
    RequestQuery requestQuery=new RequestQuery(context);
    try {
        requestQuery.openConnection();
        for (int i = 0; i < requestList.size(); i++) {
            //Добавляем данные о заявке в локальную базу
            ContentValues requestContentValue =
RequestQuery.getRequestValues(requestList.get(i),
true, true, true, true, false, true, true, true, true,
false,false);
            requestQuery.addOrUpdateRequest(requestContentValue);
        }
    } finally {
        requestQuery.closeConnection();
    }
}

/**
 * Парсит и добавляет данные о пользователях в локальную
базу данных
 * @param userData-распарсиваемый поток байтов
 * @param context-контекст приложения
 */
public void parsedAndSaveUserData(byte [] userData,Context
context) throws JSONException{
    //Разбираем полученные данные о пользователях и
записываем их в локальную базу
    UserParser userParser = new UserParser(userData);
    List<Driver> driverList = userParser.getParsedDrivers();
    UserQuery userQuery = new UserQuery(context);
    try {
        userQuery.openConnection();

        for (int i = 0; i < driverList.size(); i++) {
            //Добавляем данные о пользователе в локальную базу
            ContentValues driverContentValue =
UserQuery.getUserValues(driverList.get(i),
true, true, true, false, true, true, true);
            userQuery.addOrUpdateUser(driverContentValue);
        }
    } finally {
        userQuery.closeConnection();
    }
}

/**
 * Вычисляет расстояние по формуле гаверсинусов между
местоположением заявки
 * и текущим местоположением пользователя
 *
 * @param mRequest - заявка
 * @param userLocation -текущее местоположение
пользователя
 * @return расстояние (в метрах)
 */
public static double calculateDistance(Request mRequest,

```

## П2.29. Продолжение

## Модуль UserManager.java

```
package com.example.helponroad.ManageObjectsOfModel;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.util.Log;
import com.example.helponroad.NetworkOperations.HttpsQuery;
import com.example.helponroad.NetworkOperations.MySQLDataBaseSc
hema;
import
com.example.helponroad.NetworkOperations.QueryStrings;
import com.example.helponroad.ObjectsOfModel.Driver;
import com.example.helponroad.ObjectsOfModel.User;
```

```
import com.example.helponroad.JSONParsers.UserParser;
import org.json.JSONException;
import java.io.File;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class UserManager{

    private UserPreferences mUserPreferences;//настройки
пользователя
```

Рис.П2.30. Текст модуля RequestManager.java

## П2.30. Продолжение

```

private static final String TAG="UserManager";
private static final String
USER_IMAGE_NAME="IMG_USER_";

public String generateTempUserImageName(){
    String postfix = new
SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
    return USER_IMAGE_NAME+postfix+".jpg";
}

/**
 * Формирует объект пользователя из настроек приложения
 * @return-возвращаемый пользователь
 */
public User getUser() {
    User user=new User();
    user.setId(mUserPreferences.getStoredUserId());
    user.setLogin(mUserPreferences.getStoredUserLogin());

user.setPassword(mUserPreferences.getStoredUserPassword());
    user.setName(mUserPreferences.getStoredUserName());
    user.setIsImage(mUserPreferences.getStoredUserIsImage());
    user.setImage(mUserPreferences.getStoredUserImage());
    return user;
}

/**
 * Конструктор, создание объекта
 * @param userPreferences
 */
public UserManager(UserPreferences userPreferences){
    mUserPreferences=userPreferences;
}

/**
 * Выход пользователя из приложения
 */
public void logOut(){
    mUserPreferences.setStoredUserId(null);
    mUserPreferences.setStoredUserLogin(null);
    mUserPreferences.setStoredUserPassword(null);
    mUserPreferences.setStoredUserName(null);
    //Проверяем был ли у пользователя файл, если да, то
удаляем этот файл
if(mUserPreferences.getStoredUserIsImage()&&mUserPreferences
.getStoredImage()!=null) {
    FileManager fileManager = new
FileManager(mUserPreferences.getStoredUserImage());
try{
    fileManager.removeFile();
}catch (IOException ex){
    Log.e(TAG,ex.getMessage());
}
mUserPreferences.setStoredUserIsImage(false);
mUserPreferences.setStoredUserImage(null);
}
}

/**
 * Проверка, авторизован ли пользователь в приложении
 */
public boolean isLogIn(){
    return mUserPreferences.getStoredUserId()!=null;
}

/**
 * Регистрация нового пользователя в приложении
 * @throws UserException
 */
public void registerNewUser(User user) throws UserException {
    try {
        String url=
QueryStrings.API_URL+QueryStrings.ADD_NEW_USER_SCRIPT;
        byte[] reply=HttpsQuery.postQuery(url,user.getJSONUserData(true,true,t
rue,false,false));
        UserParser userParser=new UserParser(reply);
        Log.i(TAG,"Запрос создан. Ответ
сервера:"+userParser.stringReplay());
        if(!userParser.getReplaySuccessField()){
            throw new
UserException("Ошибка:"+userParser.getReplayMessageField());
        }
        User
replayUser=userParser.getParsedUser(true,true,true,false,false,fals
e,false);
        if(userParser.getReplaySuccessField()) {
            mUserPreferences.setStoredUserId(replayUser.getId());
            mUserPreferences.setStoredUserLogin(replayUser.getLogin());
            UserPreferences.setStoredUserPassword(replayUser.getPassword()
);
            mUserPreferences.setStoredUserName(null);
            mUserPreferences.setStoredUserIsImage(false);
            mUserPreferences.setStoredUserImage(null);
        }
    } catch (IOException |JSONException ex){
        String exceptionMessage="Произошла внутренняя
ошибка при регистрации нового пользователя.";
        Log.e(TAG,exceptionMessage+"Текст
ошибки:"+ex.getMessage());
        throw new UserException(exceptionMessage+" Текст
ошибки:"+ex.getMessage());
    }
}

/**
 * Авторизация пользователя в приложении
 * @param user-данные для авторизации пользователя
(логин, пароль)
 */
public void authorizedUser(User user)throws UserException{
    try {
        String url = QueryStrings.API_URL +
QueryStrings.AUTHORIZE_USER_SCRIPT;
        byte[] reply = HttpsQuery.postQuery(url,
user.getJSONUserData(false,true,true,false,false));
        UserParser userParser=new UserParser(reply);
        Log.i(TAG,"Запрос создан. Ответ
сервера:"+userParser.stringReplay());
        if(!userParser.getReplaySuccessField()){
            throw new
UserException("Ошибка:"+userParser.getReplayMessageField());
        }
        User
replayUser=userParser.getParsedUser(true,true,true,true,true,false,f
alse);
        if(userParser.getReplaySuccessField()) {
            mUserPreferences.setStoredUserId(replayUser.getId());
            mUserPreferences.setStoredUserLogin(replayUser.getLogin());
            UserPreferences.setStoredUserPassword(replayUser.getPassword()
);
            mUserPreferences.setStoredUserName(replayUser.getName());
            UserPreferences.setStoredUserIsImage(replayUser.getIsImage());
            mUserPreferences.setStoredUserImage(replayUser.getImage());
        }
    } catch (IOException |JSONException ex){
        String exceptionMessage="Произошла внутренняя
ошибка при авторизации нового пользователя.";
        Log.e(TAG,exceptionMessage+"Текст
ошибки:"+ex.getMessage());
    }
}

```

## П2.30. Продолжение

```

throw new UserException(exceptionMessage+" Текст
ошибки:"+ex.getMessage());
}

/**
 * Загрузить изображение пользователя
 */
public void loadUserImage(User user, Context mContext) throws
UserException{
try {
    String url = QueryStrings.API_URL +
QueryStrings.LOAD_USER_IMAGE_SCRIPT;
    byte[] reply = HttpsQuery.postQuery(url,
user.getJSONUserData(true,false,false,false));
    final Bitmap bitmapImage = BitmapFactory
        .decodeByteArray(reply, 0, reply.length);
    if(bitmapImage==null){
        throw new IOException("Не удалось загрузить
изображение пользователя");
    }
    String filePath=this.generateNameUserFile(mContext);

    PhotoManager.generateFileFromBitmap(bitmapImage,filePath);
    mUserPreferences.setStoredUserImage(filePath);
} catch (IOException | JSONException ex){
    String exceptionMessage="Произошла внутренняя
ошибка загрузке изображения пользователя.";
    Log.e(TAG,exceptionMessage+" Текст
ошибки:"+ex.getMessage());
    throw new UserException(exceptionMessage+" Текст
ошибки:"+ex.getMessage());
}
}

/**
 * Удалить изображение пользователя
 */
public void deleteUserImage(User user) throws UserException{
try {
    String url = QueryStrings.API_URL +
QueryStrings.DELETE_USER_IMAGE_SCRIPT;
    byte[] reply = HttpsQuery.postQuery(url,
user.getJSONUserData(true,false,false,false));
    UserParser userParser=new UserParser(reply);
    Log.i(TAG,"Запрос создан. Ответ
сервера:"+userParser.stringReplay());

    if(!userParser.getReplaySuccessField()){
        throw new
UserException("Ошибка:"+userParser.getReplayMessageField());
    }
    User
replayUser=userParser.getParsedUser(true,false,true,false,fals
e,false);
    if(userParser.getReplaySuccessField()) {
        mUserPreferences.setStoredUserIsImage(false);
        if(mUserPreferences.getStoredUserImage()!=null) {
            FileManager fileManager = new
FileManager(mUserPreferences.getStoredUserImage());
            fileManager.removeFile();
        }
        mUserPreferences.setStoredUserImage(null);
    }
} catch (IOException | JSONException ex){
    String exceptionMessage="Произошла внутренняя
ошибка при удалении изображения.";
    Log.e(TAG,exceptionMessage+" Текст
ошибки:"+ex.getMessage());
    throw new UserException(exceptionMessage+" Текст
ошибки:"+ex.getMessage());
}
}
}

/**
 * Изменение имени пользователя
 */
public void changeUserName(User user) throws UserException{
try {
    String url = QueryStrings.API_URL +
QueryStrings.CHANGE_USER_NAME_SCRIPT;
    byte[] reply = HttpsQuery.postQuery(url,
user.getJSONUserData(true,false,false,true));
    UserParser userParser=new UserParser(reply);
    Log.i(TAG,"Запрос создан. Ответ
сервера:"+userParser.stringReplay());
    if(!userParser.getReplaySuccessField()){
        throw new
UserException("Ошибка:"+userParser.getReplayMessageField());
    }
    User
replayUser=userParser.getParsedUser(false,false,false,true,fal
se,false);
    if(userParser.getReplaySuccessField()) {
mUserPreferences.setStoredUserName(replayUser.getName());
    }
} catch (IOException | JSONException ex){
    String exceptionMessage="Произошла внутренняя
ошибка при изменении имени пользователя.";
    Log.e(TAG,exceptionMessage+" Текст
ошибки:"+ex.getMessage());
    throw new UserException(exceptionMessage+" Текст
ошибки:"+ex.getMessage());
}
}

/**
 * Изменение пароля пользователя
 */
public void changeUserPassword(User user) throws
UserException{
try {
    String url = QueryStrings.API_URL +
QueryStrings.CHANGE_USER_PASSWORD_SCRIPT;
    byte[] reply = HttpsQuery.postQuery(url,
user.getJSONUserData(true,false,true,false));
    UserParser userParser=new UserParser(reply);
    Log.i(TAG,"Запрос создан. Ответ
сервера:"+userParser.stringReplay());
    if(!userParser.getReplaySuccessField()){
        throw new
UserException("Ошибка:"+userParser.getReplayMessageField());
    }
    User
replayUser=userParser.getParsedUser(true,false,true,false,fals
e,fals
e);
    if(userParser.getReplaySuccessField()) {
mUserPreferences.setStoredUserPassword(replayUser.getPasswor
d());
    }
} catch (IOException | JSONException ex){
    String exceptionMessage="Произошла внутренняя
ошибка при изменении пароля пользователя.";
    Log.e(TAG,exceptionMessage+" Текст
ошибки:"+ex.getMessage());
    throw new UserException(exceptionMessage+" Текст
ошибки:"+ex.getMessage());
}
}

/**
 * Изменение изображения пользователя
 */
public void changeUserImage(User user, String filepath) throws
UserException{
try {
    String url = QueryStrings.API_URL +

```

## П2.30. Продолжение

```

QueryStrings.CHANGE_USER_IMAGE_SCRIPT;
byte[] reply = HttpsQuery.postQuery(url,
user.getJSONUserData(true,false,false,false),new
File(filepath));
    UserParser userParser=new UserParser(reply);
    Log.i(TAG,"Запрос создан. Ответ
сервера:"+userParser.stringReplay());
    if(userParser.getReplySuccessField()){
        throw new
UserException("Ошибка:"+userParser.getReplyMessageField());
    }
    User
replayUser=userParser.getParsedUser(false,false,false,false,true,fal
se,false);
    if(userParser.getReplySuccessField()) {
        //Проверяем, есть ли у пользователя изображение
        boolean getIsImage=replayUser.getIsImage();
        //Если изображение добавлено
        if(getIsImage) {
            //Если у пользователя ранее было изображение,
            //то удаляем файл изображения
            // из памяти устройства и записываем в настройки
            //путь к новому файлу
            if(mUserPreferences.getStoredUserImage()!=null){
                FileManager fileManager=new
FileManager(mUserPreferences.getStoredUserImage());
                fileManager.removeFile();
            }
            mUserPreferences.setStoredUserIsImage(true);
            mUserPreferences.setStoredUserImage(filepath);
        }else {
            throw new UserException("Внутренняя ошибка. Не
 удалось изменить изображение пользователя");
        }
    }
}catch (IOException |JSONException ex){
    String exceptionMessage="Произошла внутренняя
ошибка при изменении изображения пользователя.";
    Log.e(TAG,exceptionMessage+"Текст

```

```

ошибки:"+ex.getMessage());
        throw new UserException(exceptionMessage+" Текст
ошибки:"+ex.getMessage());
    }
}

/**
 * Генерирует имя пользовательского файла в закрытом
хранилище файлов приложения
 */
public String generateNameUserFile(Context mContext){
    String timeStamp=new
SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
    String fileName="USER_"+timeStamp+".jpg";
    File fileDir=mContext.getFilesDir();
    if (fileDir == null) {
        return null;
    }
    return fileDir+"/"+fileName;
}

/**
 * Страна HTTP-запроса на получение изображения
пользователя
 * @param driver
 * @return
 */
public String getUserImageURL(Driver driver){
    return
QueryStrings.API_URL+QueryStrings.LOAD_USER_IMAGE_S
CRIPT+
"?"+
MySQLDataBaseSchema.UsersTable.Cols.ID+"="+
driver.getId()+
"&"+QueryStrings.ANDROID_KEY_NAME+"="+QueryStrings.
ANDROID_KEY_VALUE;
}
}

```

## Модуль UserPreferences.java

```

package com.example.helponroad.ManageObjectsOfModel;
import android.content.Context;
import android.preference.PreferenceManager;
import
com.example.helponroad.NetworkOperations.MySQLDataBaseSc
hema;
import
com.example.helponroad.NetworkOperations.QueryStrings;

/**
 * Класс для получения и записи данных о пользователе из
файла настроек приложения
 * Created by Oleg on 16.04.2017.
 */

public class UserPreferences {
    private Context(mContext;
    private static final String
LAST_REQUEST_ID="lastRequestId";
    public static final int SECONDS_IN_MINUTE = 60;

    public UserPreferences(Context context){
        mContext=context;
    }

    public String getStoredUserId() {
        return
PreferenceManager.getDefaultSharedPreferences(mContext)
        .getString(MySQLDataBaseSchema.UsersTable.Cols.ID,
null);
    }

    public void setStoredUserId(String userId) {
        PreferenceManager.getDefaultSharedPreferences(mContext)
        .edit()
        .putString(MySQLDataBaseSchema.UsersTable.Cols.ID, userId)
        .apply();
    }

    public String getStoredUserLogin() {
        return
PreferenceManager.getDefaultSharedPreferences(mContext)
        .getString(MySQLDataBaseSchema.UsersTable.Cols.LOGIN,
null);
    }

    public void setStoredUserLogin(String login) {
        PreferenceManager.getDefaultSharedPreferences(mContext)
        .edit()
        .putString(MySQLDataBaseSchema.UsersTable.Cols.LOGIN,
login)
        .apply();
    }
}

```

Рис.П2.31. Текст модуля UserPreferences.java

## П2.31. Продолжение

```

public String getStoredUserPassword() {
    return
PreferenceManager.getDefaultSharedPreferences(mContext)
.getString(MySQLDataBaseSchema.UsersTable.Cols.PASSWOR
D, null);
}

public void setStoredUserPassword(String password) {
    PreferenceManager.getDefaultSharedPreferences(mContext)
    .edit()
    .putString(MySQLDataBaseSchema.UsersTable.Cols.PASSWOR
D, password)
    .apply();
}

public String getStoredUserName() {
    return
PreferenceManager.getDefaultSharedPreferences(mContext)
    .getString(MySQLDataBaseSchema.UsersTable.NAME,
null);
}

public void setStoredUserName(String name) {
    PreferenceManager.getDefaultSharedPreferences(mContext)
    .edit()
    .putString(MySQLDataBaseSchema.UsersTable.NAME,
name)
    .apply();
}

/**
 * Получает наличие изображения у пользователя (true-есть
изображение,false-нет изображения)
 * @return
 */
public boolean getStoredUserIsImage() {
    return
PreferenceManager.getDefaultSharedPreferences(mContext)
    .getBoolean("IS_"+
MySQLDataBaseSchema.UsersTable.Cols.IMAGE,false);
}

/**
 * Устанавливает наличие изображения у пользователя (true-
есть изображение,false-нет изображения)
 * @param isImage-наличие изображения
 */
public void setStoredUserIsImage(boolean isImage) {
    PreferenceManager.getDefaultSharedPreferences(mContext)
    .edit()
    .putBoolean("IS_"+
MySQLDataBaseSchema.UsersTable.Cols.IMAGE, isImage)
    .apply();
}

/**
 * Возвращает имя файла пользователя
 * @return
 */
public String getStoredUserImage() {
    return
PreferenceManager.getDefaultSharedPreferences(mContext)
    .getString(MySQLDataBaseSchema.UsersTable.Cols.IMAGE,null
);
}

/**
 * Устанавливает имя файла пользователя
 * @param image-изображение пользователя
 */
public void setStoredUserImage(String image) {
    PreferenceManager.getDefaultSharedPreferences(mContext)
    .edit()
    .putString(MySQLDataBaseSchema.UsersTable.Cols.IMAGE,image)
    .apply();
}

/**
 * Возвращает дату последней загрузки данных на
устройство
 * @return- дата последней загрузки заявок на устройство
 */
public String getStoredLastUploadDate(){
    return
PreferenceManager.getDefaultSharedPreferences(mContext)
    .getString(QueryStrings.LAST_UPLOAD_DATE,null);
}

/**
 * Устанавливает дату последней загрузки данных на
устройство
 * @return- дата последней загрузки заявок на устройство
 */
public void setStoredLastUploadDate(String date){
    PreferenceManager.getDefaultSharedPreferences(mContext)
    .edit()
    .putString(QueryStrings.LAST_UPLOAD_DATE, date)
    .apply();
}

/**
 * Возвращает дату последней загрузки данных на
устройство
 * @return- дата последней загрузки заявок на устройство
 */
public String getStoredLastPreviousUploadDate(){
    return
PreferenceManager.getDefaultSharedPreferences(mContext)
    .getString(QueryStrings.LAST_PREVIOUS_UPLOAD_DATE,nul
l);
}

/**
 * Устанавливает дату последней загрузки данных на
устройство
 * @return- дата последней загрузки заявок на устройство
 */
public void setStoredLastPreviousUploadDate(String date){
    PreferenceManager.getDefaultSharedPreferences(mContext)
    .edit()
    .putString(QueryStrings.LAST_PREVIOUS_UPLOAD_DATE,
date)
    .apply();
}

/**
 * Возвращает значение того, что должен ли пользователь
получать уведомления
 * @return - получает ли пользователь уведомления (по
умолчанию-нет)
 */
public boolean getStoredIsGetNotificationsParameter(){
    return
PreferenceManager.getDefaultSharedPreferences(mContext)
    .getBoolean(QueryStrings.IS_GET_NOTIFICATION,false);
}

/**
 * Устанавливает значение того, что должен ли
пользователь получать уведомления (да-получает,нет-не
получает)
 */

```

## П2.31. Продолжение

```

    public void setStoredIsGetNotificationsParameter(boolean
isGetNotification){
    PreferenceManager.getDefaultSharedPreferences(mContext)
        .edit()
        .putBoolean(QueryStrings.IS_GET_NOTIFICATION,
isGetNotification)
        .apply();
}
/***
 * Возвращает радиус получения заявок (-1-получать все
заявки или в пределах 1-100 км)
 * @return - радиус получения уведомлений
 */
public int getStoredRadiusNotifications(){
    return
PreferenceManager.getDefaultSharedPreferences(mContext)
    .getInt(QueryStrings.RADIUS_NOTIFICATIONS,-1);
}

/***
 * Устанавливает радиус получения заявок (-1-получать все
заявки или в пределах 1-100 км)
 */
public void setStoredRadiusNotifications(int
radiusNotifications){

    PreferenceManager.getDefaultSharedPreferences(mContext)
        .edit()
        .putInt(QueryStrings.RADIUS_NOTIFICATIONS,
radiusNotifications)
        .apply();
}

/***
 * Возвращает частоту получения уведомлений в
минутах (по умолчанию 5 минут)
 * @return - радиус получения уведомлений
 */
public int getStoredFrequencyNotifications(){
    return
PreferenceManager.getDefaultSharedPreferences(mContext)
    .getInt(QueryStrings.FREQUENCY_NOTIFICATIONS,5);
}

    }

    /**
     * Устанавливает частоту получения заявок в минутах
     */
    public void setStoredFrequencyNotifications(int
frequencyNotifications){

        PreferenceManager.getDefaultSharedPreferences(mContext)
            .edit()
            .putInt(QueryStrings.FREQUENCY_NOTIFICATIONS,
frequencyNotifications)
            .apply();
    }

    /**
     * Устанавливает просмотрел ли пользователь сообщение о
маршруте
     * @return- дата последней загрузки заявок на устройство
     */
    public void setStoredIsShowRouteInformation(boolean
isShowRouteInformation){

        PreferenceManager.getDefaultSharedPreferences(mContext)
            .edit()
            .putBoolean(QueryStrings.IS_SHOW_ROUTE_MESSAGE,
isShowRouteInformation)
            .apply();
    }

    /**
     * Возвращает значение того, что должен ли пользователь
получать уведомления
     * @return - получает ли пользователь уведомления (по
умолчанию-нет)
     */
    public boolean getStoredIsShowRouteInformation(){
        return
PreferenceManager.getDefaultSharedPreferences(mContext)
    .getBoolean(QueryStrings.IS_SHOW_ROUTE_MESSAGE,false);
    }
}

```

## Модуль RequestException

```

package com.example.helponroad.ManageObjectsOfModel;

public class RequestException extends Exception{

public RequestException(){
    super();
}

```

Рис.П2.32. Текст модуля RequestException.java

## Модуль UserException

```

package com.example.helponroad.ManageObjectsOfModel;

public class RequestException extends Exception{

public RequestException(){
    super();
}

```

```

    }

    public RequestException(String message){
        super(message);
    }
}

```

Рис.П2.33. Текст модуля UserException.java

## Пакет ManagementAddNewRequest

### Модуль AddNewRequestActivity.java

```

package
com.example.helponroad.ApplicationController.ManagementAdd
NewRequest;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.graphics.Bitmap;
import android.location.Location;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.util.Log;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;
import com.example.helponroad.NetworkOperations.HttpsQuery;
import
com.example.helponroad.LocationManager.LocationListenerGPSe
rvices;
import
com.example.helponroad.LocationManager.LocatingListener;
import
com.example.helponroad.ManageObjectsOfModel.PhotoManager;
import
com.example.helponroad.ManageObjectsOfModel.RequestExcepti
on;
import
com.example.helponroad.ManageObjectsOfModel.RequestManage
r;
import
com.example.helponroad.ManageObjectsOfModel.UserManager;
import
com.example.helponroad.ManageObjectsOfModel.UserPreference
s;
import
com.example.helponroad.NetworkOperations.QueryStrings;
import com.example.helponroad.ObjectsOfModel.Request;
import com.example.helponroad.R;
import
com.example.helponroad.ApplicationController.ManagementUser.
CheckLogInDialog;
import
com.example.helponroad.CommonUserInterfaceComponentsLibra
ry.TabActivity;
import
com.example.helponroad.CommonUserInterfaceComponentsLibra
ry.ViewPagerAdapter;
import
com.google.android.gms.common.GooglePlayServicesNotAvailabl
eException;
import com.google.android.gms.common.GooglePlayServicesUtil;
import com.google.android.gms.maps.model.LatLng;
import java.io.IOException;

public class AddNewRequestActivity extends TabActivity
    implements
SelectLocationListener,SelectRequestPhotoListener,
EnterMessageListener,
LocatingListener {
    private static final String TAG="AddNewRequestActivity";
    private static final String
DIALOG_CHECK_LOG_IN="DialogCheckLogIn";
    private static final int REQUEST_ERROR=0;
    private Request mRequest;
    private UserManager mUserManager;
    private LocationListenerGPSServices mLocationListener;
    private LatLng mSelectedLocation;//выбранное
местоположение
    private Bitmap mSelectedImage;//выбранное изображение
    private UpdateLocationListener
mUpdateLocationListener;//слушатель события обновления
местоположения
    private ProgressDialog pDialog;
    private CheckLogInDialog mCheckLogInDialog;
    private final static String
IS_SHOWING_PROGRESS_BAR="is_showing_progress_bar";
    private final static String
PROGRESS_BAR="is_showing_progress_bar";

    @Override
    protected void setContentView(){
        setContentView(R.layout.activity_tab);
    }

    @Override
    protected void addFragments(ViewPagerAdapter
viewPagerAdapter) {
        viewPagerAdapter.addFragment(new
AddNewRequestFragment(),"Описание проблемы");
        AddNewRequestMapFragment mapFragment=new
AddNewRequestMapFragment();

        viewPagerAdapter.addFragment(mapFragment,"Местоположение
");
    }

    mUpdateLocationListener=(UpdateLocationListener)mapFragment
    ;
}

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mRequest=new Request();
        mUserManager=new UserManager(new
UserPreferences(AddNewRequestActivity.this));
        mLocationListener=new
LocationListenerGPSServices(AddNewRequestActivity.this,
QueryStrings.UPDATE_LOCATION_INTERVAL,QueryStrings.
UPDATE_LOCATION_INTERVAL);
        mLocationListener.setLocatingListener(this);
        mCheckLogInDialog=new CheckLogInDialog();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if(pDialog!=null) {
            if (pDialog.isShowing()) {
                pDialog.dismiss();
            }
        }
    }

    @Override
    protected void onStart() {
        super.onStart();
        //Если пользователь не авторизован, то показываем
диалог для авторизации
        if(!mUserManager.isLogIn()){

mCheckLogInDialog.show(getSupportFragmentManager(),DIALO
G_CHECK_LOG_IN);
        }
    }

    try {
}

```

Рис.П2.34. Текст модуля AddNewRequestActivity.java

## П2.34. Продолжение

```

mLocationListener.startLocating();
}catch (GooglePlayServicesNotAvailableException ex){
    Dialog errorDialog=GooglePlayServicesUtil
        .getAlertDialog(ex.errorCode,this,REQUEST_ERROR,
            new DialogInterface.OnCancelListener(){
                @Override
                public void onCancel(DialogInterface
dialogInterface) {
                    //finish();
                }
            });
    errorDialog.show();
}

@Override
protected void onStop() {
    super.onStop();
    mLocationListener.stopLocating();
}

@Override
public void onResume(){
    super.onResume();
}

@Override
public boolean onCreateOptionsMenu(Menu menu){
    getMenuInflater().inflate(R.menu.menu_add_new_request,menu);
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_item_post_request: {
            try {
                if(!mUserManager.isLoggedIn()){
                    mCheckLogInDialog.show(getSupportFragmentManager(),DIALOG_CHECK_LOG_IN);
                    return true;
                }
                if(mUserManager.getUser().getName()==null){
                    throw new RequestException("Для отправки заявки
необходимо указать имя пользователя. " +
                        "Перейдите в настройки учетной записи и
укажите имя пользователя.");
                }
                if
                (!HttpsQuery.isNetworkOnline(AddNewRequestActivity.this)) {
                    throw new RequestException("Отсутствует
подключение к сети Интернет. " +
                        "Проверьте Ваше подключение к сети
Интернет.");
                }
                if (mRequest.getMessage() == null) {
                    throw new RequestException("Не задано сообщение
заявки");
                }
                mRequest.setDriver(mUserManager.getUser());
                LatLng userLocation;
                if (mSelectedLocation == null) {
                    userLocation = new
LatLang(mLocationListener.getCurrentLocation().getLatitude(),
mLocationListener.getCurrentLocation().getLongitude());
                } else {
                    userLocation = new
LatLang(mSelectedLocation.latitude,
mSelectedLocation.longitude);
                }
                mRequest.setLocation(userLocation);
                //Если задано изображение,то создаем файл
изображения
                if (mSelectedImage != null) {
                    RequestManager mRequestManager = new
RequestManager();
                    String mSelectedImageFilename =
mRequestManager.generateNameRequestFile(AddNewRequestAct
ivity.this);
                    try {
                        PhotoManager.generateFileFromBitmap(mSelectedImage,
mSelectedImageFilename);
                        mRequest.setImage(mSelectedImageFilename);
                    } catch (IOException ex) {
                        Log.e(TAG, ex.getMessage());
                    }
                }
                //Запускаем поток, который отправляет данные
заявки на сервер
                new AddNewRequestTask().execute(mRequest);
            } catch (RequestException ex) {
                Toast toast =
Toast.makeText(AddNewRequestActivity.this,
ex.getMessage(),
Toast.LENGTH_LONG);
                toast.setGravity(Gravity.CENTER, 0, 0);
                toast.show();
            }
            return true;
        }
        default: {
            return super.onOptionsItemSelected(item);
        }
    }
}
/***
 *
 * Переопределяемый метод интерфейса обратного вызова
LocatingListener
 * Вызывается при изменении местоположения
*/
@Override
public void onUpdateLocation(final Location location) {
    mUpdateLocationListener.onUpdateLocation(location);
    Log.i(TAG,"Текущее
местоположение:"+location.toString());
}

/**
 *
 * Переопределяемый метод интерфейса обратного вызова
SelectLocationListener
 * Вызывается при выборе пользователем местоположения
на карте
*/
@Override
public void onUpdateSelectedLocation(LatLng
selectedLocation) {
    mSelectedLocation=selectedLocation;
}

/**
 *
 * Переопределяемый метод интерфейса обратного вызова
SelectRequestPhotoListener
 * Вызывается при выборе пользователем фотографии
*/
@Override
public void onSelectImage(Bitmap bitmap) {
    mSelectedImage=bitmap;
}
*/

```

## П2.34. Продолжение

```

* Переопределяемый метод интерфейса обратного вызова
EnterLocationListener
    * Вызывается при вводе пользователем текста сообщения
помощи
    */

    @Override
    public void onEnterMessage(String message) {
        mRequest.setMessage(message);
    }

    private class AddNewRequestTask extends
AsyncTask<Request,Void,String> {

        private RequestManager mRequestManager;
        private Boolean isRequestPosted=false;//отправлена ли
заявка
        /**
         * Перед началом фонового потока Show Progress Dialog
         */
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new
ProgressDialog/AddNewRequestActivity.this);
            pDialog.setMessage("Заявка добавляется...");
            pDialog.setIndeterminate(false);
            pDialog.setCancelable(false);
            pDialog.show();
        }

        @Override
        protected String doInBackground(Request... params) {
            String message="Заявка отправлена успешно";
            mRequestManager =new RequestManager();
            try {
                mRequestManager.addRequest(params[0]);
                isRequestPosted=true;
            }
            catch (RequestException ex){
                isRequestPosted=false;
                message=ex.getMessage();
                Log.e(TAG,message);
            }
            Log.i(TAG,message);
            return message;
        }

        @Override
        protected void onPostExecute(String message) {
            if(pDialog!=null) {
                if (pDialog.isShowing()) {
                    pDialog.dismiss();
                }
            }
            Toast
toast=Toast.makeText/AddNewRequestActivity.this,message,Toas
t.LENGTH_LONG;
            toast.setGravity(Gravity.CENTER, 0, 0);
            toast.show();
            //Если заявка отправлена
            if(isRequestPosted) {
                finish();
            }
        }
    }
}

```

## Модуль AddNewRequestFragment.java

```

package
com.example helponroad.ApplicationController.ManagementAdd
NewRequest;
import android.app.Activity;
import android.content.ActivityNotFoundException;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.Fragment;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AlertDialog;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import
com.example helponroad.ManageObjectsOfModel.PhotoManager;
import com.example helponroad.R;
import java.io.File;

```

```

import java.io.IOException;
import static android.app.Activity.RESULT_OK;

/**
 * Created by Олег on 19.03.2017.
 * Фрагмент для добавления новой заявки о помощи от
водителя
 * Предназначен для заполнения пользователем данных заявки
и отправки в базу на сервере
 */

public class AddNewRequestFragment extends Fragment {

    private PhotoManager mPhotoManager;
    private static final String KEY_BITMAP = "bitmap";
    private final int
PERMISSION_READ_AND_WRITE_EXTERNAL_STORAGE
=1;//обрезать фотографию
    private final int TAKE_TO_THE_CAMERA = 0;//снять на
камеру
    private final int SELECT_FROM_ALBUM=1;//выбрать из
альбома
    private final int CROP_PHOTO =2;//обрезать фотографию
    private Button mImageSelectButton;
    private Button mImageDeleteButton;
    private TextView mMessageTextView;
    private String mEnteredMessage;//введенное пользователем
сообщение
    private EnterMessageListener
mEnterMessageListener;//слушатель события ввода сообщения
    private SelectRequestPhotoListener

```

Рис.П2.35. Текст модуля AddNewRequestFragment.java

## П2.35. Продолжение

```

mSelectRequestPhotoListener;//слушатель события выбора
изображения
    private ImageView mRequestPhotoImageView;
    private Bitmap mRequestPhotoBitmap;
    private AlertDialog.Builder mSelectCameraOrAlbumDialog;

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRetainInstance(true);
    }

    @Override
    public void onViewStateRestored(@Nullable Bundle savedInstanceState) {
        super.onViewStateRestored(savedInstanceState);

        if (savedInstanceState != null){

            mRequestPhotoBitmap =
                savedInstanceState.getParcelable(KEY_BITMAP);

            //Если пользователем было выбрано новое изображение
            if(mRequestPhotoBitmap!=null) {

                mRequestPhotoImageView.setImageBitmap(mRequestPhotoBitmap);

                //Передаем выбранное изображение активности

                mSelectRequestPhotoListener.onSelectImage(mRequestPhotoBitmap);
            }
        }
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);

        //Если пользователь не выбрал новое изображение

        outState.putParcelable(KEY_BITMAP,
            mRequestPhotoBitmap);
    }

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        View v=inflater.inflate(R.layout.fragment_add_new_request,container,f
alse);
        mRequestPhotoImageView=(ImageView)v.findViewById(R.id.image_
view_request_photo);
        mImageSelectButton
        =(Button)v.findViewById(R.id.button_select_image);
        mImageSelectButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                mSelectCameraOrAlbumDialog.show();
            }
        });
        mImageDeleteButton
        =(Button)v.findViewById(R.id.button_delete_image);
        mImageDeleteButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                mRequestPhotoImageView.setImageBitmap(null);
                mRequestPhotoBitmap=null;
                mSelectRequestPhotoListener.onSelectImage(null);
            }
        });
    }
}

```

```

    });

    //Диалог для выбора способа добавления фотографии
    mSelectCameraOrAlbumDialog=new
    AlertDialog.Builder(getContext());
    mSelectCameraOrAlbumDialog.getContext().setTheme(R.style.Ap
pTheme_PrimaryAccent);
    mSelectCameraOrAlbumDialog.setTitle(R.string.user_account_sel
ect_camera_or_album);
    mSelectCameraOrAlbumDialog.setCancelable(true);
    //Выбираем фото из альбома
    mSelectCameraOrAlbumDialog.setPositiveButton("Из
альбома", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            try {
                Intent selectFromAlbumIntent = new
                Intent(Intent.ACTION_PICK);
                if(selectFromAlbumIntent.resolveActivity(getActivity().getPackag
eManager())==null){
                    throw new ActivityNotFoundException("Галерея
недоступна.");
                }
                selectFromAlbumIntent.setType("image/*");
                startActivityForResult(selectFromAlbumIntent,SELECT_FROM_
ALBUM);
            } catch (ActivityNotFoundException ex){
                Toast
                toast=Toast.makeText(getContext(),ex.getMessage(),Toast.LENG
TH_LONG);
                toast.setGravity(Gravity.CENTER, 0, 0);
                toast.show();
            }
        }
    });
    //снимаем на камеру
    mSelectCameraOrAlbumDialog.setNegativeButton("Снять
на камеру", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            try {
                Intent takeToCameraIntent = new
                Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                // проверяем, что есть приложение способное
                // обработать интент
                if
                (takeToCameraIntent.resolveActivity(getActivity().getPackageMa
nager()) == null){
                    throw new IOException("Камера недоступна");
                }
                //создаем файл
                mPhotoManager=new PhotoManager();
                File photoFile=mPhotoManager.getFile();
                if(photoFile==null){
                    throw new IOException ("Возникла внутренняя
ошибка при создании фотографии");
                }
                takeToCameraIntent.putExtra(MediaStore.EXTRA_OUTPUT,
                    Uri.fromFile(photoFile));
                startActivityForResult(takeToCameraIntent,
                    TAKE_TO_THE_CAMERA);
            } catch (IOException ex){
                Toast
                toast=Toast.makeText(getContext(),ex.getMessage(),Toast.LENG
TH_LONG);
                toast.setGravity(Gravity.CENTER, 0, 0);
                toast.show();
            }
        }
    });
    mMessageTextView=(TextView)v.findViewById(R.id.text_view_
request_message);
    mMessageTextView.addTextChangedListener(new

```

## П2.35. Продолжение

```

TextWatcher() {

    @Override
        public void beforeTextChanged(CharSequence
charSequence, int i, int i1, int i2) {
    }
    @Override
        public void onTextChanged(CharSequence charSequence,
int i, int i1, int i2) {
            if(charSequence.length()!=0){
                mEnteredMessage=charSequence.toString();
            } else {
                mEnteredMessage=null;
            }
        }

    mEnterMessageListener.onEnterMessage(mEnteredMessage);
    }
    @Override
        public void afterTextChanged(Editable editable) {
    }
};

return v;
}
@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);

    //Задаем слушателя события ввода сообщения помоши
    mEnterMessageListener=(EnterMessageListener)activity;

    //Задаем слушателя события выбора изображения
    пользователя
    mSelectRequestPhotoListener=(SelectRequestPhotoListener)
activity;
}
@Override
public void onActivityResult(int requestCode, int resultCode,
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    switch (requestCode) {
        case TAKE_TO_THE_CAMERA: {//после съемки
изображения
            if(resultCode == RESULT_OK) {
                //Если нет разрешений на чтение из галереи, то
запрашиваем их
                if (ContextCompat.checkSelfPermission(getActivity(),
                    android.Manifest.permission.READ_EXTERNAL_STORAGE) !=

PackageManager.PERMISSION_GRANTED) {
                    ActivityCompat.requestPermissions(getActivity(),
                        new
String[] { android.Manifest.permission.READ_EXTERNAL_STOR
AGE,
                    android.Manifest.permission.WRITE_EXTERNAL_STORAGE},
                    PERMISSION_READ_AND_WRITE_EXTERNAL_STORAGE);
                } else {//если есть разрешения, то выполняем
получение снимка с камеры и запускаем кадрирование
                    this.onGetPictureFromCamera();
                }
            }
        }
        break;
    }
    case SELECT_FROM_ALBUM: {//после получения из
альбома
        if(resultCode == RESULT_OK) {
            if (data != null) {
                try {
                    //устанавливаем полученное изображение
                    mRequestPhotoBitmap =
MediaStore.Images.Media.getBitmap(getActivity().getContentRes
olver(),
                        data.getData());
                }
            }
        }
    }
}

mRequestPhotoImageView.setImageBitmap(mRequestPhotoBitma
p);                                //Передаем выбранное изображение
активности
mSelectRequestPhotoListener.onSelectImage(mRequestPhotoBitm
ap);                                //запускаем кадрирование изображения
PhotoManager.performCrop/AddNewRequestFragment.this,
getActivity(), CROP_PHOTO,
data.getData(), 256, 256);

} catch (IOException e) {
    Toast toast = Toast.makeText(getApplicationContext(),
        "Не удалось установить выбранное
полноразмерное изображение. Попробуйте обрезать
изображение.");
    toast.LENGTH_LONG);
    toast.setGravity(Gravity.CENTER, 0, 0);
    toast.show();
}

}
break;
}
case CROP_PHOTO: {//после кадрирования
изображения
if(resultCode == RESULT_OK) {
    if (data != null) {
        //Получаем и устанавливаем кадрированное
изображение
        mRequestPhotoBitmap =
data.getExtras().getParcelable("data");

mRequestPhotoImageView.setImageBitmap(mRequestPhotoBitma
p);                                //Передаем выбранное изображение активности
mSelectRequestPhotoListener.onSelectImage(mRequestPhotoBitm
ap);                                //Передаем выбранное изображение активности
}
}
break;
}
}
@Override
public void onRequestPermissionsResult(int requestCode,
@NotNull String[] permissions, @NotNull int[] grantResults) {
    if(requestCode==
PERMISSION_READ_AND_WRITE_EXTERNAL_STORAGE
&& grantResults.length==1){

if(grantResults[0]==PackageManager.PERMISSION_GRANTED)
{
    //Если разрешили, то получаем изображение с
камеры
    this.onGetPictureFromCamera();
}
}
super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
}

/**
 * Получение изображения с камеры и его кадрирование
 */
private void onGetPictureFromCamera() {
}
}

```

## П2.35. Продолжение

```

File userPhoto = mPhotoManager.getFile();

if (userPhoto.exists()) {
    PhotoManager.addPhotoIntoGallery(userPhoto.getPath(),
    getActivity());
    mRequestPhotoBitmap =
    PhotoManager.getScaledBitmap(userPhoto.getPath(),
    getActivity());

mRequestPhotoImageView.setImageBitmap(mRequestPhotoBitmap);
    //Передаем выбранное изображение активности

mSelectRequestPhotoListener.onSelectImage(mRequestPhotoBitmap);

PhotoManager.performCrop(AddNewRequestFragment.this,
getActivity(), CROP_PHOTO,
Uri.fromFile(mPhotoManager.getFile()), 256, 256);
}

}

/***
* Интерфейс обратного вызова для передачи введенного
сообщения
*/

```

```

* от фрагмента AddNewRequestFragment активности-хосту
(AddNewRequestActivity)
*/
interface EnterMessageListener{
    /**
     * Событие ввода сообщения помощи
     * @param message-введенное сообщение
     */
    public void onEnterMessage(String message);
}

/**
* Интерфейс обратного вызова для передачи выбранного
изображения
* от фрагмента AddNewRequestFragment активности-хосту
(AddNewRequestActivity)
*/

```

```

interface SelectRequestPhotoListener{

    /**
     * Событие выбора изображения пользователем
     * @param bitmap-выбранное изображение
     */
    public void onSelectImage(Bitmap bitmap);
}

```

### Модуль AddNewRequestMapFragment.java

```

package
com.example.helponroad.ApplicationController.ManagementAdd
NewRequest;
import android.app.Activity;
import android.os.Bundle;
import android.location.Location;
import
com.example.helponroad.LocationManager.LocationPermission;
import com.example.helponroad.NetworkOperations.HttpsQuery;
import com.example.helponroad.R;
import
com.example.helponroad.ApplicationController.ManagementListR
equests.GetAddressRequestTask;
import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.LatLngBounds;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;

/**
 * Класс для отображения и редактирования местоположения
пользователя на карте
 * в процессе добавления заявки о помощи
 */
public class AddNewRequestMapFragment extends
SupportMapFragment
    implements
UpdateLocationListener,GoogleMap.OnMarkerClickListener{

    private Location mCurrentLocation;
    private GoogleMap mMap;
    private Marker mSelectedLocationMarker;
    private boolean isSetCameraPosition=false;
    private boolean isUserSelectedCurrentLocation=false;
    private SelectLocationListener

```

```

mSelectedLocationListener;//слушатель события выбора
местоположения на карте

@Override
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setRetainInstance(true);
    getMapAsync(new OnMapReadyCallback() {
        @Override
        public void onMapReady(GoogleMap googleMap) {
            mMap = googleMap;
            mMap.clear();
            LocationPermission locationPermission=new
LocationPermission(getApplicationContext());
            if(locationPermission.checkSelfPermission()==
            PackageManager.PERMISSION_GRANTED){
                mMap.setMyLocationEnabled(true);
            }
            mMap.getUiSettings().setMyLocationButtonEnabled(true);
            }
            mMap.getUiSettings().setCompassEnabled(true);
            mMap.getUiSettings().setZoomControlsEnabled(true);
            mMap.setOnMapClickListener(new
GoogleMap.OnMapClickListener() {
                @Override
                public void onClick(LatLng latLng) {
                    updatePositionSelectedLocationMarkerAfterClick(latLng);
                }
            });
        }

mMap.setOnMarkerClickListener(AddNewRequestMapFragment.t
his);
    });
}

```

Рис.П2.36. Текст модуля AddNewRequestMapFragment.java

## П2.36. Продолжение

```

@Override
public void onUpdateLocation(Location location) {
    mCurrentLocation=location;
    setCameraPosition();
    setOrUpdateSelectedLocationMarker();
}

public void setOrUpdateSelectedLocationMarker(){
    if ( mMap == null || mCurrentLocation==null || isUserSelectedCurrentLocation) {
        return;
    }
    LatLng currentPosition=new
LatLng(mCurrentLocation.getLatitude(),
    mCurrentLocation.getLongitude());
    if(mSelectedLocationMarker==null) {

        //Устанавливаем маркер на карте
        mSelectedLocationMarker = mMap.addMarker(new
MarkerOptions()
        .position(currentPosition)
        .flat(false)
        .title("Выбранное местоположение"));
        mSelectedLocationMarker.setTag(-1);
        if(HttpsQuery.isNetworkOnline(getActivity())) {
            new
GetAddressRequestTask(currentPosition,mSelectedLocationMarker).execute();
        }
        mSelectedLocationMarker.showInfoWindow();
    }else{
        //Обновляем положение маркера на карте
        mSelectedLocationMarker.setPosition(currentPosition);
    }
}

@Override
public boolean onMarkerClick(Marker marker) {
    Integer clickedRequestId=(Integer) marker.getTag();
    if(clickedRequestId!=null){
        if(clickedRequestId==-1) {
            if(HttpsQuery.isNetworkOnline(getActivity())) {
                new
GetAddressRequestTask(marker.getPosition(),marker).execute();
            }
            marker.setSnippet(null);
            marker.showInfoWindow();
        }
    }
    return false;
}

/**
 * Изменяет положение маркера на карте после клика
 * пользователем карты
 * @param position-местоположение которое выбрано
 * пользователем в результате клика карты
 */
public void
updatePositionSelectedLocationMarkerAfterClick(LatLng
position){
    if(mSelectedLocationMarker==null) {
        mSelectedLocationMarker = mMap.addMarker(new
MarkerOptions()
        .position(position)
        .flat(false)
        .title("Выбранное местоположение"));
        mSelectedLocationMarker.setTag(-1);
        mSelectedLocationMarker.showInfoWindow();
    }else{
        mSelectedLocationMarker.setPosition(position);
    }
}

mSelectedLocationMarker = mMap.addMarker(new
MarkerOptions()
    .position(position)
    .flat(false)
    .title("Выбранное местоположение"));
mSelectedLocationMarker.setTag(-1);
mSelectedLocationMarker.showInfoWindow();
}else{
    mSelectedLocationMarker.setPosition(position);
}

mSelectedLocationListener.onUpdateSelectedLocation(position);//
передача нового местоположения слушателю
isUserSelectedCurrentLocation=true;
}

/**
 * Устанавливает положение камеры на карте, проверяя
 * значение флага isSetCameraPosition и
 * mCurrentLocation на null
 */
public void setCameraPosition() {
    if(mMap==null || mCurrentLocation==null ||
isSetCameraPosition) {
        return;
    }
    LatLngBounds bounds = new LatLngBounds.Builder()
        .include(new
LatLng(mCurrentLocation.getLatitude(),mCurrentLocation.getLon
gitude()))
        .build();
    int margin =
getResources().getDimensionPixelSize(R.dimen.map_inset_margi
n);
    CameraUpdate update =
CameraUpdateFactory.newLatLangBounds(bounds, margin);
    mMap.animateCamera(update);
    isSetCameraPosition=true;
}

@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);
    //Задаем слушателя события выбора местоположения на
карте
    mSelectedLocationListener=(SelectLocationListener)activity;
}
}

/**
 * Интерфейс обратного вызова для передачи выбранного
 * местоположения
 * от фрагмента AddNewRequestMapFragment активности
AddNewRequestActivity
*/
interface SelectLocationListener {
    /**
     * Вызывается при выборе пользователем местоположения
     * на карте
     */
    public void onUpdateSelectedLocation(LatLng
selectedLocation);
}

```

## Модуль UpdateLocationListener.java

```
package
com.example.helponroad.ApplicationController.ManagementAdd
NewRequest;
import android.location.Location;

public interface UpdateLocationListener {
    /**
     * Событие обновления местоположения
     * @param location-местоположения
     */
    public void onUpdateLocation(Location location);
}
```

Рис.П2.37. Текст модуля UpdateLocationListener.java

## Пакет ManagementListRequests

### Модуль GetAddressRequestTask.java

```
package
com.example.helponroad.ApplicationController.ManagementListR
equests;
import android.os.AsyncTask;
import
com.example.helponroad.ManageObjectsOfModel.GoogleAPIMan
ager;
import com.example.helponroad.JSONParsers.RouteParser;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;

/**
 * Класс для получения данных о местоположении
 * пользователя
 * Created by Олег on 05.05.2017.
 */

public class GetAddressRequestTask extends
AsyncTask<Void,Void(Void> {
    private String mFormattedAddress;
    private LatLng mCurrentLocation;
    private Marker mSelectedMarker;

    public GetAddressRequestTask(LatLng mCurrentLocation,Marker
marker){
        mCurrentLocation=currentLocation;
        mSelectedMarker=marker;
    }

    @Override
    protected Void doInBackground(Void... params) {
        GoogleAPIManager googleAPIManager =new
        GoogleAPIManager();
        LatLng currentLocation=new
        LatLng(mCurrentLocation.latitude,mCurrentLocation.longitude);
        String replay=
        googleAPIManager.getAddress(currentLocation);
        if(replay!=null) {
            mFormattedAddress= RouteParser.addressParser(replay);
        }
        return null;
    }

    @Override
    protected void onPostExecute(Void message) {
        if(mFormattedAddress!=null){
            mSelectedMarker.setSnippet(mFormattedAddress);
            mSelectedMarker.showInfoWindow();
        }
    }
}
```

Рис.П2.38. Текст модуля GetAddressRequestTask.java

## Модуль MainActivity.java

```
package
com.example.helponroad.ApplicationController.ManagementListR
equests;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.location.Location;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
```

```
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;
import
com.example.helponroad.LocationManager.LocatingListener;
import
com.example.helponroad.LocationManager.LocationListenerGPSe
rvices;
import
com.example.helponroad.ManageObjectsOfModel.UserManager;
import
com.example.helponroad.ManageObjectsOfModel.UserPreference
s;
```

Рис.П2.39. Текст модуля MainActivity.java

## П2.39. Продолжение

```

import com.example.helponroad.JSONParsers.DateParser;
import com.example.helponroad.R;
import com.example.helponroad.SQLiteDataBaseInterface.RequestQuery;
import com.example.helponroad.SQLiteDataBaseInterface.UserQuery;
import com.example.helponroad.ApplicationController.ManagementAdd
NewRequest.UpdateLocationListener;
import com.example.helponroad.CommonUserInterfaceComponentsLibra
ry.TabActivity;
import com.example.helponroad.ApplicationController.ManagementUser.
EnterActivity;
import com.example.helponroad.ApplicationController.ManagementUser.
UserAccountActivity;
import com.example.helponroad.ApplicationController.ManagementUser.
UserSettingsActivity;
import com.example.helponroad.CommonUserInterfaceComponentsLibra
ry.ViewPagerAdapter;
import com.google.android.gms.common.GooglePlayServicesNotAvailabl
eException;
import com.google.android.gms.common.GooglePlayServicesUtil;
public class MainActivity extends TabActivity
    implements
NavigationView.OnNavigationItemSelectedListener,LocatingListe
ner {
    public static Intent newIntent(Context context) {
        return new Intent(context, MainActivity.class);
    }

    private static final String TAG="MainActivity";
    private static final int LOCATION_REQUEST_ERROR =0;
    private NavigationView mNavigationView;
    private UserManager mUserManager;
    private LocationListenerGPSServices mLocationListener;
    private SelectNavigationViewItemListener
mSelectNavigationViewItemRequestListListener;
    private SelectNavigationViewItemListener
mSelectNavigationViewItemRequestListMapListener;
    private UpdateLocationListener
mUpdateLocationRequestListFragmentListener;
    private UpdateLocationListener
mUpdateLocationRequestListMapFragmentListener;
    private boolean isSelectUserRequestItem=false;

    @Override
    protected void setContentView(){
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void addFragments(ViewPagerAdapter
viewPagerAdapter) {

        RequestListFragment requestListFragment=new
RequestListFragment();
        RequestListMapFragment requestListMapFragment=new
RequestListMapFragment();

        viewPagerAdapter.addFragment(requestListFragment,"Список");
        viewPagerAdapter.addFragment(requestListMapFragment,"Карта");
        mUpdateLocationRequestListFragmentListener
=(UpdateLocationListener)requestListFragment;
        mUpdateLocationRequestListMapFragmentListener=(UpdateLocat
ionListener)requestListMapFragment;
        mSelectNavigationViewItemRequestListListener=(SelectNavigatio
nViewItemListener)requestListFragment;
    }
}

```

```

nViewItemListener)requestListFragment;

mSelectNavigationViewItemRequestListMapListener=(SelectNavi
gationViewItemListener)requestListMapFragment;
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //!!!!Устанавливаем временную зону для приложения!!!!!
    DateParser.setTimeZone();
    //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    //Удаляем неактуальные заявки из базы
    //Удаляем пользователей неактуальных заявок и не
    добавивших предложений к актуальным заявкам
    this.deleteNotActualRequestsAndUsers();
    mUserManager=new UserManager(new
UserPreferences(this));
    //Указываем слушателя изменения местоположения
    mLocationListener=new
LocationListenerGPSServices(MainActivity.this,
QueryStrings.UPDATE_LOCATION_INTERVAL,QueryStrings.
UPDATE_LOCATION_INTERVAL);
    mLocationListener.setLocatingListener(this);
    DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
    ActionBarDrawerToggle toggle = new
ActionBarDrawerToggle(
        this, drawer, getToolbar(),
        R.string.navigation_drawer_open,
        R.string.navigation_drawer_close);
    drawer.setDrawerListener(toggle);
    toggle.syncState();
    mNavigationView = (NavigationView)
findViewById(R.id.nav_view);
    mNavigationView.setNavigationItemSelectedListener(this);
    this.setVisibilityNavigationDrawerItems();
    if(!HttpsQuery.isNetworkOnline(MainActivity.this)) {
        Toast
toast=Toast.makeText(MainActivity.this,"Отсутствует
подключение к сети интернет. Проверьте Ваше
подключение",Toast.LENGTH_LONG);
        toast.setGravity(Gravity.CENTER, 0, 0);
        toast.show();
    }
}

@Override
protected void onStart() {
    super.onStart();
    try {
        mLocationListener.startLocating();
    }catch (GooglePlayServicesNotAvailableException ex){
        Dialog errorDialog= GooglePlayServicesUtil
            .getErrorDialog(ex.errorCode,this,
LOCATION_REQUEST_ERROR,
        new DialogInterface.OnCancelListener(){
            @Override
            public void onCancel(DialogInterface
dialogInterface) {
            }
        });
        errorDialog.show();
    }
}

@Override
protected void onStop() {
    super.onStop();
    mLocationListener.stopLocating();
}

```

## П2.39. Продолжение

```

public void onResume() {
super.onResume();
    setVisibilityNavigationDrawerItems();
    mNavigationView.invalidate();
    if(isSelectUserRequestItem){
this.getToolbar().setTitle(R.string.main_activity_user_requests);
    } else {
        this.getToolbar().setTitle(R.string.main_activity_requests);
    }
}

@Override
public void onBackPressed() {
    DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
if(drawer.isDrawerOpen(GravityCompat.START)) {
    drawer.closeDrawer(GravityCompat.START);
} else {
    super.onBackPressed();
}
}

@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    switch (item.getItemId()){
        case R.id.nav_requests://Заявки
mSelectNavigationViewItemRequestListListener.onShowRequest
OnlyAppUser(false);
mSelectNavigationViewItemRequestListMapListener.onShowReq
uestOnlyAppUser(false);

this.getToolbar().setTitle(R.string.main_activity_requests);
        isSelectUserRequestItem=false;
        break;
    }
        case R.id.nav_user_requests://Мои заявки
mSelectNavigationViewItemRequestListListener.onShowRequest
OnlyAppUser(true);
mSelectNavigationViewItemRequestListMapListener.onShowReq
uestOnlyAppUser(true);

this.getToolbar().setTitle(R.string.main_activity_user_requests);
        isSelectUserRequestItem=true;
        break;
    }
        case R.id.nav_account://Учетная запись
        Intent intent = new Intent(MainActivity.this,
UserAccountActivity.class);
        startActivity(intent);
        break;
    }
        case R.id.nav_enter://Вход
        Intent intent = new Intent(MainActivity.this,
EnterActivity.class);
        startActivity(intent);
        break;
    }
        case R.id.nav_settings://Настройки
        Intent intent = new
Intent(MainActivity.this,UserSettingsActivity.class);
        startActivity(intent);
        break;
    }
    default:{
        break;
    }
}
DrawerLayout drawer = (DrawerLayout)

```

findViewById(R.id.drawer\_layout);  
 drawer.closeDrawer(GravityCompat.START);  
 return true;  
 }  
 }  
 //Проверяем, авторизован ли пользователь в приложении,  
 // если не авторизован, то показываем пункт Вход и не  
 показываем пункт Учетная запись  
 // если авторизован, то показываем пункт Учетная запись и  
 не показываем пункт вход  
 private void setVisibilityNavigationDrawerItems(){  
 if(mUserManager.isLoggedIn()){  
mNavigationView.getMenu().findItem(R.id.nav\_account).setVisib
le(true);  
mNavigationView.getMenu().findItem(R.id.nav\_enter).setVisible(
false);  
mNavigationView.getMenu().findItem(R.id.nav\_user\_requests).set
Visible(true);  
 }else{  
mNavigationView.getMenu().findItem(R.id.nav\_account).setVisib
le(false);  
mNavigationView.getMenu().findItem(R.id.nav\_user\_requests).set
Visible(false);  
  
mNavigationView.getMenu().findItem(R.id.nav\_enter).setVisible(t
rue);  
 }  
 }  
 @Override  
public void onUpdateLocation(Location location) {  
  
mUpdateLocationRequestListFragmentListener.onUpdateLocation(
location);  
  
mUpdateLocationRequestListMapFragmentListener.onUpdateLoc
ation(location);  
  
Log.i(TAG,"Текущее  
местоположение:"+location.toString());  
 }  
  
private void deleteNotActualRequestsAndUsers(){  
 try {  
 //Удаляем неактуальные заявки
 RequestQuery mRequestQuery = new
RequestQuery(MainActivity.this);
 mRequestQuery.openConnection();
 mRequestQuery.deleteNotActualRequest();
 mRequestQuery.closeConnection();
 //Удаляем пользователей неактуальных заявок
 UserQuery mUserQuery = new
UserQuery(MainActivity.this);
 mUserQuery.openConnection();
 mUserQuery.deleteUserNotActualRequestAndOffers();
 mUserQuery.closeConnection();
 } catch (Exception ex){
 Log.e(TAG,ex.getMessage());
 Toast
toast=Toast.makeText(MainActivity.this,"Произошла внутренняя
ошибка при обновлении заявок",Toast.LENGTH\_LONG);
 toast.setGravity(Gravity.CENTER, 0, 0);
 toast.show();
 }
}
}

## Модуль NewRequestFetcherService.java

```

package
com.example.helponroad.ApplicationController.ManagementListR
equests;
import android.app.Notification;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.res.Resources;
import android.location.Location;
import android.net.ConnectivityManager;
import android.support.v4.app.NotificationCompat;
import android.support.v4.app.NotificationManagerCompat;
import android.util.Log;
import
com.example.helponroad.LocationManager.LocatingListener;
import
com.example.helponroad.LocationManager.LocationListenerGPSe
rvices;
import
com.example.helponroad.ManageObjectsOfModel.RequestExcepti
on;
import
com.example.helponroad.ManageObjectsOfModel.RequestManage
r;
import
com.example.helponroad.ManageObjectsOfModel.UserPreference
s;
import com.example.helponroad.ObjectsOfModel.Request;
import com.example.helponroad.ObjectsOfModel.RequestList;
import com.example.helponroad.R;
import
com.google.android.gms.common.GooglePlayServicesNotAvailabl
eException;
import com.google.android.gms.gcm.GcmNetworkManager;
import com.google.android.gms.gcm.GcmTaskService;
import com.google.android.gms.gcm.PeriodicTask;
import com.google.android.gms.gcm.Task;
import com.google.android.gms.gcm.TaskParams;
import com.google.android.gms.maps.model.LatLng;
import java.util.List;

public class NewRequestFetcherService extends GcmTaskService
    implements LocatingListener {

    public static final String TAG="RequestFetcherService";
    private LocationListenerGPSServices mLocationListener;
    private Location mCurrentLocation;
    public static void startNewRequestFetcherService(Context
context){
        UserPreferences userPreferences=new
UserPreferences(context);
        Task task=new PeriodicTask.Builder()
            .setService(NewRequestFetcherService.class)
.setPeriod(userPreferences.getStoredFrequencyNotifications()*
UserPreferences.SECONDS_IN_MINUTE)

.setFlex((userPreferences.getStoredFrequencyNotifications()*User
Preferences.SECONDS_IN_MINUTE)/4)

.setTag(NewRequestFetcherService.ALARM_SERVICE)
        .setPersisted(true)
        .build();
        userPreferences.setStoredIsGetNotificationsParameter(true);
        GcmNetworkManager.getInstance(context).schedule(task);
    }
    public static void stopNewRequestFetcherService(Context
context){
        GcmNetworkManager.getInstance(context).cancelAllTasks(NewR
equestFetcherService.class);
    }
}

UserPreferences userPreferences=new
UserPreferences(context);
userPreferences.setStoredIsGetNotificationsParameter(false);
}

@Override
public void onCreate() {
    super.onCreate();
    Log.i(TAG, "onCreateTask");
    mLocationListener=new
LocationListenerGPSServices(getApplicationContext(),0,0,1);
    mLocationListener.setLocatingListener(this);
}

@Override
public int onStartCommand(Intent intent, int i, int i1) {
    Log.i(TAG, "onStartCommand");
    try {
        mLocationListener.startLocating();
    }catch (GooglePlayServicesNotAvailableException ex){
        Log.e(TAG, ex.getMessage());
    }
    return super.onStartCommand(intent, i, i1);
}

@Override
public int onRunTask(TaskParams taskParams) {
    Log.i(TAG, "onRunTask");
    RequestList
requestList=RequestList.get(getApplicationContext());
    LatLng currentLocation=null;
    if(isNetworkAvailableAndConnected()) {
        try {
            RequestManager manager = new RequestManager();
            manager.getActualRequest(getApplicationContext());
        }catch (RequestException ex){
            Log.e(TAG, ex.getMessage());
        }
    }
    if(mCurrentLocation!=null) {
        currentLocation=new
LatLng(mCurrentLocation.getLatitude(),mCurrentLocation.getLon
gitude());
    }
    Log.i(TAG,"Значение currentLocation типа LatLng:
"+currentLocation);
    List<Request> requests =
requestList.getNewRequest(currentLocation);
    for(int i=0;i<requests.size();i++){
        Request request=requests.get(i);
        this.showNotification(request);
        //Делаем запрос,что уведомление получено
        request.setGetNotify(true);
        requestList.setGetNotify(request);
    }
    Log.i(TAG,"Список заявок"+requests.toString());
    mLocationListener.stopLocating();
    return GcmNetworkManager.RESULT_SUCCESS;
}

@Override
public void onDestroy() {
    super.onDestroy();
    Log.i(TAG, "onDestroyTask");
}

//Проверка доступности и подключения сети

```

Рис.П2.40. Текст модуля NewRequestFetcherService.java

## П2.40. Продолжение

```

private boolean isNetworkAvailableAndConnected() {
    ConnectivityManager cm =
        (ConnectivityManager)
    getSystemService(CONNECTIVITY_SERVICE);
    boolean isNetworkAvailable = cm.getActiveNetworkInfo() != null;
    boolean isNetworkConnected = isNetworkAvailable &&
        cm.getActiveNetworkInfo().isConnected();
    return isNetworkConnected;
}

@Override
public void onUpdateLocation(Location location) {
    mCurrentLocation=location;
    Log.i(TAG,"Текущее местоположение:
"+location.toString());
}

public void showNotification(Request request){
    Resources resources = getResources();
    Intent i=MainActivity.newIntent(getApplicationContext());
    PendingIntent pi=PendingIntent.getActivity(this,0,i,0);
}

```

```

Notification notification=new
NotificationCompat.Builder(this)
.setTicker(resources.getString(R.string.new_request_service_ticker
))

.setContentTitle(resources.getString(R.string.new_request_service
_title,request.getDriver().getName()))
.setContentText(request.getMessage())
.setContentIntent(pi)
.setAutoCancel(true)
.setSmallIcon(R.mipmap.ic_launcher_service_v2)
.setDefaults(Notification.DEFAULT_ALL)
.setPriority(Notification.PRIORITY_MAX)
.build();

notification.contentIntent=pi;

NotificationManagerCompat
notificationManager=NotificationManagerCompat.from(this);
notificationManager.notify(request.getId(),notification);
}
}

```

### Модуль RequestDetailActivity.java

```

package
com.example.helponroad.ApplicationController.ManagementListR
equests;
import android.app.Activity;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.location.Location;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import
com.example.helponroad.CommonUserInterfaceComponentsLibra
ry.TabActivity;
import
com.example.helponroad.CommonUserInterfaceComponentsLibra
ry.ViewPagerAdapter;
import
com.example.helponroad.LocationManager.LocatingListener;
import
com.example.helponroad.LocationManager.LocationListenerGPS
rvices;
import com.example.helponroad.ObjectsOfModel.Request;
import com.example.helponroad.R;
import
com.example.helponroad.ApplicationController.ManagementAdd
NewRequest.UpdateLocationListener;
import
com.google.android.gms.common.GooglePlayServicesNotAvailabl
eException;
import com.google.android.gms.common.GooglePlayServicesUtil;

public class RequestDetailActivity extends TabActivity
    implements LocatingListener,
RequestStatusListener {
    public static final String REQUEST_ID
    ="com.example.helponroad.request_id";
    public static final String IS_SHOW_USER_REQUESTS
    ="com.example.helponroad.is_show_user_requests";

    private static final int LOCATION_REQUEST_ERROR =0;
    private int mRequestId;
}

```

```

private boolean isShowUserRequests=false;//Показывать
только заявки пользователя приложения
private LocationListenerGPSServices mLocationListener;
//слушатель события обновления местоположения
private UpdateLocationListener
mRequestDetailFragmentLocationListener;
//слушатель события обновления местоположения
private UpdateLocationListener
mRequestDetailMapFragmentLocationListener;
private RequestStatusListener
mRequestMapFragmentStatusListener;

/**
 * Метод для получения интента для запуска активности
RequestDetailActivity
 * @param packageContext
 * @param requestId - идентификатор заявки
 * @return
 */
public static Intent newIntent(Context packageContext,int
requestId,boolean isShowUserRequests){
    Intent i=new
Intent(packageContext,RequestDetailActivity.class);
    i.putExtra(REQUEST_ID,requestId);

    i.putExtra(IS_SHOW_USER_REQUESTS,isShowUserRequests);
    return i;
}

@Override
protected void addFragments(ViewPagerAdapter
viewPagerAdapter) {
    //Получаем идентификатор заявки переданный при
запуске активности
    mRequestId=getIntent().getIntExtra(REQUEST_ID,-1);
    //Получаем флаг-показывать только заявки пользователя
приложения
    isShowUserRequests=getIntent().getBooleanExtra(IS_SHOW_US
ER_REQUESTS,false);
}

```

Рис.П2.41. Текст модуля RequestDetailActivity.java

## П2.41. Продолжение

```

//Если передано неверное значение идентификатора
заявки, то закрываем активность
if(mRequestId== -1){
    finish();
}
//Создаем фрагмент RequestDetailFragment, передаем ему
идентификатор заявки и устанавливаем
//его слушателем обновления местоположения
RequestDetailFragment
requestDetailFragment=RequestDetailFragment.newInstance(mRe
questId,isShowUserRequests);

viewPagerAdapter.addFragment(requestDetailFragment,"Описани
е");
//Создаем фрагмент RequestDetailMapFragment, передаем
ему идентификатор заявки и устанавливаем
//его слушателем обновления местоположения
RequestDetailMapFragment
requestDetailMapFragment=RequestDetailMapFragment.newInstance(mRequestId);

viewPagerAdapter.addFragment(requestDetailMapFragment,"Мап
шрут");

mRequestDetailFragmentLocationListener=(UpdateLocationList
ener) requestDetailFragment;

mRequestDetailMapFragmentLocationListener=(UpdateLocationL
istener)requestDetailMapFragment;
}

@Override
protected void setContentView() {
    setContentView(R.layout.activity_tab);
}
@Override
public void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //Указываем слушателя изменения местоположения
    mLocationListener=new
    LocationListenerGPSServices(RequestDetailActivity.this,
        10*1000,10*1000);
    mLocationListener.setLocatingListener(this);
}

@Override
protected void onStart() {
    super.onStart();
    try {
        mLocationListener.startLocating();
    }catch (GooglePlayServicesNotAvailableException ex){
        Dialog errorDialog= GooglePlayServicesUtil
            .getErrorDialog(ex.errorCode,this,
LOCATION_REQUEST_ERROR,
        new DialogInterface.OnCancelListener(){
            @Override
            public void onCancel(DialogInterface
dialogInterface) {
            }
        });
        errorDialog.show();
    }
}

@Override
protected void onStop() {
    super.onStop();
    mLocationListener.stopLocating();
}
@Override
public void onUpdateLocation(Location location) {
mRequestDetailFragmentLocationListener.onUpdateLocation(loc
ation);

mRequestDetailMapFragmentLocationListener.onUpdateLocation(
location);
}

//Передаем новое значение статуса map-фрагменту для
изменения цвета маркера и его заголовка

@Override
public void onRequestStatusChange(Request request) {
mRequestMapFragmentStatusListener.onRequestStatusChange(req
uest);
}
}

```

## Модуль RequestDetailMapFragment.java

```

package
com.example.helponroad.ApplicationController.ManagementListR
equests;
import android.content.DialogInterface;
import android.location.Location;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.util.Log;
import android.view.Gravity;
import android.widget.Toast;
import com.example.helponroad.LocationManager.LocationPermission;
import com.example.helponroad.ManageObjectsOfModel.GoogleAPIMan
ager;
import
com.example.helponroad.ManageObjectsOfModel.RequestManage
r;
import

```

```

com.example.helponroad.ManageObjectsOfModel.UserPreference
s;
import com.example.helponroad.NetworkOperations.HttpsQuery;
import com.example.helponroad.ObjectsOfModel.Request;
import com.example.helponroad.ObjectsOfModel.RequestList;
import com.example.helponroad.JSONParsers.RouteParser;
import com.example.helponroad.R;
import
com.example.helponroad.ApplicationController.ManagementAdd
NewRequest.UpdateLocationListener;
import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;

```

Рис.П2.42. Текст модуля RequestDetailActivity.java

## П2.42. Продолжение

```

    implements UpdateLocationListener,
    GoogleMap.OnMarkerClickListener,RequestStatusListener{
        private GoogleMap mMap;
        private static final String ARG_REQUEST_ID="request_id";
        private static final String TAG="RequestMapFragment";
        private Location mCurrentLocation;//Текущее
        место положение
        private Location mPreviousLocation;//Предыдущее
        место положение
        private Marker mCurrentLocationMarker;
        private boolean isSetCameraPosition=false;
        private Polyline mRouteLine;//линия маршрута
        private Request mRequest;//выбранная заявка
        private Marker mRequestMarker;//маркер выбранной заявки
        private AlertDialog.Builder mShowNotificationAboutRoute;

        /**
         * Создает экземпляр фрагмента с переданным
         * идентификатором заявки
         * @param requestId - идентификатор заявки
         * @return фрагмент RequestDetailFragment
         */
        public static RequestDetailMapFragment newInstance(int
requestId){
            Bundle args=new Bundle();
            args.putSerializable(ARG_REQUEST_ID,requestId);
            RequestDetailMapFragment fragment=new
RequestDetailMapFragment();
            fragment.setArguments(args);
            return fragment;
        }

        @Override
        public void onCreate(Bundle bundle) {
            super.onCreate(bundle);
            setRetainInstance (true);
            if (!HttpsQuery.isNetworkOnline(getActivity())) {
                Toast toast=Toast.makeText(getActivity(),
                    "Отсутствует подключение к сети интернет.
Проверьте Ваше подключение.");
                toast.LENGTH_LONG);
                toast.setGravity(Gravity.CENTER, 0, 0);
                toast.show();
            }
            this.setRequest();
            getMapAsync(new OnMapReadyCallback() {
                @Override
                public void onMapReady(GoogleMap googleMap) {
                    mMap = googleMap;
                    // Set a listener for marker click.
                    mMap.clear();
                    LocationPermission locationPermission=new
LocationPermission(getApplicationContext());
                    if(locationPermission.checkSelfPermission(FINE_LOCATION_PERMISSION)==
                        PackageManager.PERMISSION_GRANTED){
                        mMap.setMyLocationEnabled(true);
                    }
                    mMap.getUiSettings().setMyLocationButtonEnabled(true);
                    }
                    mMap.getUiSettings().setCompassEnabled(true);
                    mMap.getUiSettings().setZoomControlsEnabled(true);
                    mMap.setOnMapClickListener(new
GoogleMap.OnMapClickListener() {
                        @Override
                        public void onMapClick(LatLng latLng) {
                        });
                    });

                    mMap.setOnMarkerClickListener(RequestDetailMapFragment.this
                    );
                    }
                });

                //Диалог для выбора способа добавления фотографии
                mShowNotificationAboutRoute=new

```

```

        AlertDialog.Builder(getContext());
        ShowNotificationAboutRoute.getContext().setTheme(R.style.App
        Theme_PrimaryAccent);

        mShowNotificationAboutRoute.setTitle(R.string.detail_request_in
formation_about_route);
        mShowNotificationAboutRoute.setCancelable(true);
        //Выбираем фото из альбома

        mShowNotificationAboutRoute.setPositiveButton("Понятно",
new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                UserPreferences userPreferences=new
UserPreferences(getApplicationContext());

                userPreferences.setStoredIsShowRouteInformation(true);
            }
        });
        @Override
        public void onStart() {
            super.onStart();
            UserPreferences userPreferences=new
UserPreferences(getApplicationContext());
            if(!userPreferences.getStoredIsShowRouteInformation()){
                mShowNotificationAboutRoute.show();
            }
        }

        /**
         * Заполняет переданный объект заявка данными из
         * локальной базы
         */
        private void setRequest(){

            //получаем переданный фрагменту идентификатор заявки
            int
requestId=(int)getArguments().getSerializable(ARG_REQUEST_I
D);
            try {
                RequestList requestList = RequestList.get(getActivity());
                //Заполняем объект заявка данными из базы
                mRequest = requestList.getRequest(requestId);
            }catch (Exception ex){
                Log.e(TAG,ex.getMessage());
                Toast toast=Toast.makeText(getActivity(),"Ошибка при
получении данных заявки.");
                toast.LENGTH_LONG);
                toast.setGravity(Gravity.CENTER, 0, 0);
                toast.show();
            }
        }

        @Override
        public void onUpdateLocation(Location location) {
            mCurrentLocation=location;
            setCameraPosition();
            setSelectedLocationMarker();
            //Если есть подключение к сети и карта доступна, то
            добавляем маршрут
            if(mMap!=null &&
HttpsQuery.isNetworkOnline(getActivity())){
                float[] distance = new float[1];
                //Если местоположение пользователя получено впервые
                if(mPreviousLocation!=null) {
                    //Расстояние между местоположениями
                    Location.distanceBetween(mCurrentLocation.getLatitude(),
mCurrentLocation.getLongitude(),
mPreviousLocation.getLatitude(),
mPreviousLocation.getLongitude(),
distance);
                }
                // расстояние между прошлым местоположением и

```

## П2.42. Продолжение

```

текущим не превышает 1 метра, то перестраиваем маршрут
    if(mPreviousLocation==null || distance[0]>2.0){
        mCurrentLocationMarker.setPosition(
            new
            LatLng(mCurrentLocation.getLatitude(),mCurrentLocation.getLongitude()));
            new GetRouteTask(mMap).execute();
        }
    }
    mPreviousLocation=mCurrentLocation;
}

public void setCameraPosition() {
    if (mMap == null || mCurrentLocation == null || isSetCameraPosition) {
        return;
    }
    //Текущее местоположение пользователя
    LatLng currentLocation = new
    LatLng(mCurrentLocation.getLatitude(),
    mCurrentLocation.getLongitude());
    //Пакет, который включает текущее местоположение и
    маркеры заявок
    LatLngBounds.Builder bounds = new
    LatLngBounds.Builder();
    bounds.include(currentLocation);
    bounds.include(mRequest.getLocation());
    int margin =
    getResources().getDimensionPixelSize(R.dimen.map_inset_margi
    n);
    CameraUpdate update =
    CameraUpdateFactory.newLatLngBounds(bounds.build(),
    margin);

    mMap.animateCamera(update);
    isSetCameraPosition = true;
    //Устанавливаем маркер выбранной заявки на карте
    this.setSelectedRequestLocationMarkers();
}

/**
 * Добавление маркеров на карту, которые показывают
местоположения заявок
*/
public void setSelectedRequestLocationMarkers(){
    mRequestMarker=mMap.addMarker(new MarkerOptions()
        .position(mRequest.getLocation())
        .title(RequestManager.getRequestMarkerTitle(mRequest))
        .flat(false)
        .icon(BitmapDescriptorFactory.defaultMarker(RequestManager.get
        tRequestMarkerColor(mRequest))));
    mRequestMarker.showInfoWindow();
}

/**
 * Устанавливает маркер, который отображает
местоположение пользователя
 * маркер перерисовывается в соответствии с автоматически
определенным местоположением
*/
public void setSelectedLocationMarker(){
    if (mMap == null||mCurrentLocation==null) {
        return;
    }
    LatLng currentPosition=new
    LatLng(mCurrentLocation.getLatitude(),
    mCurrentLocation.getLongitude());
    if(mCurrentLocationMarker ==null) {
        //Устанавливаем маркер на карте
        mCurrentLocationMarker = mMap.addMarker(new
        MarkerOptions()
            .position(currentPosition)
            .flat(false)
            .title("Ваше местоположение"));
        mCurrentLocationMarker.showInfoWindow();
    }
}

@Override
public boolean onMarkerClick(Marker marker) {
    if(HttpsQuery.isNetworkOnline(getActivity())) {
        new
        GetAddressRequestTask(marker.getPosition(),marker).execute();
    }
    marker.setSnippet(null);
    marker.showInfoWindow();
    return false;
}

private class GetRouteTask extends
AsyncTask<Void(Void,Void> {
    private GoogleAPIManager mGoogleAPIManager;
    private String mPointsString =null;
    private GoogleMap mMap;
    public GetRouteTask(GoogleMap map) {
        mMap = map;
    }

    @Override
    protected Void doInBackground(Void... params) {
        mGoogleAPIManager =new GoogleAPIManager();

        LatLng userLocation=new
        LatLng(mCurrentLocation.getLatitude(),mCurrentLocation.getLongitude());
        //Делаем https-запрос для получения данных маршрута
        String routeReplay=
        mGoogleAPIManager.getRoute(userLocation,mRequest.getLocati
        on());
        if(routeReplay!=null) {
            //Парсим полученные данные о маршруте
            RouteParser parser = new RouteParser();
            parser.parseReplayAboutRoute(routeReplay);
            //Закодированная строка-массив точек маршрута
            mPointsString = parser.getPoints();
        }
        return null;
    }

    @Override
    protected void onPostExecute(Void message) {
        if(mPointsString ==null){
            Toast toast=Toast.makeText(getApplicationContext(),"Не удалось
построить маршрут",Toast.LENGTH_LONG);
            toast.setGravity(Gravity.CENTER, 0, 0);
            toast.show();
        }else {//Строим маршрут
            this.decodeAndPaintRoute();
        }
    }

    /**
     * Декодируем маршрут в массив точек и рисуем на
карте
     */
    public void decodeAndPaintRoute(){
        List<LatLng> mPoints=PolyUtil.decode(mPointsString);
        PolylineOptions line = new PolylineOptions();
        line.width(8).color(R.color.colorAccent);
        for (int i = 0; i < mPoints.size(); i++) {
            line.add(mPoints.get(i));
        }
        //Меняем маркер местоположения пользователя
        if(i==0){
            mCurrentLocationMarker.setPosition(mPoints.get(i));
        }
    }
}
//Если линия ранее была добавлена,то удаляем ее

```

## П2.42. Продолжение

```
mRouteLine=mMap.addPolyline(line);
```

```
}
```

```
//Изменяем цвет и сообщение маркера
```

```
@Override
public void onRequestStatusChange(Request request) {
    mRequest=request;
    if(mRequestMarker!=null){
```

```
mRequestMarker.setIcon(BitmapDescriptorFactory.defaultMarker(
    RequestManager.getRequestMarkerColor(mRequest)));
mRequestMarker.setTitle(RequestManager.getRequestMarkerTitle(
    mRequest));
    mRequestMarker.showInfoWindow();
}
}
```

### Модуль RequestListFragment.java

```
package
com.example.helponroad.ApplicationController.ManagementListR
equests;
import android.content.Intent;
import android.location.Location;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.support.annotation.Nullable;
import android.support.design.widget.FloatingActionButton;
import android.support.v4.app.Fragment;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ProgressBar;
import android.widget.TextView;
import
com.example.helponroad.ManageObjectsOfModel.PhotoManager;
import
com.example.helponroad.ManageObjectsOfModel.RequestExcepti
on;
import
com.example.helponroad.ManageObjectsOfModel.RequestManage
r;
import
com.example.helponroad.ManageObjectsOfModel.UserManager;
import
com.example.helponroad.ManageObjectsOfModel.UserPreference
s;
import com.example.helponroad.NetworkOperations.HttpsQuery;
import com.example.helponroad.ObjectsOfModel.Driver;
import com.example.helponroad.ObjectsOfModel.Request;
import com.example.helponroad.ObjectsOfModel.RequestList;
import com.example.helponroad.JSONParsers.DateParser;
import com.example.helponroad.R;
import
com.example.helponroad.ApplicationController.ManagementAdd
NewRequest.AddNewRequestActivity;
import
com.example.helponroad.ApplicationController.ManagementAdd
NewRequest.UpdateLocationListener;
import com.google.android.gms.maps.model.LatLng;
import java.util.Date;
import java.util.List;

public class RequestListFragment extends Fragment
    implements
```

```
UpdateLocationListener,SelectNavigationViewItemListener {

    private static final String TAG="RequestListFragment";
    private static final int ADD_NEW_REQUEST_ACTIVITY =1;
    private RecyclerView mRequestRecycleView;
    private RequestAdapter mAdapter;
    private boolean isLoadRequestsList=false;//загружены ли
заявки из базы
    private Location mCurrentLocation;
    private ProgressBar mProgressBar;
    private SwipeRefreshLayout mSwipeRefreshLayout;
    private boolean isShowOnlyUserRequest=false;
    private LinearLayout mNotRequestLinealLayout;

    @Override
    public void onShowRequestOnlyAppUser(boolean
isShowOnlyUserRequest) {
        this.isShowOnlyUserRequest=isShowOnlyUserRequest;
        updateUI();
    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRetainInstance(true);
    }

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        View
view=inflater.inflate(R.layout.fragment_request_list,container,fals
e);
        mNotRequestLinealLayout=(LinearLayout)view.findViewById(R.
id.lineal_layout_not_request);
        mSwipeRefreshLayout=(SwipeRefreshLayout)view.findViewById
(R.id.swipe_refresh_layout);
        mSwipeRefreshLayout.setOnRefreshListener(new
SwipeRefreshLayout.OnRefreshListener() {
            @Override
            public void onRefresh() {
                new Handler().postDelayed(new Runnable() {
                    @Override
                    public void run() {
                        new GetActualRequestsTask(false).execute();
                    }
                }, 2500);
            }
        });
    }
}
```

Рис.П2.43. Текст модуля RequestListFragment.java

## П2.43. Продолжение

```

        mRequestRecycleView = (RecyclerView)
view.findViewById(R.id.request_recycler_view);
        mRequestRecycleView.setLayoutManager(new
LinearLayoutManager(getActivity()));
        FloatingActionButton fab = (FloatingActionButton)
view.findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getActivity(),
AddNewRequestActivity.class);

startActivityForResult(intent,ADD_NEW_REQUEST_ACTIVITY
);
    }
});

mProgressBar=(ProgressBar)view.findViewById(R.id.progress_ba
r);

        return view;
    }

    @Override
    public void onStart() {
super.onStart();

    if(!isLoadRequestsList &&
HttpsQuery.isNetworkOnline(getActivity())){
        new GetActualRequestsTask(true).execute();
    }else{
        updateUI();
    }
}

@Override
public void onActivityResult(int requestCode, int resultCode,
Intent data) {
super.onActivityResult(requestCode, resultCode, data);
switch (requestCode){
    case ADD_NEW_REQUEST_ACTIVITY:{

        if(HttpsQuery.isNetworkOnline(getActivity())){
            new GetActualRequestsTask(true).execute();
        }
        break;
    }
}

@Override
public void onUpdateLocation(Location location) {
    mCurrentLocation=location;
}
/***
 * Метод для обновления списка заявок
 */
private void updateUI(){
    RequestList requestList=RequestList.get(getActivity());
    LatLng currentLocation=null;
    if(mCurrentLocation!=null) {
        currentLocation=new
LatLng(mCurrentLocation.getLatitude(),mCurrentLocation.getLon
gitude());
    }
    List<Request> requests;
    if(isShowOnlyUserRequest){
        requests=requestList.getRequestsAppUser();
    }else {
        requests=requestList.getRequests(currentLocation);
    }
    if(requests.size()==0){
        mNotRequestLinealLayout.setVisibility(View.VISIBLE);
        }else {
            mNotRequestLinealLayout.setVisibility(View.INVISIBLE);
        }
    if(mAdapter==null) {
        mAdapter = new RequestAdapter(requests);
        mRequestRecycleView.setAdapter(mAdapter);
    }else {
        mAdapter.setRequests(requests);
        mAdapter.notifyDataSetChanged();
    }
}
private class RequestHolder extends RecyclerView.ViewHolder
    implements View.OnClickListener{
    private ImageView mUserPhotoImageView;
    private TextView mUserNameTextView;
    private TextView mMessageTextView;
    private TextView mStatusTextView;
    private TextView mAddDateTextView;
    private TextView mDistanceTextView;
    private TextView mNewRequestTextView;
    private UserPreferences mUserPreferences;
    private UserManager mUserManager;
    private Request mRequest;
    public RequestHolder(View itemView){
super(itemView);

mUserPhotoImageView=(ImageView)itemView.findViewById(R.
id.image_view_user_photo);
mUserNameTextView=(TextView)itemView.findViewById(R.id.t
ext_view_user_name);
mMessageTextView=(TextView)itemView.findViewById(R.id.tex
t_view_message);
mStatusTextView=(TextView)itemView.findViewById(R.id.text_
view_status);
mAddDateTextView=(TextView)itemView.findViewById(R.id.te
xt_view_add_date);
mDistanceTextView=(TextView)itemView.findViewById(R.id.tex
t_view_distance);
mNewRequestTextView=(TextView)itemView.findViewById(R.i
d.text_view_new_request);
mUserPreferences=new UserPreferences(getApplicationContext());
mUserManager=new UserManager(mUserPreferences);
itemView.setOnClickListener(this);
}
public void bindRequest(Request request){
    mRequest=request;
}

mUserNameTextView.setText(mRequest.getDriver().getName());
mMessageTextView.setText(getString(R.string.list_request_messa
ge,mRequest.getMessage()));
mStatusTextView.setText(getString(R.string.list_request_status,m
Request.getStatus().getStatusMessage()));
mAddDateTextView.setText(getString(R.string.list_request_date_
sub, DateParser.getDateSub(new
Date(),mRequest.getCreateDate())));
if(mCurrentLocation!=null){
    LatLng userLocation=new
LatLng(mCurrentLocation.getLatitude(),mCurrentLocation.getLon
gitude());
    double
distance=RequestManager.calculateDistance(request,userLocation)
;
    //Переводим расстояние в километры
    distance/=1000;
    mDistanceTextView.setText(getString(R.string.list_request_distan
ce,distance));
}
if(!mRequest.isNew()) {

mNewRequestTextView.setVisibility(View.INVISIBLE);
}else{
    mNewRequestTextView.setVisibility(View.VISIBLE);
    //Делаем заявку не нов mRequest.setNew(false);
}
}

```

## П2.43. Продолжение

```

RequestList mRequestList=RequestList.get(getActivity());
mRequestList.setNotNewRequest(mRequest);
}
Driver driver=request.getDriver();
//Загружаем изображение пользователя
if(driver.getIsImage()) {
    int sizeBitmap = getResources().getDimensionPixelSize
        (R.dimen.user_image_view_item);
    boolean isLoadFromNetwork=false;
}

if(mUserPreferences.getStoredLastPreviousUploadDate()!=null
&& driver.getUpdateDate()!=null) {
    //Было ли обновлено изображение водителя после
предпоследней загрузки на устройство
    isLoadFromNetwork = driver.getUpdateDate()

.after(DateParser.parseStringToDate(mUserPreferences.getStoredL
astPreviousUploadDate()));
}
PhotoManager.loadImageIntoTarget(getActivity(),mUserPhotoIma
geView,
mUserManager.getUserImageURL(driver),sizeBitmap,sizeBitmap,
isLoadFromNetwork);
} else {
mUserPhotoImageView.setImageDrawable(getResources().getDra
wable(R.drawable.ic_action_save_image));
}
@Override
public void onClick(View view) {
    //Запускаем активность для просмотра сведений
детальной заявки
    Intent
intent=RequestDetailActivity.newIntent(getActivity(),mRequest.ge
tId(),isShowOnlyUserRequest);
    startActivityForResult(intent);
}
private class RequestAdapter extends
RecyclerView.Adapter<RequestHolder>{
    private List<Request> mRequests;
    public RequestAdapter(List<Request> requests){
        mRequests=requests;
    }
    @Override
    public RequestHolder onCreateViewHolder(ViewGroup
parent, int viewType) {
        LayoutInflater
layoutInflater=LayoutInflater.from(getActivity());
        View
view=layoutInflater.inflate(R.layout.list_item_request,parent,false)
;
        return new RequestHolder(view);
    }
    @Override
    public void onBindViewHolder(RequestHolder holder, int
position) {
        Request request=mRequests.get(position);
        holder.bindRequest(request);
    }
}
@Override
public int getItemCount() {
    return mRequests.size();
}

public void setRequests(List<Request> crimes) {
    mRequests = crimes;
}

private class GetActualRequestsTask extends
AsyncTask<Void,Void,String> {
    private RequestManager mRequestManager;
    private boolean isShowProgressDialog;
    public GetActualRequestsTask(boolean
isShowProgressDialog) {
        this.isShowProgressDialog=isShowProgressDialog;
    }
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        if(isShowProgressDialog) {
            mProgressBar.setVisibility(View.VISIBLE);
        }
    }
    @Override
    protected String doInBackground(Void... params) {
        String message="Заявки получены успешно";
        mRequestManager =new RequestManager();
        try {
            UserPreferences userPreferences=new
UserPreferences(getApplicationContext());
            mRequestManager.getActualRequest(getApplicationContext());
userPreferences.setStoredLastPreviousUploadDate(userPreferences
.getStoredLastUploadDate());
userPreferences.setStoredLastUploadDate(DateParser.parseDateTo
String(new Date()));
        } catch (RequestException ex){
            message=ex.getMessage();
            Log.e(TAG,message);
        }
        Log.i(TAG,message);
        return message;
    }
    @Override
    protected void onPostExecute(String message) {
        if(isShowProgressDialog) {
            mProgressBar.setVisibility(View.INVISIBLE);
        }
        updateUI();
        isLoadRequestsList=true;
        mSwipeRefreshLayout.setRefreshing(false);
    }
}
}

```

## Модуль RequestDetailFragment.java

```

package
com.example.helponroad.ApplicationController.ManagementListR
equests;
import android.app.Activity;
import android.app.AlertDialog;
import android.location.Location;
import android.media.Ringtone;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v4.widget.NestedScrollView;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import
com.example.helponroad.ManageObjectsOfModel.OfferManager;
import
com.example.helponroad.ManageObjectsOfModel.PhotoManager;
import
com.example.helponroad.ManageObjectsOfModel.RequestExcepti
on;
import
com.example.helponroad.ManageObjectsOfModel.RequestManage
r;
import
com.example.helponroad.ManageObjectsOfModel.UserManager;
import
com.example.helponroad.ManageObjectsOfModel.UserPreference
s;
import com.example.helponroad.NetworkOperations.HttpsQuery;
import com.example.helponroad.ObjectsOfModel.Driver;
import com.example.helponroad.ObjectsOfModel.Offer;
import com.example.helponroad.ObjectsOfModel.OfferList;
import com.example.helponroad.ObjectsOfModel.Request;
import com.example.helponroad.ObjectsOfModel.RequestList;
import com.example.helponroad.JSONParsers.DateParser;
import com.example.helponroad.R;
import
com.example.helponroad.ApplicationController.ManagementAdd
NewRequest.UpdateLocationListener;
import
com.example.helponroad.ApplicationController.ManagementUser
CheckLogInDialog;
import com.google.android.gms.maps.model.LatLng;
import java.util.Date;
import java.util.List;
import java.util.Timer;
import java.util.TimerTask;

/**
 * Фрагмент для просмотра сведений заявки и добавления
 сообщений предложения помочи к заявке
 */

```

```

*/
public class RequestDetailFragment extends Fragment
    implements UpdateLocationListener {
    private static final String ARG_REQUEST_ID="request_id";
    private static final String
IS_SHOW_ONLY_USER_REQUEST="is_show_only_user_requ
est";
    private static final String TAG="RequestDetailFragment";
    private static final String
DIALOG_CHECK_LOG_IN="DialogCheckLogIn";
    private Request mRequest;//просматриваемая заявка
    private UserManager mUserManager;
    private RequestManager mRequestManager;
    private boolean isGetOffers=false;//получен ли список
предложений
    private boolean isShowUserRequests=false;//показывать
только заявки пользователя приложения
    private Location mCurrentLocation;
    private Offer mOffer;
    private ImageView mUserPhotoImageView;
    private TextView mUserNameTextView;
    private TextView mStatusTextView;
    private Spinner mStatusSpinner;
    private TextView mMessageTextView;
    private TextView mAddDateTextView;
    private TextView mDistanceTextView;
    private TextView mNotOffersTextView;
    private TextView mRequestPhotoTextView;
    private ImageView mRequestPhotoImage;
    private RecyclerView mOffersRecycleView;
    private EditText mOfferMessageEditText;
    private ImageButton mSendImageButton;
    private ProgressBar mProgressBar;
    private LinearLayoutManager mRecycleViewLLM;
    private NestedScrollView mNestedScrollView;
    private CheckLogInDialog mCheckLogInDialog;
    private OfferAdapter mAdapter;
    private Timer mTaskTimer;
    private RequestStatusListener mActivityRequestStatusListener;

    public static RequestDetailFragment newInstance(int
requestId,boolean isShowUserRequests){
        Bundle args=new Bundle();
        args.putSerializable(ARG_REQUEST_ID,requestId);
        args.putSerializable(IS_SHOW_ONLY_USER_REQUEST,isSho
wUserRequests);
        RequestDetailFragment fragment=new
RequestDetailFragment();
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRetainInstance(true);
        //получаем переданный фрагменту идентификатор заявки
        int
requestId=(int) getArguments().getSerializable(ARG_REQUEST_I
D);
        //получаем флаг-показывать только заявки пользователя
приложения
        isShowUserRequests=(boolean) getArguments().getSerializable(IS
_SHOW_ONLY_USER_REQUEST);
        //Если задано показывать заявки пользователя
приложения, то показываем пункты меню
        if(isShowUserRequests) {
            setHasOptionsMenu(true);
        }
    }
}

```

Рис.П2.44. Текст модуля RequestDetailFragment.java

## П2.44. Продолжение

```

try {
    RequestList requestList = RequestList.get(getActivity());
    //Заполняем объект заявка данными из базы
    mRequest = requestList.getRequest(requestId);
} catch (Exception ex) {
    Log.e(TAG, ex.getMessage());
    Toast toast=Toast.makeText(getActivity(),"Ошибка при
получении данных заявки.",Toast.LENGTH_LONG);
    toast.setGravity(Gravity.CENTER, 0, 0);
    toast.show();
}

@Override
public View onCreateView(LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
    mUserManager = new UserManager(new
UserPreferences(getActivity()));
    mRequestManager = new RequestManager();
    mCheckLogInDialog = new CheckLogInDialog();
    View view =
inflater.inflate(R.layout.fragment_request_detail, container, false);
mNestedScrollView=(NestedScrollView)view.findViewById(R.id.
nested_scroll_view);
    mNotOffersTextView = (TextView)
view.findViewById(R.id.text_view_not_offer);
    mProgressBar = (ProgressBar)
view.findViewById(R.id.progress_bar);
    mUserPhotoImageView = (ImageView)
view.findViewById(R.id.image_view_user_photo);
    Driver driver = mRequest.getDriver();
    //Заполняем объект предложения помощи
    mOffer = new Offer();
    mOffer.setRequestId(mRequest.getId());
    //Загружаем изображение пользователя, если оно у него
есть
    if (driver.getIsImage()) {
        int sizeBitmap = getResources().getDimensionPixelSize
        (R.dimen.user_image_view_item);

        PhotoManager.loadImageIntoTarget(getActivity(),
mUserPhotoImageView,
            mUserManager.getUserImageURL(driver),
sizeBitmap, sizeBitmap, false);
    }
    mUserNameTextView = (TextView)
view.findViewById(R.id.text_view_user_name);

    mUserNameTextView.setText(mRequest.getDriver().getName());
    //В дальнейшем изменим видимость объектов
    mStatusTextView = (TextView)
view.findViewById(R.id.text_view_status);
    //Если задано показывать заявки только пользователя
приложения. то показываем выпадающий
    //список для изменения заявки и показываем в списке
статус заявки
    if (isShowUserRequests) {
        //Настройка выпадающего списка для выбора статуса
        mStatusSpinner = (Spinner)
view.findViewById(R.id.spinner_status);
        mStatusSpinner.setVisibility(View.VISIBLE);
        ArrayAdapter<?> adapter =
ArrayAdapter.createFromResource(getActivity(),
            R.array.status_values,
            android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spin
ner_dropdown_item);
        mStatusSpinner.setAdapter(adapter);
    }

    mStatusSpinner.setSelection(mRequest.getStatus().getStatusValue(
));
    mStatusSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> adapterView,

```

```

View view,
int selectedItemId, long selectedId)
{
    mRequest.getStatus().setStatusValue(selectedItemId);

    @Override
    public void onNothingSelected(AdapterView<?>
adapterView) {
    });
}
else{//Если задано показывать все заявки,то показываем
статус заявки в textView
mStatusTextView.setText(mRequest.getStatus().getStatusMessage
());
}

mMessageTextView=(TextView)view.findViewById(R.id.text_vie
w_message);
    mMessageTextView.setText(mRequest.getMessage());

mAddDateTextView=(TextView)view.findViewById(R.id.text_vie
w_add_date);

mAddDateTextView.setText(getString(R.string.list_request_date_
sub,
DateParser.getDateSub(new
Date(),mRequest.getCreateDate())));
mDistanceTextView=(TextView)view.findViewById(R.id.text_vie
w_distance);
    this.setDistanceDistanceTextView();
    mRequestPhotoTextView = (TextView)
view.findViewById(R.id.text_view_request_photo);
    mRequestPhotoImageView = (ImageView)
view.findViewById(R.id.image_view_request_photo);
    if(!mRequest.getIsImage()) {
        mRequestPhotoTextView.setVisibility(View.GONE);
        mRequestPhotoImageView.setVisibility(View.GONE);
    } else {
        PhotoManager.loadImageIntoTarget(getActivity(),mRequestPhoto
ImageView,
mRequestManager.getRequestImageURL(mRequest),false);
    }
    mOffersRecycleView=(RecyclerView)
view.findViewById(R.id.offers_recycle_view);
    mRecycleViewLLM=new
LinearLayoutManager(getActivity());
    mOffersRecycleView.setLayoutManager(mRecycleViewLLM);
    mOffersRecycleView.setLayoutManager(mRecycleViewLLM);
    mOffersRecycleView.setNestedScrollingEnabled(false);
    mOfferMessageEditText
=(EditText)view.findViewById(R.id.edit_text_offers_message);
    mOfferMessageEditText.addTextChangedListener(new
TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence
charSequence, int i, int i1, int i2) {
        }

        @Override
        public void onTextChanged(CharSequence charSequence,
int i, int i1, int i2) {
            if(charSequence.length()!=0){
                mOffer.setMessage(charSequence.toString());
            } else {
                mOffer.setMessage(null);
            }
        }

        @Override
        public void afterTextChanged(Editable editable) {
        }
});
    mSendImageButton=(ImageButton)view.findViewById(R.id.imag

```

## П2.44. Продолжение

```

e_button_send_offer);
    mSendImageButton.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {
        try {
            if (!mUserManager.isLogIn()) {
                mCheckLogInDialog.show(
getActivity().getSupportFragmentManager(),DIALOG_CHECK_L
OG_IN);

                return;
            }
            if(mUserManager.getUser().getName()==null){
                throw new RequestException("Для отправки
сообщения необходимо указать имя пользователя. " +
                    "Перейдите в настройки учетной записи и
укажите имя пользователя. ");
            }
            if(!HttpsQuery.isNetworkOnline(getActivity())){
                throw new RequestException("Отсутствует
подключение к сети Интернет. " +
                    "Проверьте Ваше подключение к сети
Интернет.");
            }
            if(mOffer.getMessage()==null){
                throw new RequestException("Не задано
сообщение.");
            }
            mOffer.setDriver(mUserManager.getUser());
            new AddNewOfferTask().execute(mOffer);
        }catch (RequestException ex){
            Toast toast = Toast.makeText(getActivity(),
                ex.getMessage(),
                Toast.LENGTH_LONG);
            toast.setGravity(Gravity.CENTER, 0, 0);
            toast.show();
        }
    });
    return view;
}

@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater
inflater) {
    super.onCreateOptionsMenu(menu, inflater);
    inflater.inflate(R.menu.menu_request_detail,menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()){
        case R.id.item_change_request://Сохраняем изменение
заявки
            try {
                if(!HttpsQuery.isNetworkOnline(getActivity())))
                    throw new RequestException("Отсутствует
подключение к сети Интернет. " +
                        "Проверьте Ваше подключение к сети
Интернет.");
            }
            new UpdateRequestStatusTask().execute(mRequest);
        }catch (RequestException ex){

            Toast toast = Toast.makeText(getApplicationContext(),
                ex.getMessage(), Toast.LENGTH_LONG);
            toast.setGravity(Gravity.CENTER, 0, 0);
            toast.show();
        }
    return true;
}
}

default:{  

    return super.onOptionsItemSelected(item);  

}  

}  

}  

@Override  

public void onStart() {  

    super.onStart();  

    if(!isGetOffers &&  

HttpsQuery.isNetworkOnline(getActivity())){  

        mOffer.setDriver(mUserManager.getUser());  

        new UpdateOffersTask().execute(mOffer);  

    }else{  

        updateUI();  

    }  

    this.checkNewOffers();  

}  

private void setDistanceDistanceTextView(){  

    //Вычисляем расстояние до заявки  

    if(mCurrentLocation!=null){  

        LatLng userLocation=new  

LatLang(mCurrentLocation.getLatitude(),mCurrentLocation.getLongitude());  

        double  

distance=RequestManager.calculateDistance(mRequest,userLocati  

on);  

        //Переводим расстояние в километры  

        distance/=1000;  

mDistanceTextView.setText(getString(R.string.list_request_distan  

ce,distance));  

    }  

}  

@Override  

public void onDestroy() {  

    super.onDestroy();  

    mTaskTimer.cancel();  

}  

@Override  

public void onAttach(Activity activity) {  

    super.onAttach(activity);  

mActivityRequestStatusListener=(RequestStatusListener)activity;  

}  

/**  

 * Вызывается при изменении местоположения  

 * @param location-местоположение  

*/
@Override  

public void onUpdateLocation(Location location) {  

    mCurrentLocation=location;  

    if(mDistanceTextView!=null){
        setDistanceDistanceTextView();
    }
}  

private class OfferHolder extends RecyclerView.ViewHolder {  

    private ImageView mUserPhotoImageView;  

    private TextView mUserNameTextView;  

    private TextView mMessageTextView;  

    private TextView mAddDateTextView;  

    private TextView mNewOfferTextView;  

    private Offer mOffer;  

    public OfferHolder(View itemView){  

        super(itemView);  

UserPhotoImageView=(ImageView)itemView.findViewById(R.id
.image_view_user_photo);  

mUserNameTextView=(TextView)itemView.findViewById(R.id.t

```

## П2.44. Продолжение

```

ext_view_user_name);
MessageTextView=(TextView)itemView.findViewById(R.id.text_
view_message);
mAddDateTextView=(TextView)itemView.findViewById(R.id.te
xt_view_add_date);
NewOfferTextView=(TextView)itemView.findViewById(R.id.tex
t_view_new_offer);
    //mUserPreferences=new UserPreferences(getApplicationContext());
    //mUserManager=new UserManager(mUserPreferences);
}

public void bindOffer(Offer offer){
    mOffer=offer;

mUserNameTextView.setText(mOffer.getDriver().getName());
    mMessageTextView.setText(mOffer.getMessage());

AddDateTextView.setText(getString(R.string.list_request_date_su
b, DateParser.getDateSub(new Date(),mOffer.getCreateDate())));
if(!mOffer.isNew()) {
    mNewOfferTextView.setVisibility(View.GONE);
}else{
    mNewOfferTextView.setVisibility(View.VISIBLE);
    //Делаем предложение не новым
    mOffer.setNew(false);
    OfferList mOfferList=OfferList.get(getActivity());
    mOfferList.setNotNewOffer(mOffer);
}
Driver driver=mOffer.getDriver();
//Загружаем изображение пользователя
if(driver.getIsImage()) {
    int sizeBitmap = getResources().getDimensionPixelSize
        (R.dimen.user_image_view_item);
    PhotoManager.loadImageIntoTarget(getActivity(),mUserPhotoIma
geView,
    mUserManager.getUserImageURL(driver),sizeBitmap,sizeBitmap)
;
}
}

private class OfferAdapter extends
RecyclerView.Adapter<RequestDetailFragment.OfferHolder>{
    private List<Offer> mOffers;

    public OfferAdapter(List<Offer> offers){
        mOffers=offers;
    }

    @Override
    public RequestDetailFragment.OfferHolder
onCreateViewHolder(ViewGroup parent, int viewType) {
        LayoutInflator
layoutInflater=LayoutInflater.from(getActivity());
        View
view=layoutInflater.inflate(R.layout.list_item_offer,parent,false);
        return new RequestDetailFragment.OfferHolder(view);
    }

    @Override
    public void
onBindViewHolder(RequestDetailFragment.OfferHolder holder,
int position) {

        Offer offer=mOffers.get(position);
        holder.bindOffer(offer);
    }

    @Override
    public int getItemCount() {
        return mOffers.size();
    }

    public void setOffers(List<Offer> offers) {
        mOffers = offers;
    }
}

}
/***
 * Метод для обновления списка заявок
 */
private void updateUI(){
    OfferList offerList=OfferList.get(getApplicationContext());
    List<Offer> offers=offerList.getOffers(mRequest.getId());
    if(offers.size()==0){
        mNotOffersTextView.setVisibility(View.VISIBLE);
    }else {
        mNotOffersTextView.setVisibility(View.INVISIBLE);
    }
    if(mAdapter==null) {
        mAdapter = new
RequestDetailFragment.OfferAdapter(offers);
        mOffersRecycleView.setAdapter(mAdapter);
    }else {
        mAdapter.setOffers(offers);
        mAdapter.notifyDataSetChanged();
    }
}

mNestedScrollView.postDelayed(new Runnable{
    @Override
    public void run() {

mNestedScrollView.fullScroll(NestedScrollView.FOCUS_DOWN
);
    }
},1000);
}

private void checkNewOffers(){
    mTaskTimer=new Timer();
    mTaskTimer.scheduleAtFixedRate(new TimerTask() {
        @Override
        public void run() {
            if(getApplicationContext()!=null) {
                if (HttpsQuery.isNetworkOnline(getApplicationContext())) {
                    new GetNewOffersTask().execute(mRequest);
                }
            }
        }
    },1000*30,1000*60); //повторяем каждые 60 секунд
}
}

private class AddNewOfferTask extends
AsyncTask<Offer,Void,String> {

    private OfferManager mOfferManager;
    private Boolean isOfferPosted =false;//добавлено ли
сообщение

    @Override
    protected void onPreExecute() {
        mSendImageButton.setEnabled(false);
        mProgressBar.setVisibility(View.VISIBLE);
    }

    @Override
    protected String doInBackground(Offer... params) {
        String message="Сообщение добавлено успешно";
        mOfferManager =new OfferManager();
        try {
            mOfferManager.addOffer(params[0]);
            isOfferPosted =true;
        }
        catch (RequestException ex){
            isOfferPosted =false;
            message=ex.getMessage();
            Log.e(TAG,message);
        }
        Log.i(TAG,message);
    }
}

```

## П2.44. Продолжение

```

return message;
}

@Override
protected void onPostExecute(String message) {
    mProgressBar.setVisibility(View.INVISIBLE);
    if(!isOfferPosted) {
        mSendImageButton.setEnabled(true);
        Toast toast=Toast.makeText(getApplicationContext(),"Произошла
ошибка при добавлении сообщения.",Toast.LENGTH_LONG);
        toast.setGravity(Gravity.CENTER, 0, 0);
        toast.show();
    } else{ //Если сообщение отправлено успешно
        //Обновляем список
        mOfferMessageEditText.setText(null);
        mOffer.setDriver(mUserManager.getUser());
        new UpdateOffersTask().execute(mOffer);
    }
}

private class UpdateOffersTask extends
AsyncTask<Offer(Void,Void> {
    private OfferManager mOfferManager;

    /**
     * Перед началом фонового потока Show Progress Dialog
     */
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        mProgressBar.setVisibility(View.VISIBLE);
    }

    @Override
    protected Void doInBackground(Offer... params) {
        String message="Список получен успешно";
        mOfferManager =new OfferManager();
        try {
            mOfferManager.getOffers(getApplicationContext(),params[0]);
            isGetOffers =true;
        } catch (RequestException ex){
            isGetOffers =false;
            message=ex.getMessage();
            Log.e(TAG,message);
        }
        Log.i(TAG,message);
        return null;
    }

    @Override
    protected void onPostExecute(Void message) {
        mProgressBar.setVisibility(View.INVISIBLE);
        updateUI();
        mSendImageButton.setEnabled(true);
    }
}

/**
 * Фоновое задание для получения новых сообщений
 */
private class GetNewOffersTask extends
AsyncTask<Request(Void,Boolean> {

    private OfferManager mOfferManager;

    @Override
    protected Boolean doInBackground(Request... params) {
        Log.e(TAG,"Start GetNewOffersTask");
        boolean isGetNewOffer=false;
        isGetNewOffer=mOfferManager.getNewOffers(getApplicationContext(),para
ms[0]);
    }
}

@Override
protected void onPostExecute(Boolean isGetNewOffer) {
    //Если получены новые предложения, то обновляем
    //список и воспроизводим звуковой сигнал
    if(isGetNewOffer){
        updateUI();
        Uri
alertRingtone=RingtoneManager.getDefaultUri(RingtoneManager.
TYPE_NOTIFICATION);
        Ringtone
ringtone=RingtoneManager.getRingtone(getApplicationContext(),alertRington
e);
        ringtone.play();
    }
}

private class UpdateRequestStatusTask extends
AsyncTask<Request(Void,String> {
    private RequestManager mRequestManager;
    private ProgressDialog pDialog;

    /**
     * Перед началом фонового потока Show Progress Dialog
     */
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        pDialog = new ProgressDialog(getApplicationContext());
        pDialog.setMessage("Выполняется изменение
заявки...");
        pDialog.setIndeterminate(true);
        pDialog.setCancelable(false);
        pDialog.show();
    }

    @Override
    protected String doInBackground(Request... params) {
        String message="Заявка изменена успешно.";
        mRequestManager=new RequestManager();
        try {
            mRequestManager.updateRequestStatus(getApplicationContext(),params[0]);
        } catch (RequestException ex){
            message=ex.getMessage();
            Log.e(TAG,message);
        }
        Log.i(TAG,message);
        return message;
    }

    @Override
    protected void onPostExecute(String message) {
        if(pDialog!=null&&pDialog.isShowing()) {
            pDialog.dismiss();
        }
    }
}
//Передаем данные активности об измененном статусе

```

## П2.44. Продолжение

заявки

```
\mActivityRequestStatusListener.onRequestStatusChange(mReque
st);
    Toast toast = Toast.makeText(getApplicationContext(), message,
Toast.LENGTH_LONG);
    toast.setGravity(Gravity.CENTER, 0, 0);
```

### Модуль RequestListMapFragment.java

```
package
com.example.helponroad.ApplicationController.ManagementListR
equests;
import android.content.DialogInterface;
import android.content.Intent;
import android.location.Location;
import android.os.Bundle;
import android.os.Handler;
import android.support.v7.app.AlertDialog;
import
com.example.helponroad.LocationManager.LocationPermission;
import
com.example.helponroad.ManageObjectsOfModel.RequestManage
r;
import com.example.helponroad.NetworkOperations.HttpsQuery;
import com.example.helponroad.ObjectsOfModel.Request;
import com.example.helponroad.ObjectsOfModel.RequestList;
import com.example.helponroad.R;
import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import
com.example.helponroad.ApplicationController.ManagementAdd
NewRequest.UpdateLocationListener;
import
com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.LatLngBounds;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import java.util.ArrayList;
import java.util.List;

/**
 * Фрагмент для отображения списка отметок
 * местоположений заявок
 * Created by Олег on 02.05.2017.
 */
public class RequestListMapFragment extends
SupportMapFragment
    implements UpdateLocationListener,
        GoogleMap.OnMarkerClickListener,
        SelectNavigationViewItemListener {

private GoogleMap mMap;
private Marker mSelectedLocationMarker;
private AlertDialog.Builder mSelectRequestDialog;
Integer mClickedRequestId;
private Location mCurrentLocation;
private boolean isSetCameraPosition=false;
private boolean isShowOnlyUserRequest=false;
private List<Marker> mRequestsMarkers;
private RequestList requestList;

@Override
public void onShowRequestOnlyAppUser(boolean
```

```
isShowOnlyUserRequest) {
    this.isShowOnlyUserRequest=isShowOnlyUserRequest;
    //Перерисовываем маркеры заявок и положение камеры
    isSetCameraPosition = false;
    setMapContent();
}

@Override
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setRetainInstance(true);
    mRequestsMarkers=new ArrayList<>();
    requestList = RequestList.get(getActivity());
    getMapAsync(new OnMapReadyCallback() {
        @Override
        public void onMapReady(GoogleMap googleMap) {
            mMap = googleMap;
            // Set a listener for marker click.
            mMap.clear();
            LocationPermission locationPermission=new
LocationPermission(getApplicationContext());
if(locationPermission.checkSelfPermission()==true) {
                mMap.setMyLocationEnabled(true);
            }
            mMap.getUiSettings().setMyLocationButtonEnabled(true);
        }
        mMap.getUiSettings().setCompassEnabled(true);
        mMap.getUiSettings().setZoomControlsEnabled(true);

        mMap.setOnMarkerClickListener(RequestListMapFragment.this);
    });
}

//Диалог для просмотра деталей заявки
mSelectRequestDialog=new
AlertDialog.Builder(getContext());
mSelectRequestDialog.setTitle(R.string.detail_request_show_detai
l_of_request);
    mSelectRequestDialog.setCancelable(true);
    mSelectRequestDialog.getContext().setTheme(R.style.AppTheme_
PrimaryAccent);
    mSelectRequestDialog.setPositiveButton("Ok", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            if(mClickedRequestId!=null) {
                //Запускаем активность для просмотра сведений
                Intent intent =
RequestDetailActivity.newIntent(getActivity(),
mClickedRequestId,
isShowOnlyUserRequest);
                startActivity(intent);
            }
        }
    });
}
```

Рис.П2.45. Текст модуля RequestListMapFragment.java

## П2.45. Продолжение

```

    }
});

mSelectRequestDialog.setNegativeButton("Отмена", new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        dialogInterface.cancel();
    }
});

}

@Override
public boolean onMarkerClick(Marker marker) {
    mClickedRequestId=(Integer) marker.getTag();
    if(httpsQuery.isNetworkOnline(getActivity())) {
        new
GetAddressRequestTask(marker.getPosition(),marker).execute();
    }
    marker.setSnippet(null);
    if(mClickedRequestId!=null){
        mSelectRequestDialog.show();
    }
    return false;
}

@Override
public void onUpdateLocation(Location location) {
    mCurrentLocation=location;
    setMapContent();
}

public void setMapContent(){
    if(mMap != null || mCurrentLocation!=null){
        //Текущее местоположение пользователя
        LatLng currentLocation = new
LatLang(mCurrentLocation.getLatitude(),
        mCurrentLocation.getLongitude());
        //заявки пользователей
        List<Request> requests;
        //Если задано показывать только заявки пользователя,
показываем только их
        if(isShowOnlyUserRequest){
            requests=requestList.getRequestsAppUser();
        }else {
            requests=requestList.getRequests(currentLocation);
        }
        setCameraPosition(requests);
        setOrUpdateSelectedLocationMarker();
        setPositionRequestsMarkers(requests);
    }
}
private void setCameraPosition(List<Request> requests) {
    if (isSetCameraPosition) {
        return;
    }
    //Текущее местоположение пользователя
    LatLang currentLocation = new
}

}

LatLang(mCurrentLocation.getLatitude(),
mCurrentLocation.getLongitude());
//Пакет, который включает текущее местоположение и
маркеры заявок
LatLangBounds.Builder bounds = new
LatLangBounds.Builder();
bounds.include(currentLocation);
for (int i = 0; i < requests.size(); i++) {
    bounds.include(requests.get(i).getLocation());
}
int margin =
getResources().getDimensionPixelSize(R.dimen.map_inset_margi
n);
CameraUpdate update =
CameraUpdateFactory.newLatLangBounds(bounds.build(),
margin);
mMap.animateCamera(update);
isSetCameraPosition = true;
}

private void setPositionRequestsMarkers(List<Request>
requests){
    //Очищаем маркеры, если они есть
    for(int i=0;i<mRequestsMarkers.size();i++){
        mRequestsMarkers.get(i).remove();
    }
    //И очищаем список с маркерами
    mRequestsMarkers.clear();
    for(int i=0;i< requests.size();i++){
        Request requestItem=requests.get(i);
        Marker requestItemMarker=mMap.addMarker(new
MarkerOptions()
    .position(requestItem.getLocation())
    .title(RequestManager.getRequestMarkerTitle(requestItem))
    .flat(true)
    .icon(BitmapDescriptorFactory.defaultMarker(RequestManager.ge
tRequestMarkerColor(requestItem))));
        mRequestsMarkers.add(requestItemMarker);
        requestItemMarker.setTag(requestItem.getId());
    }
}

private void setOrUpdateSelectedLocationMarker(){
    LatLang currentPosition=new
LatLang(mCurrentLocation.getLatitude(),
mCurrentLocation.getLongitude());
    if(mSelectedLocationMarker==null) {
        //Устанавливаем маркер на карте
        mSelectedLocationMarker = mMap.addMarker(new
MarkerOptions()
    .position(currentPosition)
    .flat(true)
    .title("Ваше местоположение"));
        mSelectedLocationMarker.showInfoWindow();
    }else{
        //Обновляем положение маркера на карте
        mSelectedLocationMarker.setPosition(currentPosition);
    }
}

```

### Модуль SelectNavigationViewItemListener.java

```

package
com.example.helponroad.ApplicationController.ManagementListR
equests;

public interface SelectNavigationViewItemListener{

```

```

    public void onShowRequestOnlyAppUser(boolean
isShowOnlyUserRequest);
}

```

Рис.П2.46. Текст модуля SelectNavigationViewItemListener.java

## Пакет ManagementUser

### Модуль CheckLogInDialog.java

```

package
com.example.helponroad.ApplicationController.ManagementUser;
import android.app.Dialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.DialogFragment;
import android.support.v7.app.AlertDialog;
import com.example.helponroad.R;

public class CheckLogInDialog extends DialogFragment {
    @NonNull
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        //Диалог для выхода из учетной записи
        AlertDialog mSingleAlertDialog=new
        AlertDialog.Builder(getContext());
        mSingleAlertDialog.getContext().setTheme(R.style.AppTheme_Pr
        imaryAccent);
        mSingleAlertDialog.setTitle(R.string.add_request_log_in_into_ap
        p);
        mSingleAlertDialog.setCancelable(false);
        mSingleAlertDialog.setPositiveButton("Войти", new
        DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                Intent intent = new Intent(getActivity(),
                EnterActivity.class);
                startActivity(intent);
                CheckLogInDialog.this.dismiss();
            }
        });
        mSingleAlertDialog.setNegativeButton("Отмена", new
        DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                getActivity().finish();
            }
        });
        return mSingleAlertDialog.create();
    }
}

```

Рис.П2.47. Текст модуля CheckLogInDialog.java

### Модуль EnterActivity.java

```

package
com.example.helponroad.ApplicationController.ManagementUser;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import
com.example.helponroad.CommonUserInterfaceComponentsLibra
ry.SingleFragmentActivity;
public class EnterActivity extends SingleFragmentActivity {
    @Override
    public Fragment createFragment() {
        return EnterFragment.newInstance();
    }
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}

```

Рис.П2.48. Текст модуля EnterActivity.java

### Модуль EnterFragment.java

```

package
com.example.helponroad.ApplicationController.ManagementUser;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.text.Editable;
import android.text.InputType;
import android.text.TextWatcher;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.Toast;
import com.example.helponroad.NetworkOperations.HttpsQuery;
import
com.example.helponroad.ManageObjectsOfModel.UserException;
import
com.example.helponroad.ManageObjectsOfModel.UserManager;
import
com.example.helponroad.ManageObjectsOfModel.UserPreference
s;

```

Рис.П2.49. Текст модуля EnterFragment.java

## П2.49. Продолжение

```

import com.example.helponroad.ObjectsOfModel.User;
import com.example.helponroad.R;

/*
 * Фрагмент для входа в учетную запись или для перехода к
 * регистрации нового пользователя
 * Created by Oleg on 08.04.2017.
 */
public class EnterFragment extends Fragment {

    private Button mRegisterButton;
    private Button mEnterButton;
    private EditText mLoginEditText;
    private EditText mPasswordEditText;
    private CheckBox mShowPasswordCheckBox;
    private ProgressDialog pDialog;
    private boolean isShowingProgressDialog=false;
    private User mUser;//объект, описывающий данные
    //пользователя
    private static final String TAG="EnterFragment";

    public static EnterFragment newInstance() {
        return new EnterFragment();
    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRetainInstance(true);
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);

        if(isShowingProgressDialog){
            pDialog.dismiss();
        }
    }

    @Override
    public void onViewStateRestored(@Nullable Bundle savedInstanceState) {
        super.onViewStateRestored(savedInstanceState);

        if(isShowingProgressDialog){
            pDialog.show();
        }
    }

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable
    ViewGroup container, @Nullable Bundle savedInstanceState) {
        mUser=new User();
        View
        view=inflater.inflate(R.layout.fragment_enter,container,false);

        mRegisterButton=(Button)view.findViewById(R.id.button_register);
        mRegisterButton.setOnClickListener(new
        View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //запускаем активность для регистрации нового
                //пользователя
                Intent intent = new Intent(getActivity(),
                RegistrationActivity.class);
                startActivity(intent);
            }
        });
        mEnterButton=(Button)view.findViewById(R.id.button_enter);
        mEnterButton.setOnClickListener(new
        View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //запускаем поток для авторизации пользователя
                new AuthorizeNewUserTask().execute(mUser);
            }
        });
    }

    View.OnClickListener() {
        @Override
        public void onClick(View view) {

            try {
                if (!HttpsQuery.isNetworkOnline(getActivity())) {
                    throw new UserException("Отсутствует
                    подключение к сети Интернет."
                    + "Проверьте Ваше подключение к сети
                    Интернет.");
                }
                if (mUser.getLogin() == null || mUser.getPassword()
                == null) {
                    throw new UserException("Все поля обязательны
                    для заполнения.");
                }
                if (mUser.getLogin().length() < 4) {
                    throw new UserException("Логин должен
                    содержать не менее 4 символов."
                    + "Проверьте введенные данные.");
                }
                if (mUser.getPassword().length() < 4) {
                    throw new UserException("Пароль должен
                    содержать не менее 4 символов."
                    + "Проверьте введенные данные.");
                }
                //Запускаем поток для авторизации пользователя в
                //приложении
                new AuthorizeNewUserTask().execute(mUser);

            }catch (UserException ex){
                Toast
                toast=Toast.makeText(getActivity(),ex.getMessage(),Toast.LENGTH
                TH_LONG);
                toast.setGravity(Gravity.CENTER, 0, 0);
                toast.show();
            }
        });
    });

    mLoginEditText=(EditText)view.findViewById(R.id.edit_text_login);
    mLoginEditText.addTextChangedListener(new
    TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence
        charSequence, int i, int i1, int i2) {
        }
        @Override
        public void onTextChanged(CharSequence charSequence,
        int i, int i1, int i2) {
            mUser.setLogin(charSequence.toString());
        }
        @Override
        public void afterTextChanged(Editable editable) {
        }
    });
    mPasswordEditText=(EditText)view.findViewById(R.id.edit_text_password);
    mPasswordEditText.addTextChangedListener(new
    TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence
        charSequence, int i, int i1, int i2) {
        }
        @Override
        public void onTextChanged(CharSequence charSequence,
        int i, int i1, int i2) {
            mUser.setPassword(charSequence.toString());
        }
        @Override
        public void afterTextChanged(Editable editable) {
        }
    });
}

```

## П2.49. Продолжение

## Модуль RegistrationActivity.java

```
package com.example.helponroad.ApplicationController.ManagementUser;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import com.example.helponroad.CommonUserInterfaceComponentsLibrary.SingleFragmentActivity;
public class RegistrationActivity extends SingleFragmentActivity
{
}
```

```
@Override  
public Fragment createFragment() {  
    return RegistrationFragment.newInstance();  
}  
  
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
}  
}
```

Рис.П2.50. Текст модуля RegistrationActivity.java

## Модуль SetDialogListener.java

```
package com.example.helponroad.ApplicationController.ManagementUser;
interface SetDialogListener {
    public void onCancelDialog();
}
```

Рис.П2.51. Текст модуля SetDialogListener.java

## Модуль SetUserNameDialog.java

```
package com.example.helponroad.ApplicationController.ManagementUser;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.DialogFragment;
import android.support.v4.app.Fragment;
import android.support.v7.app.AlertDialog;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import com.example.helponroad.NetworkOperations.HttpsQuery;
import com.example.helponroad.ManageObjectsOfModel.UserException;
import com.example.helponroad.ManageObjectsOfModel.UserManager;
import com.example.helponroad.ManageObjectsOfModel.UserPreference;
import com.example.helponroad.ObjectsOfModel.User;
import com.example.helponroad.R;
import com.example.helponroad.CommonUserInterfaceComponentsLibrary.SingleFragmentActivity;

/**
 * Диалоговое окно для изменения имени пользователя
 */
public class SetUserNameDialog extends DialogFragment {

    private UserManager mUserManager;
    private EditText mEditText;
    private String mUserName;
    private SetDialogListener mDialogListener;
    private String TAG="SetUserNameDialog";

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        SingleFragmentActivity
activity=(SingleFragmentActivity)context;
        Fragment fragment=
activity.getSupportFragmentManager().findFragmentById(R.id.fragment_container);
        if(fragment!=null){
            mDialogListener=(SetDialogListener)fragment;
        }
    }

    @NonNull
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        mUserManager=new UserManager(new
UserPreferences(getContext()));
        AlertDialog.Builder builder=new
AlertDialog.Builder(getContext());
        LayoutInflater inflater = getActivity().getLayoutInflater();
        final View
view=inflater.inflate(R.layout.fragment_edit_dialog,null);

        mEditText=(EditText)view.findViewById(R.id.edit_text_name);
        mUserName=mUserManager.getUser().getName();
        if(mUserManager.getUser().getName()!=null){
            mEditText.setText(mUserName);
        }
        mEditText.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence
charSequence, int i, int i1, int i2) {
            }
            @Override
            public void onTextChanged(CharSequence charSequence,
int i, int i1, int i2) {
                if(charSequence.length()!=0) {
                    mUserName = charSequence.toString();
                }else{
                    mUserName=null;
                }
            }
            @Override
            public void afterTextChanged(Editable editable) {
            }
        });
        builder.setCancelable(true);
        builder.setView(view);
        builder.setTitle(R.string.user_account_change_name);
        builder.setPositiveButton("OK", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                try {
                    if (!HttpsQuery.isNetworkOnline(getActivity())) {
                        throw new UserException("Отсутствует
подключение к сети Интернет."
+ "Проверьте Ваше подключение к сети
Интернет.");
                    }
                    if(mUserName.equals("null")){
                        throw new UserException("Значение null является
системным значением "
+ "и не может быть установлено в качестве
имени.");
                    }
                }
            }
        });
    }
}
```

Рис.П2.52. Текст модуля SetUserNameDialog.java

## П2.52. Продолжение

```

}
if(mUserName==null){
    throw new UserException("Не задано имя
пользователя");
}
User user = mUserManager.getUser();
user.setName(mUserName);
new
ChangeNameUserTask(SetUserNameDialog.this.getDialog()).exec
ute(user);
}catch (UserException ex){
    Toast
toast=Toast.makeText(getApplicationContext(),ex.getMessage(),Toast.LENGTH_
LONG);
    toast.setGravity(Gravity.CENTER, 0, 0);
    toast.show();
}
}
builder.setNegativeButton("Отмена", new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        SetUserNameDialog.this.getDialog().cancel();
    }
});
return builder.create();
}

@Override
public void onCancel(DialogInterface dialog) {
    super.onCancel(dialog);
    mDialogListener.onCancelDialog();
}

/***
 * Фоновый поток для отправки запроса на сервер на
изменение имени пользователя
*/
private class ChangeNameUserTask extends
AsyncTask<User,Void,String> {

    private UserManager mUserManager;
    private Dialog mDialog;
    private ProgressDialog pDialog;
    public ChangeNameUserTask(Dialog dialog) {
        mDialog = dialog;
    }
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        pDialog = new ProgressDialog(getApplicationContext());
        pDialog.setMessage("Выполняется изменение имени...");
        pDialog.setIndeterminate(true);
        pDialog.setCancelable(false);
        pDialog.show();
    }
    @Override
    protected String doInBackground(User... user) {
        String message="Имя пользователя изменено успешно";
        mUserManager=new UserManager(new
UserPreferences(getApplicationContext()));
        try {
            mUserManager.changeUserName(params[0]);
        } catch (UserException ex) {
            message=ex.getMessage();
            Log.e(TAG ,message);
        }
        Log.i(TAG,message);
        return message;
    }
    @Override
    protected void onPostExecute(String message) {
        if(pDialog!=null&&pDialog.isShowing()){
            pDialog.dismiss();
        }
        Toast
toast=Toast.makeText(mDialog.getContext(),message,Toast.LENGTH_
LONG);
        toast.setGravity(Gravity.CENTER, 0, 0);
        toast.show();
        mDialog.cancel();
    }
}
}

```

## Модуль SetUserPasswordDialog.java

```
package com.example.helponroad.ApplicationController.ManagementUser;

import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.DialogFragment;
import android.support.v7.app.AlertDialog;
import android.text.Editable;
import android.text.InputType;
import android.text.TextWatcher;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.CheckBox;
import android.widget.CompoundButton;
```

```
import android.widget.EditText;
import android.widget.Toast;
import com.example helponroad.NetworkOperations.HttpsQuery;
import
com.example helponroad.ManageObjectsOfModel.UserException;
import
com.example helponroad.ManageObjectsOfModel.UserManager;
import
com.example helponroad.ManageObjectsOfModel.UserPreference
s;
import com.example helponroad.ObjectsOfModel.User;
import com.example helponroad.R;

/**
 * Диалог для изменения пароля пользователя
 */
public class SetUserPasswordDialog extends DialogFragment {
    private UserManager mUserManager;
```

Рис.П2.53. Текст модуля SetUserPasswordDialog.java

## П2.53. Продолжение

```

private String mUserPassword;
private String mUserConfirmPassword;
private EditText mPasswordEditText;
private EditText mConfirmPasswordEditText;
private CheckBox mShowPasswordCheckBox;
private CheckBox mShowConfirmPasswordCheckBox;
private String TAG="SetUserPasswordDialog";

@NonNull
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    mUserManager=new UserManager(new
UserPreferences(getApplicationContext()));
    AlertDialog.Builder builder=new
AlertDialog.Builder(getApplicationContext());
    LayoutInflater inflater = getActivity().getLayoutInflater();
    final View
view=inflater.inflate(R.layout.fragment_set_password_dialog,null)
    ;
    mPasswordEditText
=(EditText)view.findViewById(R.id.edit_text_password);
    mPasswordEditText.addTextChangedListener(new
TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence
charSequence, int i, int i1, int i2) {
        }
        @Override
        public void onTextChanged(CharSequence charSequence,
int i, int i1, int i2) {
            if(charSequence.length()!=0)
                mUserPassword = charSequence.toString();
            else{
                mUserPassword =null;
            }
        }
        @Override
        public void afterTextChanged(Editable editable) {
        }
    });
    mConfirmPasswordEditText
=(EditText)view.findViewById(R.id.edit_text_confirm_password)
    ;
    mConfirmPasswordEditText.addTextChangedListener(new
TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence
charSequence, int i, int i1, int i2) {
        }
        @Override
        public void onTextChanged(CharSequence charSequence,
int i, int i1, int i2) {
            if(charSequence.length()!=0)
                mUserConfirmPassword = charSequence.toString();
            else{
                mUserConfirmPassword =null;
            }
        }
        @Override
        public void afterTextChanged(Editable editable) {
        }
    });
    mShowPasswordCheckBox=(CheckBox)view.findViewById(R.id.
check_box_showPassword);

    mShowPasswordCheckBox.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton
compoundButton, boolean checkState) {
            if(checkState){
                mPasswordEditText.setInputType(InputType.TYPE_CLASS_TEXT);
                mShowPasswordCheckBox.setText(getString(R.string.registration_
hide_password));
            }else{
                mPasswordEditText.setInputType(InputType.TYPE_CLASS_TEXT
InputType.TYPE_TEXT_VARIATION_PASSWORD);
                mShowPasswordCheckBox.setText(getString(R.string.registration_
show_password));
            }
        }
    });
    mShowConfirmPasswordCheckBox=(CheckBox)view.findViewById(R.id.
check_box_showConfirmPassword);

    mShowConfirmPasswordCheckBox.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton
compoundButton, boolean checkState) {
            if(checkState){
                mConfirmPasswordEditText.setInputType(InputType.TYPE_CLASS_TEXT);
                mShowConfirmPasswordCheckBox.setText(getString(R.string.registration_
hide_password));
            }else{
                mConfirmPasswordEditText.setInputType(InputType.TYPE_CLASS_TEXT
InputType.TYPE_TEXT_VARIATION_PASSWORD);
                mShowConfirmPasswordCheckBox.setText(getString(R.string.registration_
show_password));
            }
        }
    });
    builder.setCancelable(true);
    builder.setView(view);
    builder.setTitle(R.string.user_account_change_password);
    builder.setPositiveButton("OK", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            try {
                if (!HttpsQuery.isNetworkOnline(getActivity()))
                    throw new UserException("Отсутствует
подключение к сети Интернет."
+ "Проверьте Ваше подключение к сети
Интернет.");
            }
            if(mUserPassword==null||mUserConfirmPassword==null){
                throw new UserException("Все поля обязательны
для заполнения.");
            }
            if(mUserPassword.length()<4){
                throw new UserException("Пароль должен
содержать не менее 4 символов."
+ "Проверьте введенные данные.");
            }
            if(!mUserPassword.equals(mUserConfirmPassword)){
                throw new UserException("Значения полей
\"Пароль\" и \"Подтверждение пароля\""
+ "+ не совпадают. Проверьте введенные
данные.");
            }
            User user = mUserManager.getUser();
        }
    });
}

```

## П2.53. Продолжение

```

user.setPassword(mUserPassword);

new
SetUserPasswordDialog.ChangePasswordUserTask(SetUserPassw
ordDialog.this.getDialog()).execute(user);
}catch (UserException ex){
    Toast
toast=Toast.makeText(getApplicationContext(),ex.getMessage(),Toast.LENGTH
TH_LONG);
    toast.setGravity(Gravity.CENTER, 0, 0);
    toast.show();
}
})
builder.setNegativeButton("Отмена", new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        SetUserPasswordDialog.this.getDialog().cancel();
    }
});
return builder.create();
}

/**
 * Фоновый поток для отправки запроса на сервер на
изменение пароля пользователя
*/
private class ChangePasswordUserTask extends
AsyncTask<User,Void,String> {
    private UserManager mUserManager;
    private Dialog mDialog;
    private ProgressDialog pDialog;

    public ChangePasswordUserTask(Dialog dialog) {
        mDialog = dialog;
    }

    @Override
    protected void onPreExecute() {
        pDialog = new ProgressDialog(getApplicationContext());
        pDialog.setMessage("Выполняется изменение
пароля...");
```

параметров");

```

        pDialog.setIndeterminate(false);
        pDialog.setCancelable(false);
        pDialog.show();
    }

    @Override
    protected String doInBackground(User... params) {
        String message="Пароль пользователя изменен
успешно";
        mUserManager=new UserManager(new
UserPreferences(getApplicationContext()));

        try {
            mUserManager.changeUserPassword(params[0]);
        }
        catch (UserException ex){
            message=ex.getMessage();
            Log.e(TAG ,message);
        }
        Log.i(TAG,message);
        return message;
    }

    @Override
    protected void onPostExecute(String message) {
        if(pDialog!=null&&pDialog.isShowing()){
            pDialog.dismiss();
        }
        Toast
toast=Toast.makeText(mDialog.getContext(),message,Toast.LENGTH
TH_LONG);
        toast.setGravity(Gravity.CENTER, 0, 0);
        toast.show();
        mDialog.cancel();
    }
}
```

## Модуль UserAccountActivity.java

```
package com.example helponroad ApplicationController ManagementUser;
import android.support.v4.app.Fragment;
import com.example helponroad CommonUserInterfaceComponentsLibrary.SingleFragmentActivity;

/**
 * Активность для изменения информации об авторизованном
```

```
пользователе
*/
public class UserAccountActivity extends SingleFragmentActivity
{
    @Override
    public Fragment createFragment() {
        return UserAccountFragment.newInstance();
    }
}
```

Рис.П2.54. Текст модуля UserAccountActivity.java

## Модуль UserAccountFragment.java

```
package com.example.helponroad.ApplicationController.ManagementUser;  
import android.app.AlertDialog;  
import android.content.ActivityNotFoundException;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.content.pm.PackageManager;
```

```
import android.graphics.Bitmap;  
import android.net.Uri;  
import android.os.AsyncTask;  
import android.os.Bundle;  
import android.provider.MediaStore;
```

Рис.П2.55. Текст модуля UserAccountFragment.java

## П2.55. Продолжение

```

import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.Fragment;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AlertDialog;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;
import com.example.helponroad.NetworkOperations.HttpsQuery;
import com.example.helponroad.ManageObjectsOfModel.PhotoManager;
import com.example.helponroad.ManageObjectsOfModel.UserException;
import com.example.helponroad.ManageObjectsOfModel.UserManager;
import com.example.helponroad.ManageObjectsOfModel.UserPreference
s;
import com.example.helponroad.ObjectsOfModel.User;
import com.example.helponroad.R;
import java.io.File;
import java.io.IOException;
import static android.app.Activity.RESULT_OK;

/**
 * Фрагмент для изменения информации об авторизованном
 * пользователе
 */
public class UserAccountFragment extends Fragment
    implements SetDialogListener {
    private static final String KEY_BITMAP = "bitmap";
    private static final String TAG = "UserAccountFragment";
    private final int TAKE_TO_THE_CAMERA = 0;//снять на
камеру
    private final int SELECT_FROM_ALBUM=1;//выбрать из
альбома
    private final int CROP_PHOTO =2;//обрезать фотографию
    private final int
PERMISSION_READ_AND_WRITE_EXTERNAL_STORAGE
=1;//обрезать фотографию
    private final static String
SET_USER_NAME_DIALOG="setUserNameDialog";
    private final static String
SET_USER_PASSWORD_DIALOG="setUserPasswordDialog";
    private UserManager mUserManager;
    private PhotoManager mPhotoManager;
    private TextView mLoginTextView;
    private TextView mLogOutTextView;
    private TextView mNameTextView;
    private TextView mPasswordTextView;
    private SetUserNameDialog mSetUserNameDialog;
    private SetUserPasswordDialog mSetUserPasswordDialog;
    private AlertDialog.Builder mLogOutAlertDialog;
    private AlertDialog.Builder mSelectCameraOrAlbumDialog;
    private ImageButton mUserPhotoButton;
    private ImageButton mSaveUserImageButton;
    private ImageButton mDeleteUserImageButton;
    private Bitmap mBitmap;

    public static UserAccountFragment newInstance() {
        return new UserAccountFragment();
    }

    //Возникает при закрытии диалогов

    @Override
    public void onCancelDialog() {
        String userName=mUserManager.getUser().getName();
        if(userName!=null)
            mNameTextView.setText(userName);
    }
}

@Override
public void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setRetainInstance(true);
}

@Override
public void onViewStateRestored(@Nullable Bundle
savedInstanceState) {
    super.onViewStateRestored(savedInstanceState);
    if (savedInstanceState != null){
        mBitmap =
savedInstanceState.getParcelable(KEY_BITMAP);
        //Если пользователем было выбрано новое изображение
        if(mBitmap!=null) {
            mUserPhotoButton.setImageBitmap(mBitmap);
        }
    }
}

@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    //Если пользователь не выбрал новое изображение
    outState.putParcelable(KEY_BITMAP, mBitmap);
}

@Override
public View onCreateView(LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
    mUserManager=new UserManager(new
UserPreferences(getActivity()));

    String userName=mUserManager.getUser().getName();
    View
view=inflater.inflate(R.layout.fragment_user_account,container,fal
se);
    mSetUserNameDialog=new SetUserNameDialog();
    mSetUserPasswordDialog=new SetUserPasswordDialog();
    mLoginTextView=(TextView)
view.findViewById(R.id.text_view_login);

    mLoginTextView.setText(mUserManager.getUser().getLogin());
    mLogOutTextView=(TextView)view.findViewById(R.id.text_vie
w_log_out);
    mLogOutTextView.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
            mLogOutAlertDialog.show();
        }
    });

    mNameTextView=(TextView)view.findViewById(R.id.text_view
_name);
    mNameTextView.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
            mSetUserNameDialog.show(getFragmentManager(),SET_USER_
NAME_DIALOG);
        }
    });
    if(userName!=null)
        mNameTextView.setText(userName);
    mPasswordTextView=(TextView)view.findViewById(R.id.text_v
iew_password);
    mPasswordTextView.setOnClickListener(new
View.OnClickListener() {

```

## П2.55. Продолжение

```

@Override
public void onClick(View view) {
    mSetUserPasswordDialog.show(getFragmentManager(),SET_USE
R_PASSWORD_DIALOG);
}
//Диалог для выхода из учетной записи
mLogOutAlertDialog=new
AlertDialog.Builder(getContext());
mLogOutAlertDialog.setTitle(R.string.user_account_is_log_out);
mLogOutAlertDialog.setCancelable(true);
mLogOutAlertDialog.setPositiveButton("Ok", new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface,
int i) {
        mUserManager.logOut();
        Toast toast=Toast.makeText(getApplicationContext(),"Вы
вышли из приложения",Toast.LENGTH_LONG);
        toast.setGravity(Gravity.CENTER, 0, 0);
        toast.show();
        getActivity().finish();
    }
});
mLogOutAlertDialog.setNegativeButton("Отмена", new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface,
int i) {
        dialogInterface.cancel();
    }
});
//Диалог для выбора способа добавления фотографии
mSelectCameraOrAlbumDialog=new
AlertDialog.Builder(getContext());
mSelectCameraOrAlbumDialog.setTitle(R.string.user_account_sel
ect_camera_or_album);
mSelectCameraOrAlbumDialog.setCancelable(true);
//Выбираем фото из альбома
mSelectCameraOrAlbumDialog.setPositiveButton("Из
альбома", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        try {
            Intent selectFromAlbumIntent = new
Intent(Intent.ACTION_PICK);
            if(selectFromAlbumIntent.resolveActivity(getActivity()).getPackag
eManager()==null){
                throw new ActivityNotFoundException("Галерея
недоступна.");
            }
            selectFromAlbumIntent.setType("image/*");
            startActivityForResult(selectFromAlbumIntent,SELECT_FROM_
ALBUM);
        }catch (ActivityNotFoundException ex){
            Toast
toast=Toast.makeText(getApplicationContext(),ex.getMessage(),Toast.LENGTH
TH_LONG);
            toast.setGravity(Gravity.CENTER, 0, 0);
            toast.show();
        }
    }
});
//снимаем на камеру
mSelectCameraOrAlbumDialog.setNegativeButton("Снять
на камеру", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        try {
            Intent takeToCameraIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            // проверяем, что есть приложение способное обработать
интент
if
(takeToCameraIntent.resolveActivity(getActivity()).getPackageMa
nager()) == null{
                throw new IOException ("Камера недоступна");
            }
//создаем файл
mPhotoManager=new PhotoManager();
File photoFile=mPhotoManager.getFile();
if(photoFile==null){
                throw new IOException ("Возникла внутренняя
ошибка при создании фотографии");
            }
takeToCameraIntent.putExtra(MediaStore.EXTRA_OUTPUT,
Uri.fromFile(photoFile));
startActivityForResult(takeToCameraIntent,
TAKE_TO_THE_CAMERA);
}catch (IOException ex){
            Toast
toast=Toast.makeText(getApplicationContext(),ex.getMessage(),Toast.LENGTH
TH_LONG);
            toast.setGravity(Gravity.CENTER, 0, 0);
            toast.show();
        }
    }
});
mUserPhotoButton=(ImageButton)view.findViewById(R.id.button
_user_photo);
if(mUserManager.getUser().getIsImage()){
    if(mUserManager.getUser().getImage()!=null) {
        mUserPhotoButton.setImageURI(Uri.parse(mUserManager.getUse
r().getImage()));
    }
}
mUserPhotoButton.setOnClickListener(new
View.OnClickListener() {
    //Создание новой фотографии
    @Override
    public void onClick(View view) {
        mSelectCameraOrAlbumDialog.show();
    }
});
mSaveUserImageButton=(ImageButton)view.findViewById(R.id.
button_save_user_photo);
mSaveUserImageButton.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {
        try {
            if(!HttpsQuery.isNetworkOnline(getActivity())){
                throw new UserException("Отсутствует
подключение к сети Интернет."
+ "Проверьте Ваше подключение к сети
Интернет.");
            }
            if(mBitmap!=null) {
                String nameUserFile =
mUserManager.generateNameUserFile(getActivity());
                File file =
PhotoManager.generateFileFromBitmap(mBitmap, nameUserFile);
                new
ChangeImageUserTask(file.getAbsolutePath()).execute(mUserMa
nager.getUser());
            }else{
                Toast toast=Toast.makeText(getApplicationContext(),"Для
изменения фотографии необходимо выбрать новое
изображение.",Toast.LENGTH_LONG);
                toast.setGravity(Gravity.CENTER, 0, 0);
                toast.show();
            }
        }
    }
});

```

## П2.55. Продолжение

## П2.55. Продолжение

```

* Получение изображения с камеры и его кадрирование
*/
private void onGetPictureFromCamera() {
    File userPhoto = mPhotoManager.getFile();
    if (userPhoto.exists()) {

        PhotoManager.addPhotoIntoGallery(userPhoto.getPath(),getActivity());
        mBitmap =
        PhotoManager.getScaledBitmap(userPhoto.getPath(),getActivity())
        ;
        mUserPhotoButton.setImageBitmap(mBitmap);
        PhotoManager.performCrop(UserAccountFragment.this,getContext(), CROP_PHOTO,
            Uri.fromFile(mPhotoManager.getFile()), 256, 256);
    }
}

/**
 * Задание для изменения изображения пользователя
 */

private class ChangeImageUserTask extends AsyncTask<User,Void,String> {

    private UserManager mUserManager;
    private ProgressDialog pDialog;
    private String mFilePath;
    public ChangeImageUserTask(String filepath) {
        mFilePath=filepath;
    }
    /**
     * Перед началом фонового потока Show Progress Dialog
     */
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        pDialog = new ProgressDialog(getActivity());
        pDialog.setMessage("Выполняется изменение
изображения...");
        pDialog.setIndeterminate(false);
        pDialog.setCancelable(false);
        pDialog.show();
    }

    @Override
    protected String doInBackground(User... params) {
        String message="Изображение пользователя изменено
успешно";
        mUserManager=new UserManager(new
UserPreferences(getContext()));
        try {
            //Если изменение пользовательского файла прошло
успешно
            mUserManager.changeUserImage(params[0],mFilePath);
        }
        catch (UserException ex){
            message=ex.getMessage();
            Log.e(TAG ,message);
        }
        Log.i(TAG,message);
        return message;
    }
}

@Override
protected void onPostExecute(String message) {
    if(pDialog!=null&&pDialog.isShowing()){
        pDialog.dismiss();
    }
    Toast
toast=Toast.makeText(getApplicationContext(),message,Toast.LENGTH_LONG);
    toast.setGravity(Gravity.CENTER, 0, 0);
    toast.show();
}

/**
 * Задание для удаления изображения пользователя
 */
private class DeleteImageUserTask extends AsyncTask<User(Void, String> {

    private UserManager mUserManager;
    private ProgressDialog pDialog;
    /**
     * Перед началом фонового потока Show Progress Dialog
     */
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        pDialog = new ProgressDialog(getApplicationContext());
        pDialog.setMessage("Выполняется удаление
изображения...");
        pDialog.setIndeterminate(false);
        pDialog.setCancelable(false);
        pDialog.show();
    }

    @Override
    protected String doInBackground(User... params) {
        String message="Изображение пользователя удалено
успешно";
        mUserManager=new UserManager(new
UserPreferences(getApplicationContext()));
        try {
            mUserManager.deleteUserImage(params[0]);
        }
        catch (UserException ex){
            message=ex.getMessage();
            Log.e(TAG ,message);
        }
        Log.i(TAG,message);
        return message;
    }

    @Override
    protected void onPostExecute(String message) {
        if(pDialog!=null&&pDialog.isShowing()){
            pDialog.dismiss();
        }
        mUserPhotoButton.setImageURI(null);
        mBitmap=null;
        Toast
toast=Toast.makeText(getApplicationContext(),message,Toast.LENGTH_LONG);
        toast.setGravity(Gravity.CENTER, 0, 0);
        toast.show();
    }
}

```

## Модуль UserSettingsActivity.java

```

package
com.example.helponroad.ApplicationController.ManagementUser;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import
com.example.helponroad.CommonUserInterfaceComponentsLibra
ry.SingleFragmentActivity;
public class UserSettingsActivity extends SingleFragmentActivity
{
    @Override
    public Fragment createFragment() {
        return UserSettingsFragment.newInstance();
    }
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}

```

Рис.П2.56. Текст модуля UserSettingsActivity.java

## Модуль UsersSettingsFragment.java

```

package
com.example.helponroad.ApplicationController.ManagementUser;
import android.content.Intent;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.provider.Settings;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v7.widget.SwitchCompat;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.Spinner;
import
com.example.helponroad.ManageObjectsOfModel.UserPreference
s;
import com.example.helponroad.R;
import
com.example.helponroad.ApplicationController.ManagementListR
equests.NewRequestFetcherService;
import com.google.android.gms.gcm.GcmNetworkManager;
import java.util.ArrayList;
import java.util.List;

/**
 * Фрагмент для задания настроек приложения
 */
public class UserSettingsFragment extends Fragment {
    public static UserSettingsFragment newInstance() {
        return new UserSettingsFragment();
    }
    private SwitchCompat mGetNotificationsSwitchCompat;
    private Spinner mRadiusNotificationSpinner;
    private Button mAppPermissionsButton;
    private UserPreferences mUserPreferences;

    private Spinner mFrequencyNotificationSpinner;
    private GcmNetworkManager mGcmNetworkManager;
    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRetainInstance(true);
    }
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        mGcmNetworkManager=GcmNetworkManager.getInstance(getActivity());
        mUserPreferences=new UserPreferences(getActivity());
        View
view=inflater.inflate(R.layout.fragment_user_settings,container,fal
se);
        mGetNotificationsSwitchCompat=(SwitchCompat)view.findViewById(R.id.switch_compat_get_notifications);
        mGetNotificationsSwitchCompat.setChecked(mUserPreferences.g
etStoredIsGetNotificationsParameter());

        mGetNotificationsSwitchCompat.setOnCheckedChangeListener(n
ew CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton
compoundButton, boolean state) {
                if(state) {
                    NewRequestFetcherService.startNewRequestFetcherService(getCo
nnect().getApplicationContext());
                }else {
                    NewRequestFetcherService.stopNewRequestFetcherService(getCo
nnect().getApplicationContext());
                }
            }
        });
        mRadiusNotificationSpinner=(Spinner)view.findViewById(R.id.sp
inner_radius_notifications);
        mRadiusNotificationSpinner.setAdapter(getSpinnerRadiusNotifica
tionsAdapter());
        //Задаем выбранное значение радиуса получения заявок(-
1=все заявки или 1-100 км),
        // по умолчанию устанавливается первое выбранное
        // значение( получать все заявки)

        if(mUserPreferences.getStoredRadiusNotifications()!=-1){
            mRadiusNotificationSpinner.setSelection(mUserPreferences.getSt
oredRadiusNotifications());
        }
        mRadiusNotificationSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> adapterView,
View view,
int selectedItemId, long selectedId)
            {
                //Выбрано значение (получать все заявки)
                if(selectedItemId==0){

```

Рис.П2.57. Текст модуля UsersSettingsFragment.java

## П2.57. Продолжение

```

        mUserPreferences.setStoredRadiusNotifications(-
1);//устанавливаем -1-получать все заявки
    }else {
        mUserPreferences.setStoredRadiusNotifications(selectedItemPositi-
on);//от 1 до 100 км
    }
}
@Override
public void onNothingSelected(AdapterView<?>
adapterView) {
}
});
mFrequencyNotificationSpinner=(Spinner)view.findViewById(R.i-
d.spinner_frequency_notification);
mFrequencyNotificationSpinner.setAdapter(getSpinnerFrequency
NotificationsAdapter());
mFrequencyNotificationSpinner.setSelection(mUserPreferences.ge-
tStoredFrequencyNotifications()-1);

mFrequencyNotificationSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> adapterView,
View view, int selectedItemId, long l) {
        mUserPreferences.setStoredFrequencyNotifications(selectedItemId+1);

        if(mUserPreferences.getStoredIsGetNotificationsParameter()) {
            NewRequestFetcherService.stopNewRequestFetcherService(getApplicationContext());
        }

        NewRequestFetcherService.startNewRequestFetcherService(getApplicationContext());
    }
    @Override
    public void onNothingSelected(AdapterView<?>
adapterView) {
    });
});
mAppPermissionsButton=(Button)view.findViewById(R.id.button_
_app_permissions);

if(Build.VERSION.SDK_INT<
Build.VERSION_CODES.M){
    mAppPermissionsButton.setVisibility(View.INVISIBLE);
}
mAppPermissionsButton.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new
Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS
,
Uri.parse("package:"+getActivity().getPackageName()));
        startActivity(intent);
    }
});
return view;
}

/**
 * Создаем адаптер для выпадающего списка
RadiusNotificationsSpinner
 * @return - адаптер для выпадающего списка
RadiusNotificationsSpinner
 */
private ArrayAdapter<String>
getSpinnerRadiusNotificationsAdapter(){
//Создаем и присваиваем адаптер для spinner(генерируем
значения и заполняем spinner)
List<String>
spinnerValues=this.generateValuesRadiusNotificationSpinner();
ArrayAdapter<String> spinnerAdapter=new
ArrayAdapter<>(getContext(),
        android.R.layout.simple_spinner_item,spinnerValues);
spinnerAdapter.setDropDownViewResource(android.R.layout.sim-
ple_spinner_dropdown_item);
return spinnerAdapter;
}

/**
 * Создаем адаптер для выпадающего списка
FrequencyNotificationsSpinner
 * @return - адаптер для выпадающего списка
FrequencyNotificationsSpinner
 */
private ArrayAdapter<String>
getSpinnerFrequencyNotificationsAdapter(){
//Создаем и присваиваем адаптер для spinner(генерируем
значения и заполняем spinner)
List<String>
spinnerValues=this.generateValuesFrequencyNotificationSpinner()
;

        ArrayAdapter<String> spinnerAdapter=new
ArrayAdapter<>(getContext(),
        android.R.layout.simple_spinner_item,spinnerValues);
spinnerAdapter.setDropDownViewResource(android.R.layout.sim-
ple_spinner_dropdown_item);

return spinnerAdapter;
}

/**
 * Создаем строковый список значений выпадающего
списка RadiusNotificationsSpinner
 * @return - строковый массив значений выпадающего
списка RadiusNotificationsSpinner
 */
private List<String>
generateValuesRadiusNotificationSpinner(){

List<String> valuesList=new ArrayList<>();
valuesList.add("Получать все заявки");

for(int i=1;i<=100;i++){
    String value=i+" км.";
    valuesList.add(value);
}
return valuesList;
}

private List<String>
generateValuesFrequencyNotificationSpinner(){

List<String> valuesList=new ArrayList<>();
for(int i=1;i<=60;i++){
    String value=i+" мин.";
    valuesList.add(value);
}
return valuesList;
}

@Override
public void onActivityResult(int requestCode, int resultCode,
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
}
}
}

```

## Пакет NetworkOperation.java

### Модуль HttpsQuery.java

```

package com.example.helponroad.NetworkOperations;
import android.content.Context;
import android.net.ConnectivityManager;
import android.util.Log;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.BufferedWriter;
import java.io.ByteArrayOutputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.SocketTimeoutException;
import java.net.URL;
import java.util.Iterator;
import javax.net.ssl.HttpsURLConnection;

/*
 *
 * Класс HttpsQuery
 * Предназначен для создания https-запросов
 *
 */
public class HttpsQuery {
    private static final String TAG="HttpsQuery";
    private static int CONNECT_TIMEOUT=1000*10;//время ожидания ответа от сервера
    private static int READ_TIMEOUT=1000*30;
    // Конец строки
    private final static String lineEnd = "\r\n";
    // Два тире
    private final static String twoHyphens = "--";
    // Разделитель
    private final static String boundary = "----WebKitFormBoundary9xFB2hiUhzqbBQ4M";
    // Ключ, под которым файл передается на сервер
    private final static String FORM_FILE_NAME = "imgfile";
    /**
     * Post-запрос для отправки данных на сервер
     * @param urlString-URL-строка запроса
     * @param data-отправляемые параметры
     * @return-возвращает ответ от сервера
     */
    public static byte[] postQuery(String urlString,JSONObject data)throws IOException,JSONException{
        URL url = new URL(urlString);
        HttpURLConnection.setFollowRedirects(false);
        HttpURLConnection conn = (HttpURLConnection)
        url.openConnection();
        Log.i(TAG,"Connection was opening");

        conn.setConnectTimeout(HttpsQuery.CONNECT_TIMEOUT);
        conn.setReadTimeout(HttpsQuery.READ_TIMEOUT);
        try {
            byte[] sendData=prepareData(data).getBytes("UTF-8");
            Log.e("Data Input",data.toString());
            conn.setRequestMethod("POST");
            conn.setDoOutput(true);
            conn.setDoInput(true);
            conn.setRequestProperty("Content-
Length",""+Integer.toString(sendData.length));
            conn.getOutputStream().write(sendData);
            if(conn.getResponseCode()!=
HttpURLConnection.HTTP_OK)
                throw new IOException(conn.getResponseMessage());
            with "+url.toString());
        }
        catch (IOException ex){
            throw new IOException("Превышено время ожидания от
сервера");
        }
        finally {
            conn.disconnect();
        }
    }

    /**
     * post-запрос для отправки файла на сервер с указанием
     * post-параметров
     * @param urlString-URL-строка запроса
     * @param data-параметры запроса
     * @param file-отправляемый файл в файловой системе
     * устройства
     * @return - возвращает ответ от сервера
     */
    public static byte[] postQuery(String urlString,JSONObject
data,File file)
        throws IOException,JSONException{
        // Переменные для считывания файла в оперативную
        //память
        int bytesAvailable, bufferSize,bytesRead;
        byte[] buffer;
        int maxBufferSize = 1*1024*1024;
        //-----
        // Создание соединения для отправки файла
        URL uploadUrl = new URL(urlString);
        HttpsURLConnection.setFollowRedirects(false);
        HttpURLConnection conn = (HttpURLConnection)
        uploadUrl.openConnection();
        Log.i(TAG,"Connection was opening");
        try {
            conn.setConnectTimeout(HttpsQuery.CONNECT_TIMEOUT);
            conn.setReadTimeout(HttpsQuery.READ_TIMEOUT);
            // Разрешение ввода соединению
            conn.setDoInput(true);
            // Разрешение вывода соединению
            conn.setDoOutput(true);
            // Отключение кэширования
            conn.setUseCaches(false);
            // Задание запросу типа POST
            conn.setRequestMethod("POST");
            // Задание необходимых свойств запросу
            conn.setRequestProperty("Connection", "Keep-Alive");
            conn.setRequestProperty("Content-Type", "multipart/form-
data;boundary="+boundary);
            // Создание потока для записи в соединение
            DataOutputStream outputStream = new
DataOutputStream(conn.getOutputStream());
        }
    }
}

```

Рис.П2.58. Текст модуля HttpsQuery.java

## П2.58. Продолжение

```

-----  

//Добавление параметров запроса  

Iterator<String> iter = data.keys();  

while(iter.hasNext()) {  

    String key = iter.next();  

    BufferedWriter outputStream2 = new  

BufferedWriter(new OutputStreamWriter(outputStream, "UTF-  

8"));  

    HttpsQuery.addFormField(outputStream2,key,  

data.getString(key));  

}  

-----  

//Формирование multipart контента  

//Начало контента  

outputStream.writeBytes(twoHyphens + boundary +  

lineEnd);  

//Заголовок элемента формы  

outputStream.writeBytes("Content-Disposition: form-data;  

name=\"" +  

    FORM_FILE_NAME + "\"; filename=\"" +  

file.getAbsolutePath() + "\"" + lineEnd);  

//Тип данных элемента формы  

outputStream.writeBytes("Content-Type: image/jpeg" +  

lineEnd);  

//Конец заголовка  

outputStream.writeBytes(lineEnd);  

-----  

//Считывание файла в оперативную память и загрузка в  

отправляемый поток  

FileInputStream fileInputStream = new  

FileInputStream(file);  

bytesAvailable = fileInputStream.available();  

bufferSize = Math.min(bytesAvailable, maxBufferSize);  

buffer = new byte[bufferSize];  

bytesRead = fileInputStream.read(buffer, 0, bufferSize);  

while (bytesRead > 0) {  

    outputStream.write(buffer, 0, bufferSize);  

    bytesAvailable = fileInputStream.available();  

    bufferSize = Math.min(bytesAvailable, maxBufferSize);  

    bytesRead = fileInputStream.read(buffer, 0, bufferSize);  

}  

//Конец элемента формы  

outputStream.writeBytes(lineEnd);  

outputStream.writeBytes(twoHyphens + boundary +  

twoHyphens + lineEnd);  

//получаем код ответа от сервера  

if(conn.getResponseCode()!=  

HttpURLConnection.HTTP_OK){  

    throw new IOException(conn.getResponseMessage()+"  

with "+uploadUrl.toString());  

}  

//Закрытие соединений и потоков  

fileInputStream.close();  

outputStream.flush();  

outputStream.close();  

Log.i(TAG,"Запрос отправлен успешно"+urlString);
-----  

//Получаем строковый ответ от сервера  

InputStream in = conn.getInputStream();
//поток для записи ответа от сервера  

ByteArrayOutputStream out = new  

ByteArrayOutputStream();
int inputBytesRead = 0;
byte[] inputBuffer = new byte[1024];
while ((inputBytesRead = in.read(inputBuffer)) > 0) {
    out.write(inputBuffer, 0, inputBytesRead);
}
return out.toByteArray();
}catch (SocketTimeoutException ex){
    throw new IOException("Превышено время ожидания от  

сервера");
}

}
finally {
    conn.disconnect();
}
}

//добавление параметра при отправке файла
private static void addFormField(BufferedWriter dos, String  

parameter, String value) throws IOException{
    dos.write(twoHyphens + boundary + lineEnd);
    dos.write("Content-Disposition: form-data;  

name=\"" + parameter + "\"" + lineEnd);
    dos.write("Content-Type: text/plain; charset=UTF-8" +  

lineEnd);
    dos.write(lineEnd);
    dos.write(value + lineEnd);
    dos.flush();
}

//подготовка отправляемых параметров на сервер
private static String prepareData(JSONObject data) throws  

JSONException{
    String postData="";
    Iterator<String> iter = data.keys();
    while(iter.hasNext()){
        String key = iter.next();
        if(iter.hasNext()){
            postData+=key+"="+data.getString(key)+"&";
        }else{
            postData+=key+"="+data.getString(key);
        }
    }
    return postData;
}
public static boolean isNetworkOnline(Context activity){
    ConnectivityManager  

cm=(ConnectivityManager)activity.getSystemService(Context.CO  

NNECTIVITY_SERVICE);
    return cm.getActiveNetworkInfo()!=null;
}

/**
 * Метод для выполнения GET-запроса
 * @param urlSpec -URL-запроса
 * @return -массив байтов
 * @throws IOException
 */
public static byte[] getUrlBytes(String urlSpec) throws  

IOException {
    URL url = new URL(urlSpec);

    HttpsURLConnection.setFollowRedirects(false);
    HttpURLConnection connection =
(HttpsURLConnection)url.openConnection();
    Log.i(TAG,"Connection was opening");

    connection.setConnectTimeout(HttpsQuery.CONNECT_TIMEOUT);
    connection.setReadTimeout(HttpsQuery.READ_TIMEOUT);
    try {
        ByteArrayOutputStream out = new  

ByteArrayOutputStream();
        InputStream in = connection.getInputStream();
        if (connection.getResponseCode() !=  

HttpURLConnection.HTTP_OK) {
            throw new  

IOException(connection.getResponseMessage() +  

": with " +
urlSpec);
        }
        int bytesRead = 0;
    }
}

```

## П2.58. Продолжение

```

        byte[] buffer = new byte[1024];

    while ((bytesRead = in.read(buffer)) > 0) {
        out.write(buffer, 0, bytesRead);
    }
    out.close();
    return out.toByteArray();
} catch (SocketTimeoutException ex){
    throw new IOException("Превышено время ожидания от
сервера");
}
finally {
    connection.disconnect();
}

    }

    }

    */

    * Метод для преобразования байтов полученных от
getUrlBytes строку
    * @param urlSpec - массив байтов
    * @return - строковое представление запроса
    * @throws IOException
    */
    public static String getUrlString(String urlSpec) throws
IOException {
    return new String(getUrlBytes(urlSpec));
}
}
}

```

## Модуль MySQLDataBaseSchema.java

```

package com.example.helponroad.NetworkOperations;

/**
 * Схема базы данных MySQL
 * Created by Олег on 13.04.2017.
 */

public class MySQLDataBaseSchema {
    /**
     * Структура системной таблицы для хранения данных
     * пользователей (водителей)
     */
    public static final class UsersTable {
        public static final String NAME = "users";
        public static final class Cols {
            public static final String ID = "id"; //первичный ключ
            //системное поле
            public static final String LOGIN = "login"; //логин
            public static final String PASSWORD =
                "password"; //пароль
            public static final String NAME = "name"; //имя
            public static final String IMAGE =
                "image"; //необязательное поле
            public static final String CREATE_DATE =
                "create_date"; //дата создания
            public static final String UPDATE_DATE =
                "update_date"; //дата обновления
        }
    }

    /**
     * Структура таблицы для хранения данных заявок о
     * помощи/
     */
    public static final class RequestsTable {
        public static final String NAME = "requests";
        public static final class Cols {
            public static final String ID = "id"; //первичный ключ
            public static final String MESSAGE =
                "message"; //сообщение
            public static final String LONGITUDE =
                "longitude"; //местоположение:широта
            public static final String LATITUDE =
                "latitude"; //местоположение:долгота
            public static final String IMAGE =
                "image"; //изображение (NULL)
            public static final String CREATE_DATE =
                "create_date"; //дата создания
            public static final String UPDATE_DATE =
                "update_date"; //дата изменения(NULL)
            public static final String USER_ID =
                "id_user"; //идентификатор пользователя
            public static final String STATUS_ID =
                "id_status"; //идентификатор статуса
        }
    }

    /**
     * Структура таблицы для хранения статуса заявки*/
    public static final class StatusesTable {
        public static final String NAME = "statuses";
        public static final class Cols {
            public static final String ID = "id"; //первичный ключ
            //системное поле
            public static final String VALUE = "value"; //значение
            //статуса
        }
    }

    /**
     * Структура таблицы для хранения данных предложений
     * помощи/
     */
    public static final class OffersTable {
        public static final String NAME = "offers";
        public static final class Cols {
            public static final String ID = "id"; //первичный ключ
            //системное поле
            public static final String MESSAGE =
                "message"; //сообщение
            public static final String CREATE_DATE =
                "create_date"; //дата создания
            public static final String UPDATE_DATE =
                "update_date"; //дата изменения (NULL)
            public static final String REQUEST_ID =
                "id_request"; //идентификатор заявки
            public static final String USER_ID =
                "id_user"; //идентификатор пользователя
        }
    }

    /**
     * Структура таблицы для хранения списка устройств*/
    public static final class DevicesTable {
        public static final String NAME = "devices";
        public static final class Cols {
            public static final String ID = "id"; //первичный ключ
            //системное поле
            public static final String TOKEN = "token"; //токен
            //устройства
            public static final String RADIUS = "radius"; //радиус
            //получения заявок
            public static final String USER_ID =
                "id_user"; //идентификатор пользоваптеля
        }
    }
}
}

```

Рис.П2.59. Текст модуля MySQLDataBaseSchema.java

## Модуль QueryStrings.java

```

package com.example.helponroad.NetworkOperations;
import
com.example.helponroad.LocationManager.LocationListenerGPSe
rvices;
public class QueryStrings {
    //интервал для обновления местоположения
    public static final int UPDATE_LOCATION_INTERVAL=
    LocationListenerGPSServices.MILLISECONDS_PER_SECOND*6
    0;
    //Строки- названия пользовательских настроек
    public static final String LAST_REQUEST_ID="lastRequestId";//идентификатор
    последней загруженной заявки
    public static final String LAST_UPLOAD_DATE
    ="lastUploadDate";//дата последней загрузки на устройство
    public static final String LAST_PREVIOUS_UPLOAD_DATE
    ="lastPreviousUploadDate";//дата предпоследней загрузки на
    устройство
    public static final String IS_GET_NOTIFICATION
    IS_GET_NOTIFICATION="isGetNotifications";//получает ли
    пользователь уведомления
    public static final String RADIUS_NOTIFICATIONS
    RADIUS_NOTIFICATIONS="radiusNotifications";//радиус
    получения уведомлений
    public static final String FREQUENCY_NOTIFICATIONS
    FREQUENCY_NOTIFICATIONS="frequencyNotifications";//частота
    получения уведомлений
    public static final String IS_SHOW_ROUTE_MESSAGE
    IS_SHOW_ROUTE_MESSAGE="isShowRouteMessage";//часто
    та получения уведомлений

    //Параметры для соединения с сервером
    public static final String API_URL="https://helponroad.000webhostapp.com/api/";
    public static final String ANDROID_KEY_NAME="key";

    //Имена скриптов для соединения с сервером
}

//Скрипты для работы с заявками
public static final String ADD_NEW_REQUEST_SCRIPT="requests.addRequest.php";
    public static final String GET_ACTUAL_REQUESTS_SCRIPT="requests.getActualReque
sts.php";
    public static final String LOAD_REQUEST_IMAGE_SCRIPT="requests.getImage.php";
    public static final String GET_ACTUAL_AND_CHANGED_REQUESTS_SCRIPT="requ
ests.getActualAndChangedRequests.php";
    public static final String UPDATE_REQUEST_STATUS_SCRIPT="requests.updateReque
stStatus.php";

//Скрипты для работы с пользователями
public static final String ADD_NEW_USER_SCRIPT="users.registerUser.php";
    public static final String AUTHORIZE_USER_SCRIPT="users.authorizeUser.php";
    public static final String CHANGE_USER_NAME_SCRIPT="users.changeName.php";
    public static final String CHANGE_USER_PASSWORD_SCRIPT="users.changePassword
.php";
    public static final String CHANGE_USER_IMAGE_SCRIPT="users.changeImage.php";
    public static final String LOAD_USER_IMAGE_SCRIPT="users.getImage.php";
    public static final String DELETE_USER_IMAGE_SCRIPT="users.deleteImage.php";

//Скрипты для работы с предложениями помощи
public static final String ADD_NEW_OFFER_SCRIPT="offers.addOffer.php";
    public static final String GET_OFFERS_SCRIPT="offers.getOffers.php";
    public static final String LAST_OFFER_ID="id_offer";//идентификатор последней
    загруженной заявки
}

```

Рис.П2.59. Текст модуля QueryStrings.java

## Пакет ServerClasses

### Модуль api.users.php

```

require_once 'api.files.php';
require_once 'config.php';

class UserAPI{
    //регистрация нового пользователя

    public function addUser()
    {
        try{
            $check_ipput_field=isset($_POST['key']) &&
            isset($_POST['login']) && isset($_POST['password']);
            if (!$check_ipput_field) {
                throw new Exception("В запросе не заданы обязательные поля",
0);
            }
            $key=mb_convert_encoding($_POST['key'], "UTF-8");
            $login=mb_convert_encoding($_POST['login'], "UTF-8");
            $password=mb_convert_encoding($_POST['password'], "UTF-8");
            if(strcmp($key,ANDROID_KEY)!=0){
                throw new Exception("Неверное значение ANDROID KEY.", 1);
            }
            $mysqli=new
            mysqli(DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE);
            if ($mysqli->connect_errno) {
                throw new Exception("При добавлении
пользователя произошла внутренняя ошибка", 2);
            }
            //Проверяем, есть ли пользователи в базе с данным логином
            $checkUsersQuery=$mysqli->prepare("SELECT COUNT(*)
FROM users WHERE login=?");
            $checkUsersQuery->bind_param('s', $login);
            $checkUsersQuery->execute();
            $checkUsersQuery->bind_result($countUsers);
            $checkUsersQuery->fetch();
            if($countUsers!=0){
                throw new Exception("Пользователь с данным
логином уже зарегистрирован", 3);
            }
            $checkUsersQuery->close();
        }
        $checkUsersQuery->close();
        //Добавляем нового пользователя в базу
        $addUserQuery=$mysqli->prepare("INSERT INTO users(login,
password) VALUES (?, ?)");
        $encryptedPassword=md5($password);
        $addUserQuery->bind_param('ss', $login, $encryptedPassword);
        if(!$addUserQuery->execute()){
            throw new Exception("При добавлении пользователя
произошла внутренняя ошибка", 2);
        }
        $addUserQuery->close();
        //Возвращаем данные зарегистрированного пользователя
        $getUserDataQuery=$mysqli->prepare("SELECT id, login,
password from users where login=?");
        $getUserDataQuery->bind_param('s', $login);
        if(!$getUserDataQuery->execute()){
            throw new Exception("При добавлении пользователя
произошла внутренняя ошибка", 2);
        }
        $getUserDataQuery-
        >bind_result($rezId,$rezLogin,$rezPassword);
        $getUserDataQuery->fetch();
        $returnId=$rezId;
        $returnLogin=$rezLogin;
        $returnPassword=$rezPassword;
        $getUserDataQuery->close();
        //Добавляем для пользователя директорию на сервере
        $postfix = date('_Ymd_is');
        $userDirectoryName=$returnId.$postfix;//имя директории
        $userDirectoryPath=USER_DIRECTORY DIRECTORY_SEPARATOR.
        $userDirectoryName;//полный путь к директории
        if(!mkdir($userDirectoryPath, 0770, true)){
            throw new Exception("При добавлении
пользователя произошла внутренняя ошибка", 7);
        }
        //Добавляем имя директории в базу
        $addUserDirectory=$mysqli->prepare("UPDATE users SET
directory=? WHERE id=?");
        $addUserDirectory-
        >bind_param('si', $userDirectoryName, $returnId);
        if($addUserDirectory->execute()){
            throw new Exception("При добавлении пользователя
произошла внутренняя ошибка", 8);
        }
        $addUserDirectory->close();
        $mysqli->close();

        //Возвращаем данные зарегистрированного пользователя
        $user = array();
        $user ["id"] = $returnId;
        $user ["login"] = $returnLogin;
        $user ["password"] = $returnPassword;
        $response = array();
        $response["success"]=1;
        $response["message"]="Пользователь добавлен успешно";
        $response["user"]=$user;
        echo
        json_encode($response,JSON_UNESCAPED_UNICODE);

        }catch(Exception $ex){
        $response = array();
        $response["success"]=0;
        $response["error_code"]=$ex->getCode();
        $response["message"]=$ex->getMessage();
        echo json_encode($response,JSON_UNESCAPED_UNICODE);
        }
    }

    //авторизация пользователя
    public function authorizationUser()
    {
        try{
            $check_ipput_field=isset($_POST['key']) &&
            isset($_POST['login']) && isset($_POST['password']);

            if (!$check_ipput_field) {
                throw new Exception("В запросе не заданы обязательные поля",
0);
            }
            $key=$_POST['key'];
            $login=mb_convert_encoding($_POST['login'], "UTF-8");
            $password=mb_convert_encoding($_POST['password'], "UTF-8");
            if(strcmp($key,ANDROID_KEY)!=0){
                throw new Exception("Неверное значение ANDROID KEY.", 1);
            }

            $mysqli=new
            mysqli(DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE);
            if ($mysqli->connect_errno) {
                throw new Exception("При добавлении
пользователя произошла внутренняя ошибка", 2);
            }
        }
    }
}

```

Рис.П2.60. Текст модуля users.api.php

## П2.60. Продолжение

```

$getUserDataQuery=$mysqli->prepare("SELECT id, login,
password, name, image FROM users WHERE login=? AND
password=?");

if (!$getUserDataQuery) {
    throw new Exception("При авторизации
пользователя произошла внутренняя ошибка", 2);
}

$encryptedPassword=md5($password);
$getUserDataQuery->bind_param('ss',$login,
$encryptedPassword);
if (!$getUserDataQuery->execute()){
    throw new Exception("При авторизации пользователя
произошла внутренняя ошибка", 2);
}

$getUserDataQuery-
>bind_result($rezId,$rezLogin,$rezPassword,$rezName,$rezImage
);
if (!$getUserDataQuery->fetch()){
    throw new Exception("Не удалось пройти
авторизацию. Указанны неверные значения логина и пароля",
4);
}

$user = array();
$user ["id"] = $rezId;
$user ["login"] = $rezLogin;
$user ["password"] = $rezPassword;
$user ["name"] = $rezName;
$user ["image"] = lis_null($rezImage);
$response = array();
$response["success"]=1;
$response["message"]="Пользователь авторизован успешно";
$response["user"]=$user;
echo
json_encode($response,JSON_UNESCAPED_UNICODE);

$getUserDataQuery->close();
$mysqli->close();
}catch(Exception $ex){
$response = array();
$response["success"]=0;
$response["error_code"]=$ex->getCode();
$response["message"]=$ex->getMessage();
echo json_encode($response,JSON_UNESCAPED_UNICODE);
}

//изменение имени пользователя
public function updateUserName()
{
try{
$check_ippot_field=isset($_POST['key']) && isset($_POST['id'])
&& isset($_POST['name']);

if (!$check_ippot_field){
throw new Exception("В запросе не заданы обязательные поля",
0);
}
$key=$_POST['key'];
$id=mb_convert_encoding($_POST['id'], "UTF-8");
$name=mb_convert_encoding($_POST['name'], "UTF-8");
if(strcmp($key,ANDROID_KEY)!=0){
throw new Exception("Неверное значение ANDROID KEY.", 1);
}
if(strlen($name)==0){
throw new Exception("Не задано имя пользователя", 14);
}
$mysqli=new
mysqli(DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE);

if ($mysqli->connect_errno) {
    throw new Exception("При изменении имени
пользователя произошла внутренняя ошибка", 2);
}
}

$updateUserNameQuery=$mysqli->prepare('UPDATE users SET
name=?, update_date=NOW() WHERE id=?');
$updateUserNameQuery->bind_param('si',$name, $id);
if(!$updateUserNameQuery->execute()){
    throw new Exception("При изменении имени
пользователя произошла внутренняя ошибка", 3);
}
if($updateUserNameQuery->affected_rows==0){

    throw new Exception("Пользователь с данным
идентификатором не найден", 5);
}

$updateUserNameQuery->close();

$getUserDataQuery=$mysqli->prepare('SELECT id, login,
password, name, image FROM users WHERE id=?');
$getUserDataQuery->bind_param('i',$id);
if (!$getUserDataQuery->execute()){
    throw new Exception("При изменении имени пользователя
произошла внутренняя ошибка....", 3);
}

$getUserDataQuery-
>bind_result($rezId,$rezLogin,$rezPassword,$rezName,$rezImage
);
$getUserDataQuery->fetch();
$user = array();
$user ["id"] = $rezId;
$user ["login"] = $rezLogin;
$user ["password"] = $rezPassword;
$user ["name"] = $rezName;
$user ["image"] = lis_null($rezImage);
$response = array();
$response["success"]=1;
$response["message"]="Имя пользователя изменено успешно";
$response["user"]=$user;
echo
json_encode($response,JSON_UNESCAPED_UNICODE);

$getUserDataQuery->close();
$mysqli->close();
}catch(Exception $ex){
$response = array();
$response["success"]=0;
$response["error_code"]=$ex->getCode();
$response["message"]=$ex->getMessage();
echo
json_encode($response,JSON_UNESCAPED_UNICODE);

}

}

//Загрузка пользовательского изображения
public function loadUserImage()
{
try{
//Проверяем, заданы ли обязательные поля в запросе
$check_ippot_field=isset($_POST['key']) && isset($_POST['id']);

if (!$check_ippot_field) {
    throw new Exception("В запросе не заданы обязательные
поля", 0);
}

$key=$_POST['key'];
$id=mb_convert_encoding($_POST['id'], "UTF-8");
if(strcmp($key,ANDROID_KEY)!=0){
    throw new Exception("Неверное значение
ANDROID KEY.", 1);
}
}

//Проверяем задан ли файл
if(isset($_FILES['imgfile'])){//||
empty($_FILES['imgfile']['name'])){

    throw new Exception("Ошибка при загрузке
файла. Не задан файл", 8);
}
}

```

## П2.60. Продолжение

```

    }
    //Подключаемся к базе
    $mysqli=new
    mysqli(DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE);
    if ($mysqli->connect_errno) {
        throw new Exception("При загрузке файла
        произошла ошибка подключения к базе", 2);
    }
    //Возвращаем файл и директорию пользователя
    $getUserFileAndDirectory=$mysqli->prepare('SELECT
    image,directory from users WHERE id=?');
    $getUserFileAndDirectory->bind_param('i',$id);
    if (!$getUserFileAndDirectory->execute()){
        throw new Exception("При добавлении файла
        произошла внутренняя ошибка", 9);
    }
    $getUserFileAndDirectory->bind_result($image,$directoty);

    if(!$getUserFileAndDirectory->fetch()){
        throw new Exception("При добавлении файла
        произошла внутренняя ошибка", 10);
    }
    $rezDirectory=$directoty;
    $rezImage=$image;
    $getUserFileAndDirectory->close();
    //Загружаем файл в директорию
    $filesAPI=new FilesAPI();
    $upload_dir =
    USER_DIRECTORY.DIRECTORY_SEPARATOR.$rezDirectory
    .DIRECTORY_SEPARATOR; // Место, куда будут загружаться
    файлы.
    $result = $filesAPI->upload_file($_FILES['imgfile'],$upload_dir);
    $img = 'null'; // В таблице поле должно иметь значение по
    умолчанию null
    //возникла ошибка при загрузке файла
    if(isset($result['error'])){
        $error = $result['error'];
        throw new Exception($error, 6);
    }
    // $img = "'.$result['filename'].'";
    $img = $result['filename'];
    //Добавляем имя файла в базу
    $addImageQuery = $mysqli->prepare('UPDATE users SET
    image=?, update_date=NOW() where id=?');
    $addImageQuery->bind_param('si',$img,$id);
    if(!$addImageQuery->execute()){
        throw new Exception("При добавлении файла произошла
        внутренняя ошибка", 6);
    }
    if($addImageQuery->affected_rows==0){
        throw new Exception("Пользователь с данным
        идентификатором не найден", 5);
        $addImageQuery->close();
    }
    $addImageQuery->close();

    //Удаляем старый файл изображения
    if(is_null($rezImage)){
        $userOldImage=USER_DIRECTORY.DIRECTORY_SEPARATOR
        .$rezDirectory.DIRECTORY_SEPARATOR.$rezImage;
        unlink($userOldImage);
    }
    //Возвращаем результат добавления файла
    $user = array();
    $user ["id"] = $id;
    $user ["image"] = !is_null($img);
    $response = array();
    $response["success"]=1;
    $response["message"]="Изображение пользователя загружено
    успешно";
    $response["user"]=$user;
    echo
    json_encode($response,JSON_UNESCAPED_UNICODE);
}

$mysqli->close();
}catch(Exception $ex){
$response = array();
$response["success"]=0;
$response["error_code"]=$ex->getCode();
$response["message"]=$ex->getMessage();
echo
json_encode($response,JSON_UNESCAPED_UNICODE);
}

//Удаляем изображение пользователя
public function deleteUserImage()
{
try{
//Проверяем, заданы ли обязательные поля в запросе
$check_ipput_field=isset($_POST['key']) && isset($_POST['id']);
if (!$check_ipput_field) {
    throw new Exception("В запросе не заданы обязательные
    поля", 0);
}
$key=$_POST['key'];
$id=mb_convert_encoding($_POST['id'], "UTF-8");
if(strcmp($key,ANDROID_KEY)!=0){
    throw new Exception("Неверное значение
    ANDROID KEY.", 1);
}
//Подключаемся к базе
$mysqli=new
mysqli(DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE);
if ($mysqli->connect_errno) {
    throw new Exception("Произошла ошибка
    подключения к базе", 2);
}
//Возвращаем файл и директорию пользователя
$getUserFileAndDirectory=$mysqli->prepare('SELECT
    image,directory from users WHERE id=?');
$getUserFileAndDirectory->bind_param('i',$id);
if (!$getUserFileAndDirectory->execute()){
    throw new Exception("При удалении файла
    произошла внутренняя ошибка", 11);
}
$getUserFileAndDirectory->bind_result($image,$directoty);

if(!$getUserFileAndDirectory->fetch()){
    throw new Exception("Пользователь с указанным
    идентификатором не существует", 12);
}

$rezDirectory=$directoty;
$rezImage=$image;
$getUserFileAndDirectory->close();
if(is_null($rezImage)){
    throw new Exception("Указанный пользователь не имеет
    файла", 13);
}
//Добавляем NULL в базу для изображения
$deleteImageQuery = $mysqli->prepare('UPDATE users SET
    image=NULL, update_date=NOW() where id=?');
$deleteImageQuery->bind_param('i',$id);
if(!$deleteImageQuery->execute()){
    throw new Exception("При удалении файла произошла
    внутренняя ошибка", 11);
}
$deleteImageQuery->close();
//Удаляем старый файл изображения и директории
if(is_null($rezImage)){
    $userOldImage=USER_DIRECTORY.DIRECTORY_SEPARATOR
    .$rezDirectory.DIRECTORY_SEPARATOR.$rezImage;
    unlink($userOldImage);
}

}
}

```

## П2.60. Продолжение

```

$user = array();
$user ["id"] = $id;
$user ["image"] = false;
$response = array();
$response["success"]=1;
$response["message"]="Изображение пользователя удалено успешно";
$response["user"]=$user;
echo
json_encode($response,JSON_UNESCAPED_UNICODE);

$mysqli->close();
}catch(Exception $ex){
$response = array();
$response["success"]=0;
$response["error_code"]=$ex->getCode();
$response["message"]=$ex->getMessage();
echo json_encode($response,JSON_UNESCAPED_UNICODE);
}
}

//Изменение пароля пользователя

public function changeUserPassword(){
try{
$check_ippot_field=isset($_POST['key']) && isset($_POST['id'])
&& isset($_POST['password']);

if (!$check_ippot_field) {
throw new Exception("В запросе не заданы обязательные поля", 0);
}
$key=$_POST['key'];
$Id=mb_convert_encoding($_POST['id'], "UTF-8");
$password=mb_convert_encoding($_POST['password'], "UTF-8");
if(strcmp($key,ANDROID_KEY)!=0){
throw new Exception("Неверное значение ANDROID KEY.", 1);
}
if(strlen($password)<4){
throw new Exception("Пароль должен содержать не менее 4 символов", 14);
}
$mysqli=new
mysqli(DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE);
if ($mysqli->connect_errno) {
    throw new Exception("При изменении пароля пользователя произошла внутренняя ошибка подключения", 2);
}

SET password=?, update_date=NOW() WHERE id=?);
$encryptedPassword=md5($password);
$updateUserPasswordQuery-
>bind_param('si',$encryptedPassword, $id);
if(!$updateUserPasswordQuery->execute()){
    throw new Exception("При изменении пароля пользователя произошла внутренняя ошибка", 3);
}
if($updateUserPasswordQuery->affected_rows==0){
    throw new Exception("Пользователь с данным идентификатором не найден", 5);
}
$updateUserPasswordQuery->close();
}

$updateUserPasswordQuery->close();
getUserDataQuery=$mysqli->prepare('SELECT id, password
FROM users WHERE id=?');
getUserDataQuery->bind_param('i',$id);
if(!$getUserDataQuery->execute()){
    throw new Exception("При изменении пароля пользователя произошла внутренняя ошибка", 3);
}
getUserDataQuery->bind_result($rezId,$rezPassword);
getUserDataQuery->fetch();
$user = array();
$user ["id"] = $rezId;
$user ["password"] = $rezPassword;
$response = array();
$response["success"]=1;
$response["message"]="Пароль пользователя изменен успешно";
$response["user"]=$user;
echo
json_encode($response,JSON_UNESCAPED_UNICODE);

getUserDataQuery->close();
$mysqli->close();
}catch(Exception $ex){
$response = array();
$response["success"]=0;
$response["error_code"]=$ex->getCode();
$response["message"]=$ex->getMessage();
echo
json_encode($response,JSON_UNESCAPED_UNICODE);

}
}
}
?>
```

## Модуль api.requests.php

Рис.П2.61. Текст модуля requests.api.php

## П2.61. Продолжение

```

}

//Если задан файл заявки,то загружаем его
if(isset($_FILES['imgfile'])){
//Получаем директорию пользователя
$getUserDirectory=$mysqli->prepare('SELECT directory from
users WHERE id=?');
$getUserDirectory->bind_param('i',$id);
if(!$getUserDirectory->execute()){
throw new Exception("При добавлении заявки произошла
внутренняя ошибка", 17);
}
$getUserDirectory->bind_result($directoty);
if(!$getUserDirectory->fetch()){
throw new Exception("При добавлении заявки произошла
внутренняя ошибка", 17);
}
$rezDirectory=$directoty;
$getUserDirectory->close();
//Загружаем файл в директорию
$fileAPI=new FileAPI();
$upload_dir =
USER_DIRECTORY DIRECTORY_SEPARATOR.$rezDirectory
.DIRECTORY_SEPARATOR; // Место, куда будут загружаться
файлы.
$result = $fileAPI->upload_file($_FILES['imgfile'],$upload_dir);
//возникла ошибка при загрузке файла
if(isset($result['error'])){
$error = $result['error'];
$responseMessage="Возникла ошибка при загрузке файла.
Заявка добавлена без графического файла.";
} else{//Если удалось загрузить файл
$img = $result['filename'];
}
}
//Добавляем данные заявки в базу
$query=$mysqli->prepare("INSERT INTO requests(message,
latitude, longitude, id_user,image) VALUES (?, ?, ?, ?, ?)");
if (!$query) {
throw new Exception("При добавлении заявки произошла
внутренняя ошибка", 2);
}
$query-
>bind_param('sssi',$message,strval($latitude),strval($longitude),
$id, $img);

$userLongitude=$_GET['longitude'];
}
if(strcmp($key,ANDROID_KEY)!=0){
throw new Exception("Неверное значение ANDROID KEY.", 1);
}
$mysqli=new
mysqli(DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE);
if ($mysqli->connect_errno) {
throw new Exception("При получении списка заявок произошла
внутренняя ошибка подключения", 2);
}
//Добавляем данные заявки в базу
$getRequestsQuery=$mysqli->prepare('SELECT requests.id,
requests.message, requests.latitude, requests.longitude,
CONVERT_TZ(requests.create_date, "+00:00", "+03:00"),
requests.image,requests.id_status, requests.id_user, users.name,
users.image
FROM requests,users WHERE TIMESTAMPDIFF(MINUTE,
requests.create_date,NOW())<=1440 AND
requests.id_user=users.id');
if(!$getRequestsQuery){
echo $mysqli->error;
}
if(!$getRequestsQuery->execute()){
throw new Exception("При получении списка заявок произошла
внутренняя ошибка", 2);
}

if(!($query->execute())){
throw new Exception("При добавлении заявки произошла
внутренняя ошибка", 17);
}
$query->close();
}

//Получает заявки из базы
public function getRequests(){
try{
$check_ippot_field=isset($_GET['key']);
if (!$check_ippot_field) {
throw new Exception("В запросе не заданы обязательные поля",
0);
}
$key=$_GET['key'];
$isGetAllRequest=1;
//Проверяем передан ли радиус в функцию
if(isset($_GET['radius'])){
$checkSetUserLocation=isset($_GET['latitude'])&&isset($_GET['longitude']);
//Проверяем переданы ли значения широты и долготы
пользователя
if(!$checkSetUserLocation){
throw new Exception("В запросе не заданы обязательные
поля:при указании радиуса, требуется указать широту и
долготу", 0);
}
$isGetAllRequest=0;
$radius=$_GET['radius'];
$userLatitude=$_GET['latitude'];
}

$getRequestsQuery-
>bind_result($rezRequestId,$rezMessage,$rezLatitude,$rezLongitude,$rezCreateDate,
$rezRequestsImage,$rezIdStatus,$rezidUser,$rezName,
$rezUserImage);
$response = array();
$response["success"]=1;
$response["message"]="Заявки получены успешно";
$response["requests"]=array();
while ($getRequestsQuery->fetch()){
if(!$isGetAllRequest){
$distanse=$this-
>calculateTheDistance($userLatitude,$userLongitude,$rezLatitude
,$rezLongitude);
//echo $rezRequestId." :расстояние ".$distanse;
if($distanse>$radius){
continue;
}
}
$request=array();
$request["id"]=$rezRequestId;
$request["message"]=$rezMessage;
$request["latitude"]=$rezLatitude;
$request["longitude"]=$rezLongitude;
$request["create_date"]=$rezCreateDate;
$request["image"]=$rezRequestsImage;
$request["id_status"]=$rezIdStatus;
$request["user"]=array();
}
}

```

## П2.61. Продолжение

```

$user=array();
$user["id"]=$rezIdUser;
$user["name"]=$rezName;
$user["image"]=$rezUserImage;
array_push($request["user"],$user);
array_push($response["requests"],$request);

}

$getRequestsQuery->close();
$mysqli->close();
echo json_encode($response,JSON_UNESCAPED_UNICODE);
}catch(Exception $ex){
$response = array();
$response["success"]=0;
$response["error_code"]=$ex->getCode();
$response["message"]=$ex->getMessage();
echo json_encode($response,JSON_UNESCAPED_UNICODE);
}
}

public function updateRequestStatus()
{
try{
$ccheck_ippot_field=isset($_POST['key']) && isset($_POST['id'])
&& isset($_POST['id_status']);

if (!$check_ippot_field) {
throw new Exception("В запросе не заданы обязательные поля",
0);
}
$key=$_POST['key'];
$idRequest=mb_convert_encoding($_POST['id'], "UTF-8");
$idStatus=mb_convert_encoding($_POST['id_status'], "UTF-8");
if(strcmp($key,ANDROID_KEY)!=0){
throw new Exception("Неверное значение ANDROID KEY.", 1);
}
}

$mysqli=new
mysqli(DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE);
if ($mysqli->connect_errno) {
throw new Exception("При изменении статуса произошла
ошибка", 2);
} $updateRequestStatusQuery=$mysqli->prepare('UPDATE
requests SET id_status=? WHERE id=?');
$updateRequestStatusQuery->bind_param('ii',$idStatus,
$idRequest);
if(!$updateRequestStatusQuery->execute()){
throw new Exception("При изменении статуса произошла
ошибка", 3);
}
if($mysqli->affected_rows==0){
throw new Exception("Не удалось изменить статус указаны
неверные значения id_status, id_request", 4);
}
$updateRequestStatusQuery->close();
$mysqli->close();
$response = array();
$response["success"]=1;
$response["message"]="Статус заявки изменен успешно";
echo json_encode($response,JSON_UNESCAPED_UNICODE);
}catch(Exception $ex){
$response = array();
$response["success"]=0;
$response["error_code"]=$ex->getCode();
$response["message"]=$ex->getMessage();
echo json_encode($response,JSON_UNESCAPED_UNICODE);
}
}
}
?>

```

## Модуль api.offers.php

```

<?php

require_once 'config.php';
class OffersAPI{
//Добавляет новое предложение помощи в базу
public function addOffer(){
try{
$ccheck_ippot_field=isset($_POST['key']) &&
isset($_POST['message']) && isset($_POST['id_user']) &&
isset($_POST['id_request']);

if (!$check_ippot_field) {
throw new Exception("В запросе не заданы
обязательные поля", 0);
}
$key=$_POST['key'];
$message=mb_convert_encoding($_POST['message'], "
UTF-8");
$idUser=mb_convert_encoding($_POST['id_user'], "UT
F-8");
$idRequest=mb_convert_encoding($_POST['id_request'],
]", "UTF-8");
if(strcmp($key,ANDROID_KEY)!=0){
throw new Exception("Неверное значение
ANDROID KEY.", 1);
}
}

$mysqli=new
mysqli(DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE);
if ($mysqli->connect_errno) {
throw new Exception("При добавлении предложения помощи произошла
ошибка при подключении", 2);
}
//Добавляем данные предложения
помощи в базу
$query=$mysqli->prepare('INSERT INTO
offers(message, id_request,id_user) VALUES (?, ?, ?)');
$query-
>bind_param('sii',$message,$idRequest,$idUser);

if(!$query->execute()){
throw new Exception("При добавлении
предложения помощи произошла внутренняя ошибка", 3);
}

$response = array();
$response["success"]=1;
$response["message"]="Предложение помощи добавлено
успешно";
echo
json_encode($response,JSON_UNESCAPED_UNICODE);
$query->close();
$mysqli->close();
}
}

```

Рис.П2.62. Текст модуля offer.api.php

## П2.62. Продолжение

```

}catch(Exception $ex){
    $response = array();
        $response["success"]=0;
        $response["error_code"]=$ex->getCode();
    $response["message"]=$ex->getMessage();
        echo
json_encode($response,JSON_UNESCAPED_UNICODE);
}

//Получает список предложений к заявке
public function getOffers(){
try{
$check_ipput_field=isset($_POST['key']) &&
isset($_POST['id_request']);
if (!$check_ipput_field) {
    throw new Exception("В запросе не заданы
обязательные поля", 0);
}
$key=$_POST['key'];
$requestId=mb_convert_encoding($_POST['id_request'], "UTF-
8");
$offerId=null;
if(isset($_POST['id_offer'])){
    $offerId=mb_convert_encoding($_POST['id_offer'], "U
TF-8");
    if(!((int)$offerId>=-1)){
        throw new Exception("Значение id_offer
должно быть целым положительным числом", 1);
    }
}
if(strcmp($key,ANDROID_KEY)!=0){
    throw new Exception("Неверное значение ANDROID
KEY.", 1);
}
$mysqli=new
mysqli(DB_SERVER,DB_USER,DB_PASSWORD,DB_DATAB
ASE);
if ($mysqli->connect_errno) {
    throw new Exception("При получении предложений
помощи произошла внутренняя ошибка подключения", 2);
}
//Получаем данные предложений помощи заявок
$getOffersQuery=$mysqli->prepare('SELECT id,
message, CONVERT_TZ(create_date, "+00:00", "+03:00"),
CONVERT_TZ(update_date, "+00:00", "+03:00"),
id_user,id_request FROM offers WHERE id_request=? AND (? IS
NULL OR ? IS NOT NULL AND id>?)');
if(!$getOffersQuery){
    echo $mysqli->error;
}
$getOffersQuery-
>bind_param('iiii',$requestId,$offerId,$offerId,$offerId);
if(!$getOffersQuery->execute()){
    throw new Exception("При получении списка предложений
помощи возникла ошибка", 3);
}
$getOffersQuery-
>bind_result($rezOfferId,$rezMessage,$rezCreateDate,$rezUpdate
Date,$rezIdUser,
$rezIdRequest);
}

//Получает список предложений к заявке
public function getOffers(){
try{
$check_ipput_field=isset($_POST['key']) &&
isset($_POST['id_request']);
if (!$check_ipput_field) {
    throw new Exception("В запросе не заданы
обязательные поля", 0);
}
$key=$_POST['key'];
$requestId=mb_convert_encoding($_POST['id_request'], "UTF-
8");
$offerId=null;
if(isset($_POST['id_offer'])){
    $offerId=mb_convert_encoding($_POST['id_offer'], "U
TF-8");
    if(!((int)$offerId>=-1)){
        throw new Exception("Значение id_offer
должно быть целым положительным числом", 1);
    }
}
if(strcmp($key,ANDROID_KEY)!=0){
    throw new Exception("Неверное значение ANDROID
KEY.", 1);
}
$mysqli=new
mysqli(DB_SERVER,DB_USER,DB_PASSWORD,DB_DATAB
ASE);
if ($mysqli->connect_errno) {
    throw new Exception("При получении предложений
помощи произошла внутренняя ошибка подключения", 2);
}
//Получаем данные предложений помощи заявок
$getOffersQuery=$mysqli->prepare('SELECT id,
message, CONVERT_TZ(create_date, "+00:00", "+03:00"),
CONVERT_TZ(update_date, "+00:00", "+03:00"),
id_user,id_request FROM offers WHERE id_request=? AND (? IS
NULL OR ? IS NOT NULL AND id>?)');
if(!$getOffersQuery){
    echo $mysqli->error;
}
$getOffersQuery-
>bind_param('iiii',$requestId,$offerId,$offerId,$offerId,$offerId);
if(!$getOffersQuery->execute()){
    throw new Exception("При получении списка предложений
помощи возникла ошибка", 3);
}
$getOffersQuery-
>bind_result($rezOfferId,$rezMessage,$rezCreateDate,$rezUpdate
Date,$rezIdUser,
$rezIdRequest);
}

//Получаем данные пользователей предложений
$getUsersQuery=$mysqli->prepare('SELECT
DISTINCT users.id, users.name, users.image,
CONVERT_TZ(users.create_date, "+00:00", "+03:00"),
CONVERT_TZ(users.update_date, "+00:00", "+03:00") FROM
users,offers WHERE users.id=offers.id_user AND
offers.id_request=? AND (? IS NULL OR ? IS NOT NULL AND
offers.id>?)');
if(!$getUsersQuery){
    echo $mysqli->error;
}
$getUsersQuery-
>bind_param('iiii',$requestId,$offerId,$offerId,$offerId);
if(!$getUsersQuery->execute()){
    throw new Exception("При получении списка предложений
произошла внутренняя ошибка", 4);
}
$getUsersQuery-
>bind_result($rezUserId,$rezUserName,$rezUserImage,$rezUserC
reateDate,$rezUserUpdateDate);
while ($getUsersQuery->fetch()){
    $user=array();
        $user["id"]=$rezUserId;
        $user["name"]=$rezUserName;
        $user["image"]=is_null($rezUserImage)?0:1;
        $user["create_date"]=$rezUserCreateDate;
        $user["update_date"]=$rezUserUpdateDate;
    array_push($response["users"], $user);
}
$getUsersQuery->close();
$mysqli->close();
echo json_encode($response,JSON_UNESCAPED_UNICODE);
}catch(Exception $ex){
    $response = array();
        $response["success"]=0;
        $response["error_code"]=$ex->getCode();
    $response["message"]=$ex->getMessage();
        echo
json_encode($response,JSON_UNESCAPED_UNICODE);
}
}

```

## Модуль api.files.php

```
<?php
class FilesApi{
//Загрузить файл
public function upload_file($file, $upload_dir, $allowed_types=
array('image/png','image/x-png','image/jpeg','image/gif')){
$blacklist = array(".php",".phtml",".php3",".php4");
$filename = $file['name']; // В переменную $filename заносим
точное имя файла.
$ext = substr($filename, strpos($filename,'.'), strlen($filename)-
1); // В переменную $ext заносим расширение загруженного
файла.
if(in_array($ext,$blacklist )) {
    return array('error' => 'Запрещено загружать исполняемые
файлы');
    $max_filesize = 8388608; // Максимальный размер
загружаемого файла в байтах (в данном случае он равен 8 Мб).
$prefix = date('Ymd-is_');
if(!is_writable($upload_dir)) // Проверяем, доступна ли на
запись папка, определенная нами под загрузку файлов.
```

```
return array('error' => 'Невозможно загрузить файл в папку
"'.$upload_dir.'". Установите права доступа - 777.');
if(!in_array($file['type'], $allowed_types))
    return array('error' => 'Данный тип файла не
поддерживается.');
if(filesize($file['tmp_name']) > $max_filesize)
    return array('error' => 'файл слишком большой.
максимальный размер '.intval($max_filesize/(1024*1024)).'мб');
if(!move_uploaded_file($file['tmp_name'],$upload_dir.$prefix.$fil
ename)) // Загружаем файл в указанную папку.
    return array('error' => 'При загрузке возникли ошибки.
Попробуйте ещё раз.');
    return Array('filename' => $prefix.$filename);
}
?
?>
```

Рис. П2.63. Текст модуля api.files.php

## Пакет ServerApi

### Модуль offer.addOffer.php

```
<?php
require_once 'api.config.php';
require_once API_OFFERS;
$offerAPI=new OffersAPI();
$offerAPI->addOffer();
?>
```

Рис. П2.64. Текст модуля offer.addOffer.php

### Модуль offers.getOffers.php

```
<?php
require_once 'api.config.php';
require_once API_OFFERS;
$offerAPI=new OffersAPI();
$offerAPI->getOffers();
?>
```

Рис. П2.65. Текст модуля offers.getOffers.php

### Модуль requests.addRequest.php

```
<?php
require_once 'api.config.php';
require_once API_REQUESTS;
$requestAPI=new RequestsAPI();
$requestAPI->addRequest();
?>
```

Рис. П2.66. Текст модуля requests.addRequest.php

## Модуль requests.getActualRequests.php

```
<?php
require_once 'api.config.php';
require_once API_REQUESTS;
$requestAPI=new RequestsAPI();
$requestAPI->getActualRequests();
?>
```

Рис. П2.67. Текст модуля requests.getActualRequests.php

## Модуль requests.getImage.php

```
<?php
require_once 'api.config.php';
require_once PHP_DIRECTORY.'/config.php';
try{
//Проверяем, заданы ли обязательные поля в запросе
$check_post_ippot_field=isset($_POST['key']) &&
isset($_POST['id']);
$check_get_ippot_field=isset($_GET['key']) &&
isset($_GET['id']);

if (!$check_post_ippot_field && !$check_get_ippot_field) ||
($check_post_ippot_field && $check_get_ippot_field)) {
    throw new Exception("В запросе не заданы обязательные
поля", 0);
}
if($check_post_ippot_field){
$key=$_POST['key'];
$id=mb_convert_encoding($_POST['id'], "UTF-8");
}
if($check_get_ippot_field){
$key=$_GET['key'];
$id=mb_convert_encoding($_GET['id'], "UTF-8");
}
if(strcmp($key,ANDROID_KEY)!=0){
    throw new Exception("Неверное значение ANDROID
KEY.", 1);
}
$mysqli=new
mysqli(DB_SERVER,DB_USER,DB_PASSWORD,DB_DATA
ASE);
if ($mysqli->connect_errno) {
    throw new Exception("Ошибка подключения к базе",
2);
}
//Возвращаем файл заявки и директорию пользователя
 getRequestFileAndDirectory=$mysqli->prepare('SELECT
requests.image,users.directory from users,requests
WHERE users.id=requests.id_user AND requests.id=?'
AND requests.image IS NOT NULL');
 getRequestFileAndDirectory->bind_param('i,$id);
if(!$getRequestFileAndDirectory->execute()){
    throw new Exception("При получении файла
произошла внутренняя ошибка", 9);
}
 getRequestFileAndDirectory->bind_result($image,$directoty);
if(!$getRequestFileAndDirectory->fetch()){
    throw new Exception("Заявка не имеет изображения",
19);
}
$rezDirectory=$directoty;
$rezImage=$image;
 getRequestFileAndDirectory->close();
$mysqli->close();
$file =
USER_DIRECTORY.DIRECTORY_SEPARATOR.$rezDirectory
.DIRECTORY_SEPARATOR.$rezImage;
if(is_null($image)){
    throw new Exception("Заявка не имеет изображения", 20);
}
if (!file_exists($file)) {
    throw new Exception("Заявка не имеет изображения", 21);
}
header("Content-type: image/jpeg");
$im = imagecreatefromjpeg ($file);
imagejpeg ($im);
imagedestroy($im);
}catch(Exception $ex){
$response = array();
$response["success"]=0;
$response["error_code"]=$ex->getCode();
$response["message"]=$ex->getMessage();
echo json_encode($response,JSON_UNESCAPED_UNICODE);
}
?>
```

Рис. П2.68. Текст модуля requests.getImage.php

## Модуль requests.updateRequestStatus.php

```
<?php
require_once 'api.config.php';
require_once API_REQUESTS;
$requestAPI=new RequestsAPI();
$requestAPI->updateRequestStatus();
?>
```

Рис. П2.69. Текст модуля requests.updateRequestStatus.php

## Модуль users.authorizeUser.php

```
<?php
require_once 'api.config.php';
require_once API_USERS;
$userAPI=new UserAPI();
$userAPI->authorizationUser();
?>
```

Рис. П2.70. Текст модуля users.authorizeUser.php

## Модуль users.changeImage.php

```
<?php
require_once 'api.config.php';
require_once API_USERS;
$userAPI=new UserAPI();
$userAPI->loadUserImage();
?>
```

Рис. П2.71. Текст модуля users.changeImage.php

## Модуль users.changeName.php

```
<?php
require_once 'api.config.php';
require_once API_USERS;
$userAPI=new UserAPI();
$userAPI->updateUserName();
?>
```

Рис. П2.72. Текст модуля users.changeName.php

## Модуль users.changePassword.php

```
<?php
require_once 'api.config.php';
require_once API_USERS;
$userAPI=new UserAPI();
$userAPI->changeUserPassword();
?>
```

Рис. П2.73. Текст модуля users.changeName.php

## Модуль users.deleteImage.php

```
<?php
require_once 'api.config.php';
require_once API_USERS;
$userAPI=new UserAPI();
$userAPI->deleteUserImage();
?>
```

Рис. П2.74. Текст модуля users.deleteImage.php

## Модуль users.registerUser.php

```
<?php
require_once 'api.config.php';
require_once API_USERS;
$userAPI=new UserAPI();
$userAPI->deleteUserImage();
?>
```

Рис. П2.75. Текст модуля users.registerUser.php

## Модуль users.getImage.php

```

<?php
require_once 'api.config.php';
require_once PHP_DIRECTORY.'/config.php';
try{
//Проверяем, заданы ли обязательные поля в запросе
$check_post_ippot_field=isset($_POST['key']) &&
isset($_POST['id']);
$check_get_ippot_field=isset($_GET['key']) &&
isset($_GET['id']);
if (!$check_post_ippot_field && !$check_get_ippot_field) ||
($check_post_ippot_field && $check_get_ippot_field)) {
throw new Exception("В запросе не заданы обязательные поля",
0);
}
if($check_post_ippot_field){
$key=$_POST['key'];
$id=mb_convert_encoding($_POST['id'], "UTF-8");
}
if($check_get_ippot_field){
$key=$_GET['key'];
$id=mb_convert_encoding($_GET['id'], "UTF-8");
}
if(strcmp($key,ANDROID_KEY)!=0){
throw new Exception("Неверное значение ANDROID KEY.", 1);
}
$mysqli=new
mysql(DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE);
if ($mysqli->connect_errno) {
throw new Exception("Ошибка подключения к базе", 2);
}
//Возвращаем файл и директорию пользователя
$getuserfileanddirectory=$mysqli->prepare('SELECT
image,directory from users WHERE id=?');
$getuserfileanddirectory->bind_param('i',$id);
if(!$getuserfileanddirectory->execute()){
throw new Exception("При получении файла произошла
внутренняя ошибка", 9);
}
$image=$getuserfileanddirectory->fetch();
if(!$image){
throw new Exception("Пользователь с указанным
идентификатором не существует", 10);
}
$rezDirectory=$directory;
$rezImage=$image;
$getuserfileanddirectory->close();
$mysqli->close();
$file =
USER_DIRECTORY.DIRECTORY_SEPARATOR.$rezDirectory
.DIRECTORY_SEPARATOR.$rezImage;
if(is_null($image)){
throw new Exception("У пользователя не задано изображение",
11);
}
if (!file_exists($file)) {
throw new Exception("Файл изображения не существует", 12);
}
header("Content-type: image/jpeg");
$im = imagecreatefromjpeg ($file);
imagejpeg ($im);
imagedestroy($im);
}catch(Exception $ex){
$response = array();
$response["success"]=0;
$response["error_code"]=$ex->getCode();
$response["message"]=$ex->getMessage();
echo json_encode($response,JSON_UNESCAPED_UNICODE);
}
?>

```

Рис. П2.76. Текст модуля users.getImage.php

**ПРИЛОЖЕНИЕ 3**  
**Спецификация**

Обозначение	Наименование	Примечание
<b>Документация</b>		
Расчетно-пояснительная записка	Разработка программного обеспечения для мобильных устройств «Помощь на дороге»	
Приложение 2	Текст программы	
Приложение 4	Спецификации на модули	
Приложение 5	Руководство пользователя	
Приложение 6	Наборы тестовых данных и результатов тестирования	
<b>Компоненты</b>		
HelpOnRoad.apk	Пакетный файл мобильного приложения «Помощь на дороге» для установки на мобильное устройство под управлением операционной системы Android	Установка приложения возможна на устройства под управлением операционной системы Android 4.1 (Jelly Bean) и выше

## Спецификации на модули

### 1 Спецификации на компонент «HelpOnRoad.apk»

Указанный компонент является пакетным файлом приложения для мобильных устройств на платформе Android и состоит из модулей (см. п. 5.13, рис. 5.55). Компонент содержит модули, которые реализуют пакеты классов клиентской части приложения, спроектированные в п. 5.2 - 5.9. Каждый модуль состоит из модулей, представляющих собой файлы с расширением .java, реализующие классы соответствующих пакетов.

Таблица П4.1

#### Модули компонента «HelpOnRoad.apk»

Модуль	Описание
1	2
ObjectsOfModel	Предоставляет набор классов для работы с объектами предметной области: заявка на помощь, предложение помощи, пользователь устройства, статус заявки. Является реализацией спроектированного пакета «Объекты модели» (см. п. 5.2)
LocationManager	Предоставляет набор классов для определения местоположения устройства, используя сервисы Google. Является реализацией спроектированного пакета «Объекты модели» (см. п.5.3)
SQLiteDataBase Interface	Предоставляет набор классов, обеспечивающих получение данных о заявках, предложениях помощи, пользователях системы из локальной базы данных устройства. Является реализацией спроектированного пакета «Интерфейс к базе данных SQLite» (см. п.5.4)
JSONParsers	Предоставляет набор классов для заполнения данных объектов предметной области данными, полученными от серверной части приложения в формате JSON. Является реализацией спроектированного пакета «JSON-парсеры» (см. п.5.5)
ManagementObjects OfModel	Предоставляет набор классов для работы с объектами предметной области: заявками, предложениями помощи, пользователями устройства. Методы данных классов формируют HTTP-запросы к серверной части приложения для получения и отправки данных о заявках, предложениях помощи, пользователях системы. Является реализацией спроектированного пакета «Управление объектами модели» (см. п.5.6)
ApplicationController	Модуль, который содержит модули: ManagementAddNewRequest, ManagementListRequest, ManagementUser - для реализации окон пользовательского интерфейса
ManagementAdd NewRequest	Пакетный модуль, который реализует классы для добавления новой заявки на помощь в систему. Является реализацией спроектированного пакета «Управление добавлением заявок» (см. п.5.7)
ManagementList Requests	Пакетный модуль, который реализует классы для просмотра списка заявок, отметок заявок на карте, работу с детальной заявкой. Является реализацией спроектированного пакета «Управление списком заявок» (см. п. 5.9)

## Продолжение табл. П4.1

1	2
ManagementUser	Пакетный модуль, который реализует классы для регистрации, авторизации, изменения учетной записи пользователя приложения. Является реализацией спроектированного пакета «Управление пользователем» (см. п. 5.8)
NetworkOperation	Предоставляет набор классов для выполнения GET и POST HTTP-запросов
CommonInterface Components	Предоставляет набор классов для реализации окон пользовательского интерфейса

## 1.1 Спецификация модуля «ObjectsOfModel»

Модуль состоит из следующих файлов (табл. П4.2).

Таблица П4.2

## Файлы модуля «ObjectsOfModel»

Модуль	Описание
User.java	Описывает объект пользователя приложения. Предоставляет методы для записи и обращения к свойствам пользователя
Driver.java	Описывает объект пользователя, добавившего заявку в систему. Предоставляет методы для записи и обращения к свойствам пользователя
Request.java	Описывает объект заявки на помощь. Предоставляет методы для записи и обращения к полям заявки
Status.java	Предназначен для описания объекта, представляющего статус заявки на помощь. Реализует получение строкового описания статуса заявки по ее идентификатору
Offer.java	Описывает объект предложения помощи. Предоставляет методы для записи и обращения к полям предложения помощи
RequestList.java	Предназначен для работы со списком заявок на помощь, полученных из базы данных SQLite
OfferList.java	Предназначен для работы со списком предложений помощи, полученных из базы данных SQLite

## 1.1.1 Спецификация модуля «User.java»

Модуль содержит атрибуты и функции (табл. П4.3).

Таблица П4.3

## Спецификация модуля «User.java»

Название	Описание
1	2
	Атрибуты
private mPassword	Пароль пользователя

## Продолжение табл. П4.3

1	2
private mLogin	Логин пользователя
	Функции
public void setLogin(String login)	Устанавливает значение логина
public String getLogin()	Возвращает значение логина
public void setPassword(String password)	Устанавливает значение пароля
public String getPassword()	Возвращает значение пароля
public JSONObject getJSONUserData(boolean isId, boolean isLogin, boolean isPassword, boolean isName, boolean isImage)	Представляет информацию о пользователе в JSON-формате для отправки данных пользователя серверной части приложения

## 1.1.2 Спецификация модуля «Driver.java»

Модуль содержит атрибуты и функции (табл. П4.4).

Таблица П4.4

## Спецификация модуля «Driver.java»

Название	Описание
Атрибуты	
protected String mId	Идентификатор пользователя
protected String mName	Имя пользователя
protected String mImage	Путь к изображению пользователя
protected Date mCreateDate	Дата создания записи о пользователе на сервере
protected Date mUpdateDate	Дата обновления записи о пользователе на сервере
protected Date mUploadDate	Дата загрузки записи о пользователе на устройство
Функции	
public void setId(String id)	Устанавливает значение идентификатора пользователя
public String getId()	Возвращает значение идентификатора пользователя
public void setName(String name)	Устанавливает значение имени пользователя
public String getName()	Возвращает значение имени пользователя
public void setCreateDate(Date createDate)	Устанавливает значение даты создания
public Date getCreateDate()	Возвращает значение даты создания
public void setUpdateDate()	Устанавливает значение даты обновления
public void setUpdateDate(Date updateDate)	Возвращает значение даты обновления
public void setUploadDate(Date uploadDate)	Устанавливает значение даты загрузки
public Date getUploadDate()	Возвращает значение даты загрузки

## 1.1.3 Спецификация модуля «Request.java»

Модуль содержит атрибуты и функции (табл. П4.5)

Таблица П4.5

## Спецификация модуля «Request.java»

Название	Описание
Атрибуты	
private int mId	Идентификатор заявки
private String mMessage	Текст сообщения заявки
private boolean isImage	Наличие изображения
private String mImage	Прикрепленное изображение
private Driver mDriver	Сведения о пользователе, добавившем заявку
private LatLng mLocation	Сведения о местоположении заявки
private Status mStatus	Статус заявки
private Date mCreateDate	Дата добавления заявки на сервер
private Date mUpdateDate	Дата обновления заявки на сервере
private Date mUploadDate	Дата загрузки заявки на устройство
private boolean isNew	Является ли заявка новой
private boolean isGetNotify	Было ли получено уведомление для заявки
Функции	
public int getId()	Возвращает идентификатор заявки
public void setId(int id)	Устанавливает идентификатор заявки
public String getMessage()	Возвращает сообщение
public void setMessage(String message)	Устанавливает сообщение
public boolean getIsImage()	Возвращает наличие изображения
public void setIsImage(boolean image)	Устанавливает наличие изображения
public String getImage()	Возвращает изображение
public void setImage(String image)	Устанавливает изображение
public Driver getDriver()	Возвращает данные пользователя
public void setDriver(Driver driver)	Устанавливает данные пользователя
public LatLng getLocation()	Возвращает местоположение
public void setLocation(LatLng location)	Устанавливает местоположение
public Status getStatus()	Возвращает статус
public void setStatus(Status status)	Устанавливает статус
public Date getCreateDate()	Возвращает дату добавления
public void setCreateDate(Date createDate)	Устанавливает дату добавления
public Date getUpdateDate()	Возвращает дату обновления
public void setUpdateDate(Date updateDate)	Устанавливает дату обновления
public Date getUploadDate()	Возвращает дату загрузки
public void setUploadDate(Date uploadDate)	Устанавливает дату загрузки
public boolean isNew()	Возвращает является ли заявка новой
public void setNew(boolean aNew)	Устанавливает является ли заявка новой
public boolean isGetNotify()	Возвращает получено ли уведомление о заявке
public void setGetNotify(boolean getNotify)	Устанавливает получение уведомления о заявке
public JSONObject getRequestData(boolean isId, boolean isMessage, boolean isLongitude, boolean isLatitude, boolean isUserId, boolean isImage, boolean isStatus)	Представляет информацию о заявке в JSON-формате

## 1.1.4 Спецификация модуля «Offer.java»

Модуль содержит атрибуты и функции (табл. П4.6).

Таблица П4.6

## Спецификация модуля «Offer.java»

Название	Описание
Атрибуты	
private int mId	Идентификатор предложения
private String mMessage	Текст сообщения предложения
private boolean isImage	Наличие изображения
private Driver mDriver	Сведения о пользователе, добавившем заявку
private Date mCreateDate	Дата добавления предложения на сервер
private Date mUpdateDate	Дата обновления предложения на сервере
private Date mUploadDate	Дата загрузки предложения на устройство
private boolean isNew	Является ли предложение новым
private boolean isGetNotify	Было ли получено уведомление для предложения
private int mRequestId	Идентификатор заявки
Функции	
public int getId()	Возвращает идентификатор предложения
public void setId(int id)	Устанавливает идентификатор предложения
public String getMessage()	Возвращает сообщение
public void setMessage(String message)	Устанавливает сообщение
public Driver getDriver()	Возвращает данные пользователя
public void setDriver(Driver driver)	Устанавливает данные пользователя
public Date getCreateDate()	Возвращает дату добавления
public void setCreateDate(Date createDate)	Устанавливает дату добавления
public Date getUpdateDate()	Возвращает дату обновления
public void setUpdateDate(Date updateDate)	Устанавливает дату обновления
public Date getUploadDate()	Возвращает дату загрузки
public void setUploadDate(Date uploadDate)	Устанавливает дату загрузки
public boolean isNew()	Возвращает, является ли заявка новой
public void setNew(boolean aNew)	Устанавливает, является ли заявка новой
public boolean isGetNotify()	Возвращает получено ли уведомление о заявке
public void setGetNotify(boolean getNotify)	Устанавливает получение уведомления о заявке
public int getRequestID()	Возвращает идентификатор заявки
public void setRequestId(int requestId)	Устанавливает идентификатор заявки
public JSONObject getOfferData(boolean isId, boolean isMessage, boolean isUserId, boolean isRequestId)	Представляет информацию о предложении помощи в JSON-формате

## 1.1.5 Спецификация модуля «Status.java»

Модуль содержит атрибуты и функции (табл. П4.7).

Таблица П4.7

## Спецификация модуля «Status.java»

Атрибут	Описание
1	2
Атрибуты	
private int mStatusValue	Идентификатор статуса заявки. Значения 0 – требуется помощь, 1- помощь оказывается, 2- помощь оказана

Продолжение табл. П4.7

1	2
Функции	
public int getStatusValue()	Возвращает числовое значение статуса
public void setStatusValue(int statusValue)	Устанавливает числовое значение статуса
public String getStatusMessage()	Возвращает строковое значение статуса

### 1.1.6 Спецификация модуля «OfferList.java»

Модуль содержит атрибуты и функции (табл. П4.8)

Таблица П4.8

#### Спецификация модуля «OfferList.java»

Название	Описание
Атрибуты	
private static OfferList sOfferList	Ссылка на объект OfferList
private OfferManager mOfferManager	Ссылка на объект OfferManager
private OfferQuery mOfferQuery	Ссылка на объект OfferQuery
private Context mContext	Ссылка на контекст приложения
Функции	
public static OfferList get(Context context)	Возвращает объект класса OfferList, если он создан, и создает объект класса OfferList, если он не создан
private OfferList(Context context)	Конструктор, создание объекта
public List<Offer> getOffers(int requestId)	Возвращает список предложений помощи из базы данных приложения для заявки с идентификатором int requestId
public void setNotNewOffer(Offer offer)	Устанавливает, то, что заявка была просмотрена пользователем приложения

### 1.1.7 Спецификация модуля «RequestList.java»

Модуль содержит атрибуты и функции (табл. П4.9).

Таблица П4.9

#### Спецификация модуля «RequestList.java»

Название	Описание
1	2
Атрибуты	
private static RequestList sRequestList	Ссылка на объект RequestList
private RequestManager mRequestManager	Ссылка на объект RequestManager
private RequestQuery mRequestQuery	Ссылка на объект RequestQuery
private Context mContext	Ссылка на контекст приложения
Функции	
public static RequestList get(Context context)	Возвращает объект класса RequestList, если он создан, и создает объект класса RequestList, если он не создан

1	2
private RequestList(Context context)	Конструктор, создание объекта
public List<Request> getRequests(LatLng userLocation)	Получение списка заявок из базы SQLite с данным местоположением: в процессе получения вычисляем расстояние от пользователя до заявки по формуле гаверсинусов, если это расстояние не превышает радиус получения заявок, то добавляем заявку в список. В случае, если радиус получения заявок равен -1 (получать все заявки), то возвращаем все имеющиеся актуальные заявки из базы
public List<Request> getRequestsAppUser()	Получаем из базы приложения список заявок авторизованного пользователя
public List<Request> getNewRequest(LatLng userLocation)	Получаем из базы приложения список новых заявок на помощь. В случае, если радиус получения заявок равен -1 (получать все заявки), то возвращаем все имеющиеся актуальные заявки из базы
public List<Request> getRequestsAppUser()	Получаем из базы приложения список заявок авторизованного пользователя
public List<Request> getNewRequest(LatLng userLocation)	Получаем из базы приложения список новых заявок на помощь
public Request getRequest(int requestId)	Возвращаем из базы приложения заявку с идентификатором requestId
public void setNotNewRequest(Request request)	Устанавливаем, что заявка была просмотрена
public void setGetNotify(Request request)	Устанавливаем, что получено уведомление о заявке
public void setStatus(Request request)	Устанавливаем статус заявки

## 1.2 Спецификация модуля «LocationManager»

Модуль состоит из следующих файлов (табл. П4.10).

Таблица П4.10

### Файлы модуля «LocationManager»

Модуль	Описание
LocatingListener.java	Предоставляет интерфейс слушателя определения местоположения
LocationException.java	Предназначен для работы с исключениями, которые возникают в процессе определения местоположения устройства
LocationListener	Реализует класс для определения местоположения устройства, используя сервис FusedLocationProvider
GPSServices.java	
LocationPermission.java	Предоставляет функции для работы с разрешениями приложения на определение местоположения устройства

#### 1.2.1 Спецификация модуля «LocatingListener.java»

Модуль содержит функции (табл. П4.11).

Таблица П4.11

## Спецификация модуля «LocatingListener.java»

Название	Описание
Функции	
public void onUpdateLocation	Функция вызывается, когда местоположение пользователя определяется устройством

## 1.2.2 Спецификация модуля «LocationListenerGPServices.java»

Модуль содержит атрибуты и функции (табл. П4.12)

Таблица П4.12

## Спецификация модуля «LocationListenerGPServices.java»

Название	Описание
1	2
Атрибуты	
public static final int MILLISECONDS_PER_SECOND = 1000	Константа - количество миллисекунд в одной секунде
private Location mLocation	Текущее местоположение устройства (широта, долгота)
private GoogleApiClient mGoogleApiClient	Ссылка на клиент API сервисов Google для доступа к сервисам для определения местоположения
private Context mContext	Контекст приложения
private int mInterval	Интервал обновления местоположения
private int mFastUpdateInterval	Интервал обновления местоположения, используя значение местоположения, полученного другими приложениями
private int mNumUpdates=Integer.MAX_VALUE	Число обновлений местоположения
private LocatingListener mLocatingListener	Слушатель события изменения местоположения
Функции	
public LocationListenerGPService(Context context,int interval,int fastUpdateInterval)	Конструктор, создания слушателя обновления местоположения с заданным интервалом
public LocationListenerGPService(Context context,int interval,int fastUpdateInterval,int numUpdates)	Конструктор, создания слушателя обновления местоположения с заданным интервалом, числом обновлений
public void onConnected(@Nullable Bundle bundle)	Событие подключения к службам Google для определения местоположения, вызывает запуск получения обновлений местоположения
public void onConnectionSuspended(int i)	Событие остановки определения местоположения
public void onConnectionFailed(@NonNull ConnectionResult connectionResult)	Событие неуспешного подключения к сервисам Google
public void onLocationChanged(Location location)	Событие обновления местоположения, которое вызывает обновление поля текущего местоположения
public Location getCurrentLocation()	Возвращает последнее полученное местоположение устройства
public void setLocatingListener(LocatingListener locationRunnable)	Устанавливает слушателя locationRunnable на событие нахождения местоположения пользователя

## Продолжение табл. П4.12

1	2
public void startLocating()	Включает определения местоположения. Проверяет доступность GoogleApiClient, настраивает GoogleApiClient для работы с API Location Services и подключает его
public void stopLocating()	Выключает обновления местоположения
private LocationRequest locationRequestBuilder(int priority,int interval, int fastestInterval, int numUpdates)	Метод-обертка для построения запроса на определение местоположения у служб Google
private void enableLocationUpdates(int updateInterval,int fastUpdateInterval,int numUpdates)	Запускает определение местоположения, путем создания запроса LocationRequest и передачи его GoogleApiClient

## 1.2.3 Спецификация модуля «LocationPermission.java»

Модуль содержит атрибуты и функции (табл. П4.13)

Таблица П4.13

## Спецификация модуля «LocationPermission.java»

Название	Описание
Атрибуты	
private Context mContext	Контекст приложения
Функции	
public LocationPermission(final Context context)	Конструктор, создание объекта
public boolean checkAccessFineLocationPermission ()	Проверка в файле манифеста приложения разрешения на определение точного местоположения
public boolean checkAccessCoarseLocationPermission()	Проверка в файле манифеста приложения разрешения на определение приближенного местоположения
public boolean googlePlayServicesConnected()	Проверка подключения сервисов Google

## 1.3 Спецификация модуля «SQLiteDataBaseInterface»

Модуль состоит из следующих файлов (табл. П4.14).

Таблица П4.14

## Файлы модуля «SQLiteDataBaseInterface»

Модуль	Описание
1	2
HelpOnRoadBase.java	Предназначен для создания базы данных приложения на мобильном устройстве при первом запуске приложения
OfferCursorWrapper.java	Курсор-обертка для таблицы Offers

Продолжение табл. П4.14

1	2
OfferQuery.java	Предоставляет запросы к таблице Offers
RequestCursorWrapper.java	Курсор-обертка для таблицы Requests
RequestQuery.java	Предоставляет запросы к таблице Requests
SQLiteDataBaseShema.java	Структура, описывающая схему базы данных, содержит вложенные структуры, описывающие название таблиц: Requests, Offers, Users - базы данных SQLite (см. п. 5.12)
UserQuery.java	Предоставляет запросы к таблице Users

### 1.3.1 Спецификация модуля «HelpOnRoadBase.java»

Модуль содержит функции (табл. П4.15).

Таблица П4.15

#### Спецификация модуля «HelpOnRoadBase.java»

Название	Описание
Функции	
public HelpOnRoadBase(Context context)	Конструктор, создание объекта
public void onCreate(SQLiteDatabase db)	Создает базу данных с указанной в методе схемой при запуске устройства
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1)	Обновляет схему базы данных

### 1.3.2 Спецификация модуля «OfferCursorWrapper.java»

Модуль содержит функции (табл. П4.16)

Таблица П4.16

#### Спецификация модуля «OfferCursorWrapper.java»

Функция	Описание
Функции	
public OfferCursorWrapper(Cursor cursor)	Конструктор, создание объекта
public Offer getOffer()	Возвращает объект предложение Offer, заполненный данными из базы

### 1.3.3 Спецификация модуля «OfferQuery.java»

Модуль содержит следующие атрибуты и функции (табл. П4.17).

Таблица П4.17

## Спецификация модуля «OfferQuery.java»

Атрибут	Описание
Атрибуты	
private HelpOnRoadBase mHelpOnRoadBase	Ссылка на объект класса HelpOnRoadBase для обращения к базе данных приложения
private SQLiteDatabase mDatabase	Ссылка на объект класса SQLiteDatabase для подключения к базе данных приложения
private Context mContext	Контекст приложения
Функции	
public OfferQuery(Context context)	Конструктор, создание объекта
public void openConnection()	Открывает подключение к базе данных приложения
public void closeConnection()	Закрывает подключение к базе данных приложения
public boolean checkConnection()	Проверка подключения к базе данных приложения
public static ContentValues getOfferValues(Offer offer, boolean isId, boolean isMessage, boolean isCreateDate, boolean isUpdateDate, boolean isUploadDate, boolean isUserId, boolean isRequestId, boolean isNew, boolean isGetNotify)	Преобразует объект класса Offer в объект ContentValues (имя поля в БД, значение) для добавления и обновления записи в таблице Offers
public void addOffer(ContentValues offerContentValue)	Вставка данных предложения помохи в таблицу БД Offers
public void updateOffer(ContentValues requestContentValue)	Обновление записи в таблице Offers
public OfferCursorWrapper getOffers(int requestId)	Получение списка предложений заявки с идентификатором requestId из таблиц БД Offers, Users
public int getLastOfferId(int requestId)	Возвращает идентификатор последнего добавленного предложения для заявки с идентификатором requestId
public OfferCursorWrapper getOffers(int requestId)	Получение списка предложений помохи заявки с идентификатором requestId из таблиц БД Offers, Users
public int getLastOfferId(int requestId)	Возвращает идентификатор последнего добавленного предложения для заявки с идентификатором requestId

## 1.3.4 Спецификация модуля «RequestCursorWrapper.java»

Модуль содержит следующие функции (табл. П4.18).

Таблица П4.18

## Спецификация модуля «RequestCursorWrapper.java»

Название	Описание
Функция	
public RequestCursorWrapper(Cursor cursor)	Конструктор, создание объекта
public Request getRequest()	Возвращает объект предложение Request, заполненными данными из базы

### 1.3.5 Спецификация модуля «RequestQuery.java»

Модуль содержит атрибуты, аналогичные атрибутам модуля «OfferQuery» и функции (табл. П4.19)

Таблица П4.19

#### Спецификация модуля «RequestQuery.java»

Название	Описание
Функции	
public OfferQuery(Context context)	Конструктор, создание объекта
public void openConnection()	Открывает подключение к базе данных приложения
public void closeConnection()	Закрывает подключение к базе данных приложения
public boolean checkConnection()	Проверка подключения к базе данных приложения
public static ContentValues getRequestValues (Request request, boolean isId, boolean isMessage, boolean isLatitude, boolean isLongitude, boolean isImage, boolean isImage, boolean isCreateDate, boolean isUpdateDate, boolean isUploadDate, boolean isStatus, boolean isUserId, boolean isNew, boolean isGetNotify)	Преобразует объект класса Request в объект ContentValues (имя поля в БД, значение) для добавления и обновления записи в таблице Offers
public void addRequest(ContentValues offerContentValue)	Вставка заявки на помошь в таблицу БД Requests
public void addOrUpdateRequest(ContentValues requestContentValue)	Вставка или обновление записи, если запись уже существует, в таблице БД Requests
public void updateRequest(ContentValues requestContentValue)	Обновление записи в таблице Offers
public RequestCursorWrapper getRequests()	Получение списка заявок из таблиц БД Requests, Users
public RequestCursorWrapper getNewRequests(Driver driver)	Получение из списка новых заявок, которые не добавлены авторизованным пользователем приложения
public RequestCursorWrapper getRequestsAppUser(Driver driver)	Получение списка заявок, добавленных авторизованным пользователем приложения
public RequestCursorWrapper getRequest(int requestId)	Получение сведений о заявке с идентификатором id из базы из таблиц Requests, Users

### 1.4 Спецификация модуля «JSONParsers»

Модуль состоит из следующих файлов (табл. П4.20).

Таблица П4.20

## Файлы модуля «JSONParsers»

Модуль	Описание
DateParser.java	Предназначен для преобразования даты из строкового представления в объект Date и наоборот
OfferParser.java	Предназначен для разбора данных о предложениях помоши от серверной части приложения в формате JSON
ReplayParser.java	Класс для разбора данных от серверной части о статусе выполненного HTTP-запроса
RequestParser.java	Предназначен для разбора данных о заявках на помощь от серверной части приложения в формате JSON
RouteParser.java	Предназначен для разбора данных о маршруте от сервера Google
UserParser.java	Предназначен для разбора данных пользователях от серверной части приложения в формате JSON

## 1.4.1 Спецификация модуля «DateParser.java»

Модуль содержит атрибуты и функции (табл. П4.21).

Таблица П4.21

## Спецификация модуля «DateParser.java»

Название	Описание
Атрибуты	
private static final String DATE_FORMAT_STRING="yyyy-MM-dd HH:mm:ss"	Формат даты в строковом представлении
private static final String TIME_ZONE="GMT+3:00"	Часовой пояс приложения
Функции	
public static String parseDateToString(Date date)	Преобразует значение даты в строковое представление
public static Date parseStringToDate(String date)	Преобразует строковое значение даты, полученное от сервера к типу Date
public static void setTimeZone()	Устанавливает временную зону приложения по московскому времени
public boolean checkConnection()	Проверка подключения к базе данных приложения
public static String getDateSub(Date endDate, Date startDate)isRequestId, boolean isNew,boolean isGetNotify)	Вычисляет разницу между датами: endDate, startDate - в формате час. мин.

## 1.4.2 Спецификация модуля «OfferParser.java»

Модуль содержит функции (табл. П4.22).

Таблица П4.22

## Спецификация модуля «OfferParser.java»

Название	Описание
Функции	
public OfferParser(byte[] replay)	Конструктор, принимает байтовый массив, полученный от серверной части приложения
public List<Offer> getParsedOffers()	Разбирает, полученные данные о предложениях помоши в формате JSON, в список объектов Offer

## 1.4.3 Спецификация модуля «RequestParser.java»

Модуль содержит функции (табл. П4.23).

Таблица П4.23

## Функции модуля «RequestParser.java»

Название	Описание
Функции	
public OfferParser(byte[] replay)	Конструктор, принимает байтовый массив, полученный от серверной части приложения
public List<Offer> getParsedOffers()	Разбирает полученные данные о заявках помоши в формате JSON в список объектов Request

## 1.4.4 Спецификация модуля «UserParser.java»

Модуль содержит функции (табл. П4.24).

Таблица П4.24

## Спецификация модуля «UserParser.java»

Название	Описание
Функции	
public UserParser(byte[] replay)	Конструктор, принимает байтовый массив, полученный от серверной части приложения
public User getParsedUser(boolean isId, boolean isLogin, boolean isPassword, boolean isName, boolean isImage, boolean isCreateDate, boolean isUpdateDate)	Возвращает разобранные параметры пользователя приложения
public List<Driver> getParsedDrivers()	Разбирает полученные данные пользователей в формате JSON в список объектов User

#### 1.4.5 Спецификация модуля «RouteParser.java»

Модуль содержит атрибуты и функции (табл. П4.25).

Таблица П4.25

#### Спецификация модуля «RouteParser.java»

Название	Описание
Атрибуты	
private String mPoints	Массив закодированных точек, представляющий сглаженный маршрут
Функция	
public void parseReplayAboutRoute(String replay)	Разбирает ответ от сервера Google о маршруте между двумя точками в массиве точек
public static String addressParser(String replay)	Разбирает данные об адресе по текущему местоположению от сервера Google

#### 1.5 Спецификация модуля «ManagementObjectsOfModel»

Модуль состоит из следующих файлов (табл. П4.26).

Таблица П4.26

#### Файлы модуля «ManagementObjectsOfModel»

Модуль	Описание
FileManager.java	Предназначен для управления графическими файлами
GoogleAPIManager.java	Предназначен для формирования запросов к серверу Google (о маршруте и адресе)
OfferManager.java	Предназначен для формирования запросов к серверной части для добавления и получения списка заявок с данным идентификатором
PhotoManager.java	Предназначен для редактирования графических файлов
RequestException.java	Предназначен для работы с исключениями, возникающими при получении списка заявок
UserException.java	Предназначен для работы с исключениями при работе с данными пользователя приложения
UserManager.java	Предназначен для формирования запросов к серверной части для авторизации, регистрации пользователей в приложении
UserPreferences.java	Предназначен для работы с файлом настроек приложения

#### 1.5.1 Спецификация модуля «FileManager.java»

Модуль содержит атрибуты и функции (табл. П4.27).

Таблица П4.27

## Спецификация модуля «FileManager.java»

Название	Описание
Атрибуты	
String mFilePath	Путь к файлу
File mFile	Ссылка на редактируемый файл
Функции	
public FileManager(String filePath)	Конструктор, создание объекта
public File getFile()	Возвращает объект mFile
public static void setTimeZone()	Устанавливает временную зону приложения по московскому времени
public boolean checkConnection()	Проверка подключения к базе данных приложения
public void removeFile()	Функция удаления файла

## 1.5.2 Спецификация модуля «GoogleAPIManager.java»

Модуль содержит функции (табл. П4.28).

Таблица П4.28

## Спецификация модуля «GoogleAPIManager.java»

Название	Описание
Функции	
public String getRoute(LatLng originLoc,LatLng destinationLoc)	Получает данные о маршруте между начальной и конечной точками в формате JSON
public String getAddress(LatLng location)	Получает данные об адресе по значению местоположения

## 1.5.3 Спецификация модуля «OfferManager.java»

Модуль содержит функции (табл. П4.29).

Таблица П4.29

## Спецификация модуля «OfferManager.java»

Название	Описание
Функции	
public void addOffer(Offer offer)	Отправляет данные предложения помощи серверной части
public void getOffers(Context context,Offer offer)	Разбирает полученные данные о заявках помощи в формате JSON в список объектов Request
public boolean getNewOffers(Context context,Request request)	Получение списка новых предложений помохи к заявке request
public void parsedAndSaveOffersData(byte [] offersData,Context context)	Разбор данных о предложениях помохи в формате JSON от серверной части, сохранение разобранных данных в БД SQLite

### 1.5.4 Спецификация модуля «PhotoManager.java»

Модуль содержит атрибуты и функции (табл. П4.30).

Таблица П4.30

#### Спецификация модуля «PhotoManager.java»

Название	Описание
Атрибуты	
private File mFile	Ссылка на графический файл
Функции	
public PhotoManager()	Конструктор, создание нового файла
private void createImageFile()	Выделяет память для графического файла
public File getFile()	Возвращает файл
public static void addPhotoIntoGallery(String path, Activity activity)	Добавление файла в галерею устройства
public static Bitmap getScaledBitmap(String path, Activity activity)	Масштабирование графического файла
private static Bitmap getScaledBitmap(String path,int targetWidth,int targetHeight)	Масштабирование графического файла
public static void performCrop(Fragment fragment,Context context, int requestCode, Uri uri, int width, int height)	Кадрирование графического файла
public static File generateFileFromBitmap(Bitmap bitmap,String filePath)	Создание файла из объекта Bitmap
public static void loadImageIntoTarget(Activity activity, ImageView target, String URL,int width,int height)	Загрузка изображения по URL в графический компонент
public static void loadImageIntoTarget(Context activity,final ImageView target,final String URL, final boolean isLoadFromNetwork)	Загрузка изображения по URL в графический компонент, с проверкой наличия изображения в кеше приложения

### 1.5.5 Спецификация модуля «RequestManager.java»

Модуль содержит атрибуты и функции (табл. П4.31).

Таблица П4.31

#### Спецификация модуля «RequestManager.java»

Название	Описание
1	2
Функции	
public void addRequest(Request request)	HTTP-запрос серверной части для добавления новой заявки на помошь в базу данных MySQL
public void updateRequestStatus(Context context, Request request)	HTTP-запрос серверной части для изменения статуса заявки request

## Продолжение табл. П4.31

1	2
public String generateNameRequestFile(Context mContext)	Генерация уникального имени графического файла заявки в мобильном приложении
public void getActualRequest(Context context)	HTTP-запрос серверной части для получения списка актуальных заявок из базы данных MySQL
public void parsedAndSaveRequestsData(byte [] requestData,Context context)	Разбор списка заявок от серверной части в формате JSON, сохранение разобранных данных в БД SQLite
public void parsedAndSaveUserData(byte [] userData,Context context)	Разбор списка пользователей от серверной части в формате JSON, сохранение разобранных данных в БД SQLite
public static double calculateDistance(Request mRequest, LatLng userLocation)	Вычисляет расстояние по формуле гаверсинусов между местоположением заявки и текущим местоположением пользователя
private static double calculateDistance(LatLng location1,LatLng location2)	Вычисляет расстояние по формуле гаверсинусов между двумя координатами на земной поверхности
public String getRequestImageURL(Request request)	Формирует строку URL к фотографии заявки
public static String getRequestMarkerTitle(Request request)	Возвращает заголовок маркера заявки
public static float getRequestMarkerColor(Request request)	Возвращает цвет маркера заявки

## 1.5.6 Спецификация модуля «UserManager.java»

Модуль содержит атрибуты и функции (табл. П4.32).

Таблица П4.32

## Спецификация модуля «UserManager.java»

Название	Описание
1	2
Атрибуты	
private UserPreferences mUserPreferences	Ссылка на файл настроек пользователя
Функции	
public User getUser()	Формирует объект пользователя из настроек приложения
public UserManager(UserPreferences userPreferences)	Конструктор, создание объекта
public void logOut()	Выход пользователя из приложения
public boolean isLoggedIn()	Проверка, авторизован ли пользователь в приложении
public void registerNewUser(User user)	HTTP-запрос на регистрацию пользователя в приложении
public void authorizedUser(User user)	HTTP-запрос на авторизацию пользователя в приложении
public void loadUserImage (User user, Context mContext)	HTTP-запрос на загрузку изображения пользователя
public void deleteUserImage (User user)	HTTP-запрос на удаление пользователя
public void changeUserName(User user)	HTTP-запрос на изменение имени пользователя

## Продолжение табл. П4.32

1	2
public void changeUserPassword (User user)	HTTP-запрос на изменение пароля пользователя
public void changeUserImage(User user, String filepath)	HTTP-запрос на изменение изображения пользователя
public String generateNameUserFile(Context mContext)	Генерирует имя пользовательского файла в закрытом хранилище файлов приложения
public String getUserImageURL(Driver driver)	Строка HTTP-запроса на получение изображения пользователя

## 1.5.7 Спецификация модуля «UserPreferences.java»

Модуль содержит атрибуты и функции (табл. П4.33).

Таблица П4.33

## Спецификация модуля «UserPreferences.java»

Название	Описание
1	2
Атрибуты	
private Context mContext	Ссылка на контекст приложения
Функции	
public UserPreferences (Context context)	Конструктор, создание объекта
public String getStoredUserId()	Получение идентификатора пользователя из файла настроек
public void setStoredUserId(String userId)	Запись идентификатора пользователя в файл настроек приложения
public String getStoredUserLogin()	Получение логина пользователя из файла настроек
public void setStoredUserLogin(String login)	Запись логина пользователя в файл настроек
public String getStoredUserPassword ()	Получение пароля пользователя из файла настроек
public void setStoredUserPassword (String password)	Запись пароля пользователя в файл настроек
public String getStoredUserName()	Получение имени пользователя из файла настроек
public void setStoredUserName (String name)	Запись имени пользователя в файл настроек
public boolean getStoredUserIsImage()	Получение наличия фотографии пользователя из файла настроек
public void setStoredUserIsImage(boolean isImage)	Запись наличия фотографии пользователя в файл настроек
public String getStoredUserImage()	Получение имени изображения пользователя из файла настроек
public void setStoredUserImage (String image)	Запись имени изображения пользователя в файл настроек
public boolean getStoredIsGetNotificationsParameter ()	Получение параметра получения уведомлений из файла настроек
public void setStoredIsGetNotifications Parameter (boolean isGetNotification)	Запись параметра получения уведомлений в файл настроек
public int getStoredRadiusNotifications()	Получение радиуса заявок из файла настроек
public void setStoredRadiusNotifications (int radiusNotifications)	Запись радиуса получения заявок в файл настроек приложения

## Продолжение табл. П4.33

1	2
public int getStoredFrequencyNotifications()	Получение частоты получения заявок из файла настроек
public void setStoredFrequencyNotifications (int frequencyNotifications)	Запись частоты получения заявок в файл настроек

## 1.6 Спецификация модуля «ManagementAddNewRequest»

Модуль состоит из следующих файлов (табл. П4.34).

Таблица П4.34

## Файлы модуля «ManagementAddNewRequest»

Модуль	Описание
AddNewRequest Activity.java	Реализует окно приложения для добавления новой заявки. Содержит классы: <ul style="list-style-type: none"> <li>- AddNewRequestActivity – окно для добавления новой заявки;</li> <li>- AddNewRequestTask – класс, представляющее фоновое задание для отправки данных заявки на сервер в методе doInBackground()</li> </ul>
AddNewRequest Fragment.java	Содержит классы: <ul style="list-style-type: none"> <li>- AddNewRequestFragment – фрагмент окна для добавления новой заявки;</li> <li>- SelectRequestPhotoListener – интерфейс обратного вызова, который содержит метод для передачи выбранного изображения от активности фрагменту активности;</li> <li>- EnterMessageListener – интерфейс обратного вызова, который содержит метод для передачи введенного сообщения от активности фрагменту активности.</li> </ul>
AddNewRequest MapFragment.java	Реализует фрагмент окна для добавления новой заявки, содержащий карту с отметкой местоположения пользователя

## 1.6.1 Спецификация модуля «AddNewRequestActivity.java»

Модуль содержит атрибуты и функции (табл. П4.35).

Таблица П4.35

## Спецификация модуля «AddNewRequestActivity.java»

Название	Описание
1	2
Атрибуты	
private Request mRequest	Ссылка на объект заявка - Request
private UserManager mUserManager	Ссылка на объект UserManager
private LocationListenerGPSServices mLocationListener	Ссылка на объект LocationListenerGPSServices
private LatLng mSelectedLocation	Выбранное местоположение

## Продолжение табл. П4.35

1	2
private Bitmap mSelectedImage	Выбранное местоположение
private Bitmap mSelectedImage	Выбранное местоположение
private UpdateLocationListener mUpdateLocationListener	Ссылка на слушателя изменения местоположения
private ProgressDialog pDialog	Графический компонент для отображения прогресса выполнения фонового задания
private CheckLogInDialog mCheckLogInDialog	Диалог для ввода пароля пользователя
	Функции
protected void addFragments (ViewPagerAdapter viewPagerAdapter)	Добавление фрагментов к окну
public void onCreate(@Nullable Bundle savedInstanceState)	Событие создания окна
protected void onDestroy()	Событие уничтожения окна
protected void onStart()	Событие старта окна: запускает определение местоположения
protected void onStop()	Событие остановки окна: приостанавливает определения местоположения
public boolean onCreateOptionsMenu (Menu menu)	Событие создания меню окна
public boolean onOptionsItemSelected (MenuItem item)	Событие нажатие на пункты меню окна, которое содержит пункт, при нажатии на который происходит передача данных заявки серверной части приложения
public void onUpdateLocation(final Location location)	Получает обновленное местоположение пользователя
public void onUpdateSelectedLocation (LatLng selectedLocation)	Получает выбранное на карте местоположение пользователя
public void onSelectImage(Bitmap bitmap)	Получает выбранное изображение пользователя
public void onEnterMessage(String message)	Получает введенное сообщение пользователя

## 1.6.2 Спецификация модуля «AddNewRequestFragment.java»

Модуль содержит атрибуты и функции (табл. П4.36).

Таблица П4.36

## Спецификация модуля «AddNewRequestFragment.java»

Название	Описание
1	2
	Атрибуты
private Button mImageSelectButton	Кнопка для выбора изображения
private Button mImageDeleteButton	Кнопка для отмены выбранного изображения
private TextView mMessageTextView	Текстовое поле для сообщения заявки
private String mEnteredMessage	Сообщение заявки
private EnterMessageListener mEnter MessageListener	Слушатель ввода текста сообщения
private SelectRequestPhotoListener mSelectRequestPhotoListener	Слушатель выбора изображения

## Продолжение табл. П4.36

1	2
private ImageView mRequestPhoto ImageView	Графический компонент для отображения изображения заявки
private Bitmap mRequestPhotoBitmap	Изображение заявки
private AlertDialog.Builder mSelectCameraOrAlbumDialog	Диалог для выбора снятия изображения на камеру или выбора из файловой системы устройства
	Функции
public void onCreate(@Nullable Bundle savedInstanceState)	Событие создания фрагмента окна
public void onViewStateRestored(@Nullable Bundle savedInstanceState)	Событие восстановления окна из памяти
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState)	Событие заполнения окна из файла разметки интерфейса окна
public void onAttach(Activity activity)	Событие прикрепления фрагмента к активности
public void onActivityResult(int requestCode, int resultCode, Intent data)	Событие обработки закрытия запущенного приложения для выбора изображения: сохраняет выбранное пользователем изображение
private void onGetPictureFromCamera()	Получение изображения с камеры и его кадрирование
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults)	Обработка результата получения разрешения на доступ к камере мобильного устройства

## 1.6.3 Спецификация модуля «AddNewRequestMapFragment.java»

Модуль содержит атрибуты и функции (табл. П4.37).

Таблица П4.37

## Спецификация модуля «AddNewRequestMapFragment.java»

Название	Описание
1	2
Атрибуты	
private Location mCurrentLocation	Текущее местоположение пользователя
private Marker mSelectedLocationMarker	Маркер, показывающий выбранное местоположение пользователя на карте
private boolean isSetCameraPosition	Установлена ли камера на карте
private boolean isUserSelectedCurrent Location	Выбрал ли пользователь местоположение на карте
Функции	
public void onCreate(Bundle bundle)	Событие создания карты: устанавливает позицию камеры, местоположение пользователя, содержит обработчики уточнения местоположения пользователем
public void onUpdateLocation(Location location)	Событие обновления текущего местоположения пользователя устройства
public void setOrUpdateSelectedLocation Marker()	Устанавливает маркер, который отображает выбранное пользователем местоположение

## Продолжение табл. П4.37

1	2
public boolean onMarkerClick(Marker marker)	Событие нажатия маркера пользователем, при котором в информационном окне маркера отображается информация об адресе местоположения
public void updatePositionSelectedLocation MarkerAfterClick (LatLng position)	Изменяет положение маркера на карте после клика пользователем карты
public void setCameraPosition()	Устанавливает положение камеры на карте, проверяя значение флага isSetCameraPosition и mCurrentLocation на null
public void onAttach(Activity activity)	Событие добавления фрагмента к активности: задает слушателя выбранного местоположения на карте

## 1.7 Спецификация модуля «ManagementListRequests»

Модуль состоит из следующих файлов (табл. П4.38).

Таблица П4.38

## Файлы модуля «ManagementListRequests»

Модуль	Описание
GetAddress RequestTask.java	Реализует фоновое задание, которое содержит метод doInBackground (Void... params) для выполнения запроса к серверу Google для получения адреса местоположения. Выполняется в отдельном потоке
MainActivity.java	Реализует работу с главным окном приложения, содержащее вкладки для просмотра списка заявок и отметок заявок на карте
RequestList Fragment.java	Содержит классы: <ul style="list-style-type: none"> <li>- RequestList Fragment.java - реализует фрагмент для просмотра списка заявок на помощь;</li> <li>- RequestHolder - описывает графический компонент списка, содержит метод bindRequest, который связывает объект Request с его графическим компонентом;</li> <li>- RequestAdapter – предназначен для управления списком заявок в графическом компоненте RecyclerView, добавляет в компонент объекты класса RequestHolder и реализует их связывание с объектами класса Request;</li> <li>- GetActualRequestsTask – класс, содержащий метод doInBackground(Void... params), реализующий фоновое задание в отдельном потоке для выполнения HTTP-запроса серверной части на получение списка заявок на помощь</li> </ul>
RequestList MapFragment.java	Реализует фрагмент для просмотра отметок заявок на карте.
NewRequest FetcherService.java	Фоновый сервис, который обеспечивает HTTP-запрос на сервер, проверяющий наличие новых заявок и посыпает пользователю уведомление на устройство
RequestDetail Activity.java	Содержит классы: RequestDetailActivity - реализует окно для просмотра сведений детальной заявки, содержащее вкладки для просмотра сведений о детальной заявке и построения маршрута до заявки;

## Продолжение табл. П4.38

1	2
RequestDetail Fragment.java	<p>Содержит классы:</p> <ul style="list-style-type: none"> <li>- RequestDetail Fragment - реализует фрагмент для просмотра списка заявок на помощь;</li> <li>- OfferHolder- описывает графический компонент списка, содержит метод bindOffer, который связывает объект Offer с его графическим компонентом;</li> <li>- OfferAdapter – предназначен для управления списком заявок в компоненте RecyclerView, добавляет в компонент объекты OfferHolder;</li> <li>- AddNewOfferTask – реализует запрос на сервер на добавление предложения помощи к заявке;</li> <li>- UpdateOffersTask – запрос на сервер для обновления списка заявок;</li> <li>- UpdateRequestStatusTask – запрос на сервер для обновления статуса заявки;</li> <li>- GetNewOffersTask – запрос на сервер для получения новых предложений помощи</li> </ul>
RequestDetail MapFragment.java	Содержит классы: RequestDetailMapFragment - реализует фрагмент для просмотра маршрута заявки; GetRouteTask – выполняет получение данных маршрута и его построение на карте
SelectNavigation ViewItemListener.java	Слушатель события выбора пользователем свойства просматривать только свои заявки

## 1.7.1 Спецификация модуля «MainActivity.java»

Модуль содержит атрибуты и функции (табл. П4.39).

Таблица П4.39

## Спецификация модуля «MainActivity.java»

Название	Описание
1	2
Атрибуты	
private NavigationView mNavigationView	Главное меню приложения
private UserManager mUserManager	Ссылка на объект UserManager
private LocationListenerGPSServices mLocationListener	Ссылка на объект LocationListenerGPSServices
private SelectNavigationViewItemListener mSelectNavigationViewItemRequestListListener	Слушатель выбора пункта меню приложения
private SelectNavigationViewItemListener mSelectNavigationViewItemRequestListMapListener	Слушатель выбора пункта меню приложения
private UpdateLocationListener mUpdateLocationRequestListFragmentListener	Слушатель события обновления местоположения
private UpdateLocationListener mUpdateLocationRequestListMapFragmentListener	Слушатель события обновления местоположения
Функции	

## Продолжение табл. П4.39

1	2
protected void addFragments(ViewPagerAdapter viewPagerAdapter)	Добавляет вкладки окна для просмотра списка заявок и отметок заявок
public void onCreate(Bundle savedInstanceState)	Событие создания окна
protected void onStart()	Событие старта окна
protected void onStop()	Событие приостановки окна
public void onResume()	Событие запуска окна
public void onBackPressed()	Событие нажатия клавиши Back
public boolean onNavigationItemSelected(MenuItem item)	Событие выбора пункта меню, которое запускает требуемое окно приложения
private void deleteNotActualRequestsAndUsers()	Удаление заявок на помощь из локальной базы данных, которые добавлены более суток назад

## 1.7.2 Спецификация модуля «RequestListFragment.java»

Модуль содержит атрибуты и функции (табл. П4.40).

Таблица П4.40

## Спецификация модуля «RequestListFragment.java»

Название	Описание
1	2
Атрибуты	
private RecyclerView mRequestRecycleView	Графический компонент: список заявок
private RequestAdapter mAdapter	Объект для управления заполнения графического компонента mRequestRecycleView
private boolean isLoadRequestsList	Список заявок загружен
private Location mCurrentLocation	Текущее местоположение
private ProgressBar mProgressBar	Прогресс-диалог
Функции	
public void onShowRequestOnlyAppUser(boolean isShowOnlyUserRequest)	Событие показа заявок только пользователя приложения
public void onCreate(@Nullable Bundle savedInstanceState)	Событие создания окна
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState)	Заполнение окна интерфейсными компонентами из файла разметки приложения
public void onStart()	Событие запуска окна
public void onActivityResult(int requestCode, int resultCode, Intent data)	Событие получения результата от активности
private void updateUI()	Обновление списка заявок

## 1.7.3 Спецификация модуля «RequestDetailMapFragment.java»

Модуль содержит атрибуты и функции (табл. П4.41).

Таблица П4.41

## Спецификация модуля «RequestListMapFragment.java»

Название	Описание
Атрибуты	
private GoogleMap mMap	Карта
private Marker mSelectedLocationMarker	Выбранное местоположение
Integer mClickedRequestId	Кликнутая заявка
private Location mCurrentLocation	Текущее местоположение
private List<Marker> mRequestsMarkers	Список маркеров-заявок
private RequestList requestList	Объект для управления списком заявок
Функции	
public void onShowRequestOnlyAppUser(boolean isShowOnlyUserRequest)	Событие показа заявок только пользователя приложения
public void onCreate(Bundle bundle)	Событие создания карты
public boolean onMarkerClick(Marker marker)	Событие клика маркера заявки
public void onUpdateLocation(Location location)	Событие обновления местоположения
public void setMapContent()	Заполнение карты списком маркеров
private void setCameraPosition(List<Request> requests)	Устанавливает положение камеры на карте
private void setPositionRequestsMarkers(List<Request> requests)	Устанавливает положение маркеров заявок на карте
private void setOrUpdateSelectedLocationMarker()	Обновление положения маркера текущего местоположения пользователя

## 1.7.4 Спецификация модуля «RequestDetailActivity.java»

Модуль содержит атрибуты и функции (табл. П4.42).

Таблица П4.42

## Спецификация модуля «RequestDetailActivity.java»

Название	Описание
Атрибуты	
private int mRequestId	Идентификатор выбранной заявки
private boolean isShowUserRequests	Показывать только заявки пользователя приложения
private UpdateLocationListener mRequestDetailFragmentLocationListener	Слушатель события обновления местоположения
private RequestStatusListener mRequestMapFragmentStatusListener	Слушатель изменения статуса заявки
Функции	
public static Intent newIntent(Context packageContext, int requestId, boolean isShowUserRequests)	Метод для запуска активности RequestDetailActivity
protected void addFragments(ViewPagerAdapter viewPagerAdapter)	Добавление фрагментов в активность
public void onCreate(@Nullable Bundle savedInstanceState)	Событие создания окна

## Продолжение табл. П4.42

1	2
protected void onStart()	Событие запуска активности
protected void onStop()	Событие приостановки выполнения активности
public void onUpdateLocation(Location location)	Событие обновления местоположения
public void onRequestStatusChange(Request request)	Событие изменения статуса заявки

## 1.7.5 Спецификация модуля «RequestDetailFragment.java»

Модуль содержит атрибуты и функции (табл. П4.43).

Таблица П4.43

## Спецификация модуля «RequestDetailFragment.java»

Название	Описание
1	2
Атрибуты	
private Request mRequest	Просматриваемая заявка
private UserManager mUserManager	Объект для управления пользователем
private RequestManager mRequestManager	Объект для управления заявками
private Location mCurrentLocation	Текущее местоположение
private Offer mOffer	Объект предложения помощи
private ImageView mUserPhotoImageView	Графический компонент – изображение пользователя
private TextView mUserNameTextView	Тестовое поле – имя пользователя
private TextView mStatusTextView	Тестовое поле – статус заявки
private Spinner mStatusSpinner	Выпадающий список статусов заявок
private TextView mMessageTextView	Текстовое поле – сообщение заявки
private TextView mAddDateTextView	Тестовое поле – дата добавления заявки
private TextView mDistanceTextView	Тестовое поле – расстояние между пользователем и заявкой
private ImageView mRequestPhotoImage	Графический компонент – фото заявки
private RecyclerView mOffersRecycleView	Графический компонент – список заявок
private EditText mOfferMessageEditText	Поле для добавления сообщения помощи
private ImageButton mSendImageButton	Кнопка для отправки сообщения помощи
private ProgressBar mProgressBar	Прогресс-диалог
private OfferAdapter mAdapter	Объект для управления списком заявок mOffersRecycleView
private Timer mTaskTimer	Таймер
Функции	
public static RequestDetailFragment newInstance(int requestId, boolean isShowUserRequests)	Создание фрагмента для просмотра сведений детальной заявки
public void onCreate(@Nullable Bundle savedInstanceState)	Событие создания фрагмента: выполняет получение списка предложений помощи к заявке, заполняет графический компонент

## Продолжение табл. П4.43

1	2
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState)	Заполнение компонента из файла разметки с расширением .xml
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater)	Создание меню окна
public boolean onOptionsItemSelected(MenuItem item)	Событие нажатия пункта меню окна для изменения статуса просматриваемой заявки пользователя приложения
public void onStart()	Запуск окна
public void onDestroy()	Уничтожение окна
public void onUpdateLocation(Location location)	Событие обновления текущего местоположения пользователя
private void updateUI()	Обновление списка заявок
private void checkNewOffers()	Проверка наличия новых предложений помощи в системе

## 1.7.6 Спецификация модуля «RequestDetailMapFragment.java»

Модуль содержит атрибуты и функции (табл. П4.44).

Таблица П4.44

## Спецификация модуля «RequestDetailMapFragment.java»

Название	Описание
1	2
Атрибуты	
private Location mCurrentLocation	Текущее местоположение
private Location mPreviousLocation	Предыдущее местоположение
private Marker mCurrentLocationMarker	Маркер текущего местоположения пользователя
private Polyline mRouteLine	Линия маршрута
private Request mRequest	Выбранная заявка
private Marker mRequestMarker	Маркер выбранной заявки
Функции	
public static RequestDetailMapFragment newInstance(int requestId)	Создание фрагмента с переданным идентификатором заявки
public void onCreate(Bundle bundle)	Метод создания карты: выполняет ее получение, устанавливает камеру, маркеры местоположения пользователя, местоположения заявки, строит маршрут
public void onStart()	Метод запуска фрагмента окна
private void setRequest()	Заполнение переданного объекта заявка данными из локальной базы
public void onUpdateLocation(Location location)	Метод обновления местоположения пользователя
public void setCameraPosition()	Устанавливает положение камеры на карте
public void setSelectedRequestLocationMarkers()	Добавляет маркер заявки на карту

Продолжение табл. П4.44

1	2
public void setSelectedLocationMarker()	Добавляет маркер выбранного местоположения пользователя
public boolean onMarkerClick(Marker marker)	Показывает информацию об адресе нажатого маркера
public void decodeAndPaintRoute()	Декодирует полученные данные о маршруте в массив точек и рисует линию маршрута на карте
public void onRequestStatusChange(Request request)	Событие изменения статуса заявки, которое вызывает изменение цвета маркера и надпись

### 1.7.7 Спецификация модуля «NewRequestFetcherService.java»

Модуль содержит атрибуты и функции (табл. П4.45).

Таблица П4.45

#### Спецификация модуля «NewRequestFetcherService.java»

Название	Описание
Атрибуты	
private LocationListenerGPSServices mLocationListener	Слушатель GPS
private Location mCurrentLocation	Текущее местоположение
Функции	
public static void startNew RequestFetcherService (Context context)	Запуск фонового сервиса
public static void stopNew RequestFetcherService (Context context)	Уничтожение фонового сервиса
public void onCreate()	Создание фонового сервиса: запуск определения местоположения устройства
public int onStartCommand(Intent intent, int i, int i1)	Запуск команды
public int onRunTask (TaskParams taskParams)	Выполнение фонового сервиса: запрос на сервер для получения списка актуальных заявок, сравнение заявок с последней просмотренной заявкой пользователем устройства, отправка пользователю уведомления о новой заявке
private boolean AvailableAndConnected()	Проверка доступности сети
public void showNotification (Request request)	Показывает уведомление о заявке

### 1.8 Спецификация модуля «ManagementListRequests»

Пакетный модуль состоит из следующих файлов (табл. П4.46).

Таблица П4.46

## Файлы модуля «ManagementUser»

Модуль	Описание
1	2
CheckLogIn Dialog.java	Описывает диалоговое для авторизации пользователя в приложении
EnterActivity.java	Активность для авторизации в приложении, содержит метод createFragment для добавления фрагмента активности и метод onCreate() для создания активности
EnterFragment.java	Содержит классы: – EnterFragment.java - реализует фрагмент окна для авторизации в приложении; – AuthorizeNewUserTask – класс, содержащий метод doInBackground(Void... params), реализующий фоновое задание в отдельном потоке для выполнения HTTP-запроса серверной части на авторизацию пользователя в приложении
Registration Activity.java	Активность для регистрации в приложении, содержит метод createFragment для добавления фрагмента активности и метод onCreate() для создания активности
Registration Fragment.java	Содержит классы: – RegistrationFragment.java - реализует фрагмент окна для регистрации в приложении; – RegisterNewUserTask – класс, содержащий метод doInBackground(Void... params), реализующий фоновое задание в отдельном потоке для выполнения HTTP-запроса серверной части на регистрацию нового пользователя в приложении
SetUserName Dialog.java	Содержит классы: – SetUserNameDialog – диалог для изменения имени пользователя приложения, который в методе onCreate делает соответствующий HTTP-запрос серверной части приложения, используя объект класса ChangeNameUserTask; – ChangeNameUserTask – класс, содержащий метод doInBackground(Void... params), реализующий фоновое задание в отдельном потоке для выполнения HTTP-запроса серверной части на изменение имени авторизованного пользователя в приложении
SetUserPassword Dialog.java	Содержит классы: – SetUserPasswordDialog – диалог для изменения пароля пользователя приложения, который в методе onCreate: проверяет введенное значение пароля, путем сравнения со значением подтверждения пароля, делает соответствующий HTTP-запрос серверной части приложения, используя объект класса ChangeNameUserTask; – ChangePasswordUserTask – класс, содержащий метод doInBackground(Void... params), реализующий фоновое задание в отдельном потоке для выполнения HTTP-запроса серверной части на изменение имени авторизованного пользователя в приложении
UserAccount Activity.java	Активность для просмотра сведений учетной записи пользователя в приложении, содержит метод createFragment для добавления фрагмента активности и метод onCreate() для создания активности
UserAccount Fragment.java	Содержит классы: UserAccount Fragment - фрагмент для изменения свойств учетной записи пользователя; ChangeImageUserTask - класс, содержащий метод doInBackground(Void... params), реализующий фоновое задание в отдельном потоке для выполнения

## Продолжение табл. П4.46

1	2
	HTTP-запроса серверной части на отправку нового файла изображения пользователя; DeleteImageUserTask - класс, содержащий метод doInBackground (Void... params), реализующий фоновое задание в отдельном потоке для выполнения HTTP-запроса серверной части на удаление файла изображения пользователя
UserSettings Activity.java	Активность для изменения настроек пользователя в приложении, содержит метод createFragment для добавления фрагмента активности и метод onCreate() для создания активности
UserSettings Fragment.java	Реализует изменение настроек пользователя, связанных с получением уведомлений: включение получения уведомлений, радиус, частота получения уведомлений

## 1.8.1 Спецификация модуля «EnterFragment.java»

Модуль содержит атрибуты и функции (табл. П4.47).

Таблица П4.47

## Спецификация модуля «EnterFragment.java»

Название	Описание
1	2
Атрибуты	
private Button mRegisterButton	Кнопка для регистрации пользователя
private Button mEnterButton	Кнопка для перехода к окну для авторизации пользователя
private EditText mLoginEditText	Тестовое поле – логин пользователя
private EditText mPasswordEditText	Тестовое поле – пароль пользователя
private CheckBox mShowPasswordCheckBox	Переключатель для скрытия и показа пароля пользователя
private ProgressDialog pDialog	Прогресс-диалог
private User mUser	Объект, описывающий данные пользователя
Функции	
public void onCreate(@Nullable Bundle savedInstanceState)	Создание окна
public void onSaveInstanceState(Bundle outState)	Сохранение конфигурации окна перед поворотом устройства
public void onViewStateRestored(@Nullable Bundle savedInstanceState)	Востановление конфигурации окна после поворота устройства
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState)	Создание графических компонентов окна из файла разметки с расширением .xml, задание обработчиков кнопок для авторизации

## 1.8.2 Спецификация модуля «RegistrationFragment.java»

Модуль содержит атрибуты и функции (табл. П4.48).

Таблица П4.48

## Спецификация модуля «RegistrationFragment.java»

Название	Описание
1	2
Атрибуты	
private Button mRegisterButton	Кнопка для регистрации пользователя
private Button mEnterButton	Кнопка для перехода к окну для авторизации пользователя
private EditText mLoginEditText	Тестовое поле – логин пользователя
private EditText mPasswordEditText	Тестовое поле – пароль пользователя
private EditText mConfirmPasswordEditText	Текстовое поле – подтверждение пароля
private CheckBox mShowPasswordCheckBox	Переключатель для скрытия и показа пароля пользователя
private CheckBox mShowConfirmPasswordCheckBox	Переключатель для скрытия и показа текстовой строки для подтверждения пароля пользователя
private ProgressDialog pDialog	Прогресс-диалог
private User mUser	Объект, описывающий данные пользователя
private String mConfirmPassword	Подтверждение пароля
Функции	
public void onCreate(@Nullable Bundle savedInstanceState)	Создание окна
public void onSaveInstanceState(Bundle outState)	Сохранение конфигурации окна перед поворотом устройства
public void onViewStateRestored(@Nullable Bundle savedInstanceState)	Востановление конфигурации окна после поворота устройства
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState)	Создание графических компонентов окна из файла разметки с расширением .xml, задание обработчиков кнопок для регистрации, перехода к процессу авторизации в приложении

## 1.8.3 Спецификация модуля «UserAccountFragment.java»

Модуль содержит атрибуты и функции (табл. П4.49).

Таблица П4.49

## Спецификация модуля «UserAccountFragment.java»

Название	Описание
1	2
Атрибуты	
private UserManager mUserManager	Объект для управления пользователем приложения
private PhotoManager mPhotoManager	Объект для управления фотографиями
private TextView mLoginTextView	Тестовое поле - логин
private TextView mLogOutTextView	Тестовое поле – выйти из приложения
private TextView mNameTextView	Тестовое поле – имя пользователя

## Продолжение табл. П4.49

1	2
private TextView mPasswordTextView	Тестовое поле – пароль пользователя
private SetUserNameDialog mSetUserNameDialog	Диалог для задания имени пользователя
private SetUserPasswordDialog mSetUserPasswordDialog	Диалог для задания пароля пользователя
private AlertDialog.Builder mLogOutAlertDialog	Диалог для выхода из приложения
private AlertDialog.Builder mSelectCameraOrAlbumDialog	Диалог для выбора способа получения фотографии
private ImageButton mUserPhotoButton	Графический компонент – изображение пользователя
private ImageButton mSaveUserImageButton	Кнопка сохранения изображения пользователя
private ImageButton mDeleteUserImageButton	Кнопка удаления изображения пользователя
Функции	
public void onCreate(@Nullable Bundle savedInstanceState)	Создание окна
public void onSaveInstanceState(Bundle outState)	Сохранение конфигурации окна перед поворотом устройства
public void onViewStateRestored(@Nullable Bundle savedInstanceState)	Востановление конфигурации окна после поворота устройства
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState)	Создание графических компонентов окна из файла разметки с расширением .xml, получение данных о пользователе и заполнение текстовых полей, задание обработчиков кнопок
public void onActivityResult(int requestCode, int resultCode, Intent data)	Получение выбранного изображение с камеры или галереи устройства, запуск операции кадрирования, сохранение изображения в памяти устройства, вывод в графический компонент
private void onGetPictureFromCamera()	Получение изображения с камеры устройства и его кадрирование

## 1.8.4 Спецификация модуля «UserSettingsFragment.java»

Модуль содержит атрибуты и функции (табл. П4.50).

Таблица П4.50

## Спецификация модуля «UserSettingsFragment.java»

Название	Описание
1	2
Атрибуты	
private SwitchCompat mGetNotificationsSwitchCompat	Переключатель для включения/отключения уведомлений
private Spinner mRadiusNotificationSpinner	Выпадающий список для выбора радиуса получения уведомлений
private Button mAppPermissionsButton	Кнопка для перехода к настройкам приложения в системе

Продолжение табл. П4.50

1	2
private UserPreferences mUserPreferences	Объект для работы с настройками пользователя приложения
private Spinner mFrequencyNotificationSpinner	Выпадающий список для выбора частоты получения уведомлений
	Функции
public void onCreate(@Nullable Bundle savedInstanceState)	Создание окна
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState)	Создание графических компонентов окна из файла разметки с расширением .xml, получение данных о пользователе и заполнение текстовых полей, задание значений выпадающим спискам, задания обработчиков изменения значений выпадающих списков частоты и радиуса уведомлений
private ArrayAdapter<String> getSpinnerRadiusNotificationsAdapter()	Создание адаптера для выпадающего списка RadiusNotificationsSpinner, заполнение списка значениями
private ArrayAdapter<String> getSpinnerFrequencyNotificationsAdapter()	Создание адаптера для выпадающего списка FrequencyNotificationsSpinner, заполнение списка значениями
private List<String> generateValuesRadiusNotificationSpinner()	Генерация списка значений радиуса получения уведомлений от 1 до 100км или получать все заявки
private List<String> generateValuesFrequencyNotificationSpinner()	Генерация списка значений частоты уведомлений от 1 до 60мин

## 1.9 Спецификация модуля «NetworkOperation»

Модуль состоит из следующих файлов (табл. П4.51).

Таблица П4.51

### Файлы модуля «NetworkOperation»

Модуль	Описание
HttpsQuery.java	Реализует обертку методов POST, GET сетевого протокола HTTP, используя класс HttpURLConnection
MySQL DataBaseSchema.java	Вложенная структура, которая содержит строки-константы, которые содержат названия полей таблиц серверной БД MySQL (см. 5.11)

#### 1.9.1 Спецификация модуля «HttpsQuery.java»

Модуль содержит атрибуты и функции (табл. П4.52).

Таблица П4.52

## Спецификация модуля «HttpsQuery.java»

Название	Описание
Функции	
public static byte[] postQuery(String urlString,JSONObject data)	POST-запрос для отправки данных на сервер
public static byte[] postQuery(String urlString,JSONObject data,File file)	POST-запрос для отправки файла на сервер с указанием параметров
private static String prepareData(JSONObject data)	Подготовка отправляемых параметров POST-запроса на сервер
public static byte[] getUrlBytes(String urlSpec)	GET-запрос для отправки данных на сервер
public static String getUrlString(String urlSpec)	Метод для преобразования байтов, полученных от getUrlBytes строку

## 2 Спецификации на компонент «ServerPartApp»

Компонент содержит два модуля, которые реализуют пакеты серверной части приложения, описание которых представлено в п. 5.1, 5.10. Каждый модуль содержит файлы скриптов на языке PHP (см. п. 5.13, рис. 5.54).

Таблица П4.53

## Модули компонента «ServerPartApp»

Модуль	Описание
ServerAPI	Предоставляет набор скриптов, вызываемых клиентской частью приложения для добавления, получения, изменения, редактирования данных базы MySQL о заявках на помощь, предложениях помощи, пользователях системы
ServeClasses	Предоставляет набор классов для работы с таблицами БД MySQL (см. п. 5.11): Requests, Offers, Users

Описание файлов модуля ServerClasses представлено в табл. П4.54.

Таблица П4.54

## Файлы модуля «ServerClasses»

Модуль	Описание
api.users.php	Предназначен для управления пользователями: регистрация, авторизация, изменение имени, пароля, логина, загрузка, удаление, изменение фото
api.files.php	Предназначен для загрузки графических файлов на сервер
api.requests.php	Предназначен для работы с заявками: добавление, изменение статуса, получение списка актуальных заявок, получение изображения заявки
api.offers.php	Предназначен для работы с предложениями помощи: добавление, получение списка предложений к заявке

## Руководство пользователя

### 1 Общие сведения о программе

HelpOnRoad.apk – пакетный файл мобильного приложения «Помощь на дороге» для устройств на платформе Android с минимальной допустимой для установки приложения версией системы Android 4.1 (Jelly Bean).

Мобильное приложение «Помощь на дороге» представляет собой сервис взаимопомощи водителей на дороге. Приложение позволяет водителю, у которого произошла поломка (неисправность) дорожно-транспортного средства, попросить помочь у других участников дорожного движения, создав в приложении новую заявку на помощь, включающую краткое описание и фотографию возникшей проблемы и выбранное на карте Google местоположение. После добавления заявки другие пользователи приложения получат уведомление о новой заявке в системе и смогут вести переписку с пользователем, который просит помочь.

Пользователи приложения имеют возможность:

- регистрироваться и управлять учетной записью в приложении;
- настраивать детали получения заявок (радиус и периодичность получения новых заявок на помощь);
- получать уведомления о новых заявках в приложении;
- просматривать актуальные заявки на помощь в виде списка заявок и отметок заявок на карте Google;
- просматривать детали заявки и вести переписку с пользователем, который добавил заявку в систему;
- построить маршрут до местоположения пользователя, который добавил заявку в систему;
- добавлять новую заявку на помощь в приложение;
- управлять статусом добавленной заявки.

## 2 Описание установки

Установка приложения возможна на мобильное устройство на платформе Android с минимальной допустимой для установки версией системы Android 4.1 (Jelly Bean). Устройство должно поддерживать: Wi-Fi, 2G/3G/4G - модуль. На устройстве должно быть установлено приложение Google Play Services (Сервисы Google Play) для определения местоположения, доступа к карте Google, получения уведомлений.

Перед установкой приложения необходимо вставить диск, который прилагается к данной работе, в дисковод персонального компьютера, подключить мобильное устройство к компьютеру через usb-кабель в режиме usb-накопителя и скопировать файл приложения HelpOnRoad.apk на мобильное устройство.

Для запуска процесса установки необходимо на мобильном устройстве запустить установочный файл приложения HelpOnRoad.apk. В случае успешного запуска на экране устройства будет отображено окно установки приложения, в котором для запуска процесса установки необходимо нажать на кнопку «Установить» (рис. П5.1).

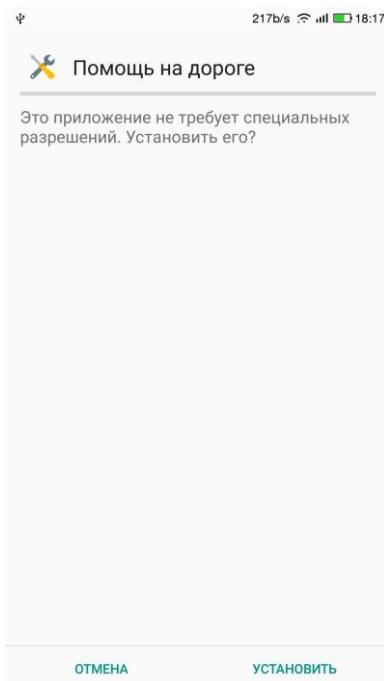


Рис. П5.1. Окно установки приложения

Начнется процесс установки приложения (рис. П5.2).

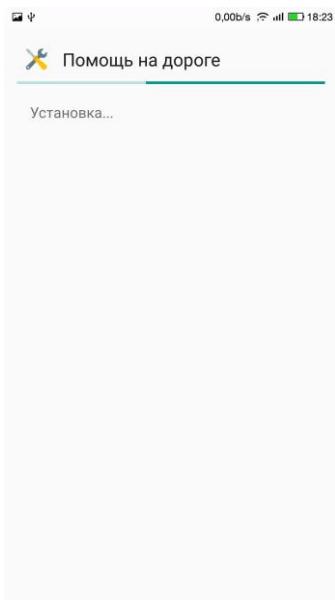


Рис. П5.2. Установка приложения на устройство

В случае успешной установки будет отображено окно о завершении процесса установки (рис. П5.3), в котором для подтверждения необходимо нажать на кнопку «Готово», для запуска приложения – кнопку «Открыть».

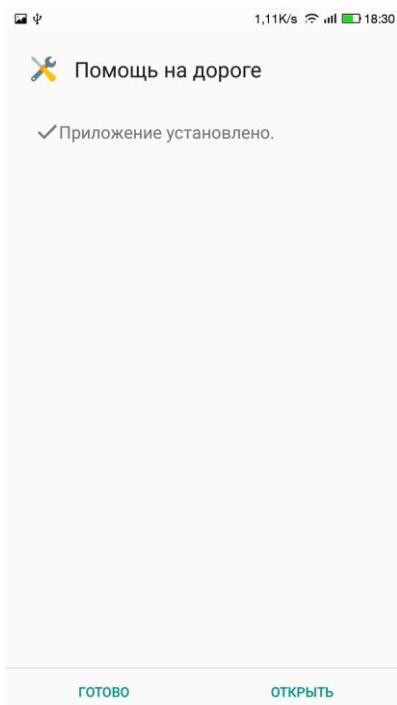


Рис. П5.3. Окно с сообщением об успешной установке приложения на устройство

Примечание: если в процессе установки приложения на устройство получено сообщение о запрете установки приложений из неизвестных источников, то необходимо в настройках устройства разрешить установку приложений из неизвестных источников и повторить процесс установки приложения.

### 3 Описание запуска

Для работы с приложением необходимо в списке приложений устройства запустить приложение «Помощь на дороге». В случае успешного запуска на экране устройства будет отображено главное окно приложения – «Заявки в системе» (рис. П5.4).

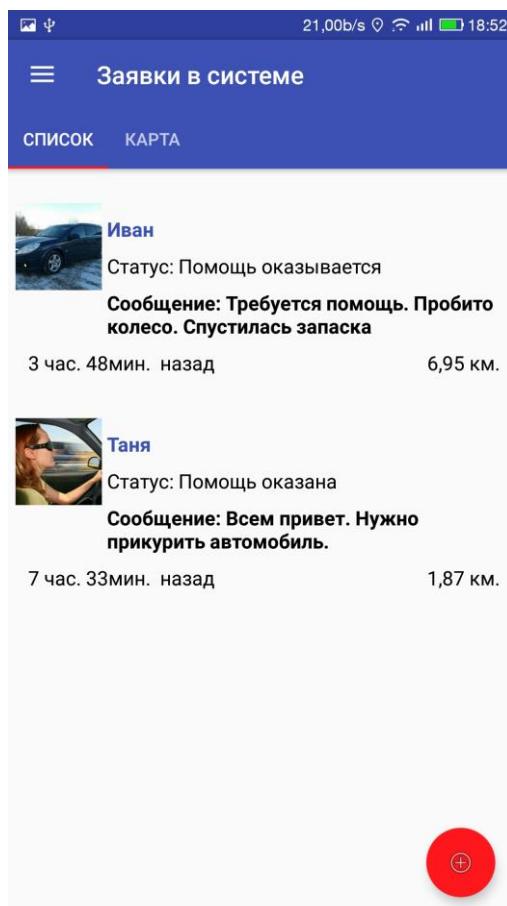


Рис. П5.4. Главное окно приложения – «Заявки в системе»

Примечание: При первом запуске приложения будет получено сообщение о том, что приложению требуется разрешение для определения местоположения

устройство. Необходимо в диалоговом окне нажать на кнопку «Разрешить» для разрешения определения местоположения устройством.

## 4 Работа с приложением

### 4.1 Главное меню приложения

Главное меню приложения запускается правым свайпом (специальный жест, когда пользователь кладет палец на экран смартфона и ведет в нужном направлении) в окне приложения - «Заявки в системе» (рис. П5.4).

Для неавторизованного пользователя приложения главное меню содержит пункты (рис. П5.5): «Заявки в системе» - для просмотра актуальных заявок на помощь (см. п. 4.6), «Войти» - для прохождения процесса авторизации (см. п. 4.3) или регистрации (см. п. 4.2) в приложении, «Настройки» - для изменения настроек приложения (см. п. 4.8).

Для авторизованного пользователя приложения главное меню содержит пункты (рис. П5.6): «Заявки в системе», «Мои заявки» - для просмотра заявок на помощь, которые добавлены пользователем приложения (см. п. 4.7), «Учетная запись» - для изменения учетной записи пользователя (см. п. 4.4), «Настройки».

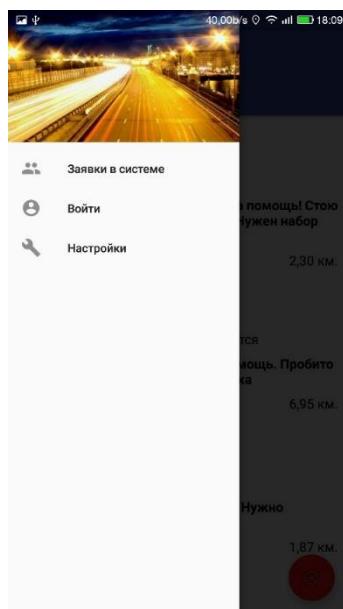


Рис. П5.5. Главное меню для неавторизованного пользователя приложения

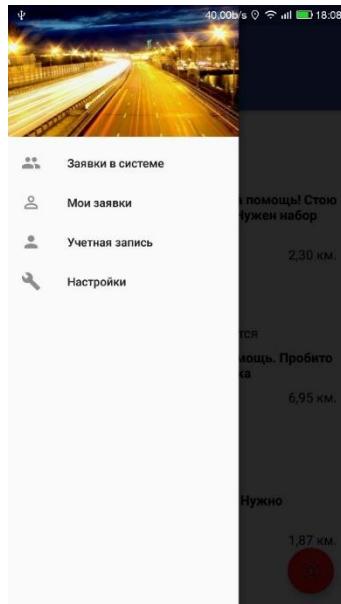


Рис. П5.6. Главное меню для авторизованного пользователя приложения

## 4.2 Регистрация в приложении

Для прохождения процедуры регистрации пользователю необходимо в главном меню (рис. П5.5) выбрать пункт «Войти». Откроется окно для авторизации в приложении (рис. П5.7), в котором для перехода к окну для регистрации необходимо нажать на кнопку «Зарегистрироваться» (рис. П5.8).

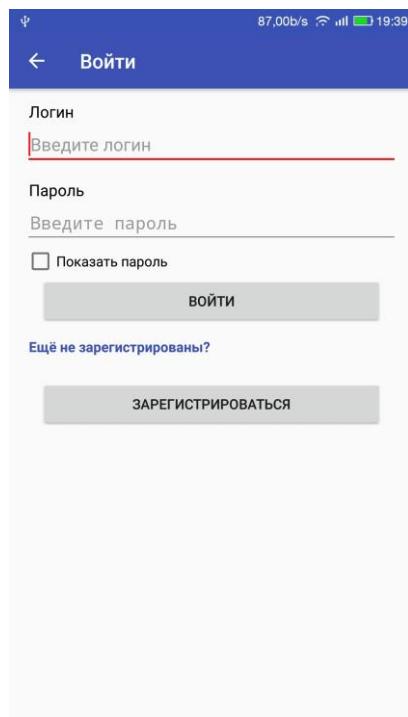


Рис. П5.7. Окно для авторизации в приложении

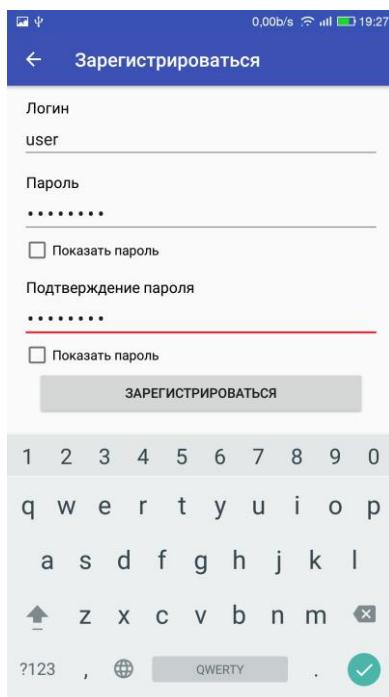


Рис. П5.8. Окно для регистрации в приложении

Затем в окне регистрации (рис. П5.8) необходимо в поле «Логин» ввести логин, который должен содержать не менее 4 символов и быть уникальным в системе, в поле «Пароль» ввести пароль, в поле «Подтверждение пароля» необходимо повторить введенный пароль. Для подтверждения процедуры регистрации необходимо нажать на кнопку «Зарегистрироваться».

Если регистрация нового пользователя прошла успешно, то будет получено сообщение «Пользователь зарегистрирован успешно».

Возможны следующие сообщения об ошибках, если пользователю не удалось пройти регистрацию в приложении:

- «Отсутствует подключение к сети интернет». В данном случае необходимо проверить подключение к сети интернет и повторно нажать на кнопку «Зарегистрироваться»;
- «Пользователем с данным логином уже зарегистрирован» - если введенный пользователем логин уже используется в системе;
- «Логин/Пароль должен содержать не менее 4 символов» - если введенный пользователем логин/пароль меньше 4 символов;

- «Значения полей “Пароль”, “Подтверждение пароля не совпадают”» - если значения указанных полей не совпадают.

При получении сообщений о неправильно введенных данных необходимо проверить введенные данные и повторно нажать на кнопку «Зарегистрироваться».

#### 4.3 Авторизация в приложении

Для прохождения процедуры авторизации необходимо в главном меню (рис. П5.5) выбрать пункт «Войти». Откроется окно для авторизации в приложении (рис. П5.9), в котором необходимо ввести логин и пароль пользователя приложения. Для подтверждения авторизации необходимо нажать на кнопку «Войти». Если авторизация прошла успешно, то будет получено сообщение «Вы вошли в приложение».

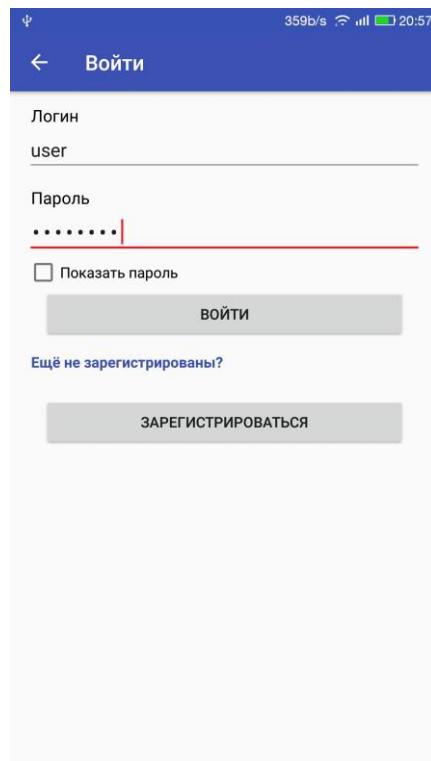


Рис. П5.9. Окно для авторизации в приложении

В случае неудачной авторизации возможны следующие сообщения об ошибках:

- «Отсутствует подключение к сети интернет» - если отсутствует подключение к сети интернет. В данном случае необходимо проверить подключение к сети интернет и повторно нажать на кнопку «Войти»;
- «Указаны неверные значения логина и пароля» - если пользователь с введенными значениями логина и пароля не зарегистрирован в приложении. В данном случае необходимо проверить введенные значения логина и пароля и повторно нажать на кнопку «Войти».

#### 4.4 Изменение учетной записи

Каждый авторизованный пользователь может изменять настройки учетной записи, под которой он авторизовался в приложении (см. п. 4.3).

Для перехода к настройкам учетной записи необходимо в главном меню выбрать пункт «Учетная запись» (рис. П5.6). Откроется окно для просмотра сведений учетной записи (рис. П5.10), в котором отображаются настройки учетной записи пользователя: имя, логин, фото пользователя.

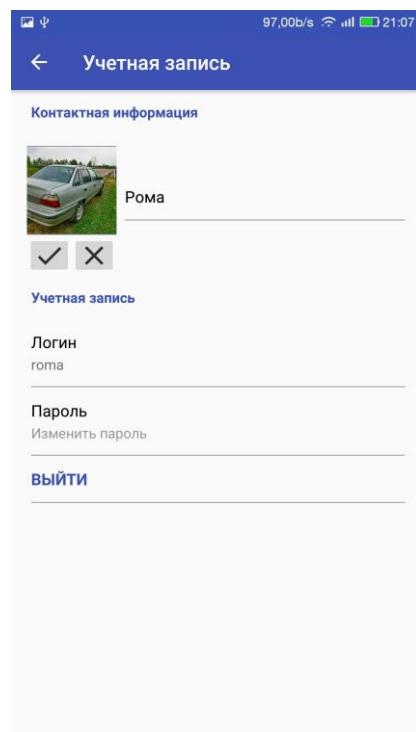


Рис. П5.10. Окно для просмотра сведений учетной записи пользователя

В данном окне пользователь может изменить имя, пароль, фотографию пользователя или выйти из учетной записи.

#### 4.4.1 Изменение имени пользователя

Для изменения имени пользователя необходимо нажать на текстовое поле с именем пользователя (рис. П5.10), откроется диалоговое окно для изменения имени (рис. П5.11), в котором необходимо ввести новое значение имени и нажать на кнопку «OK» для подтверждения операции или нажать на кнопку «Отмена» для отмены операции. В случае успешного изменения имени пользователя будет получено сообщение «Имя пользователя изменено успешно».

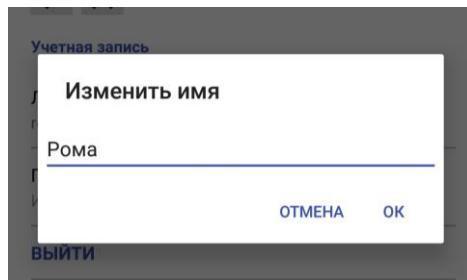


Рис. П5.11. Изменение имени пользователя

В случае, если не удалось изменить имя пользователя, будет получено сообщение «При изменении имени пользователя произошла ошибка». В данном случае необходимо проверить введенное значение имени пользователя и повторить попытку.

При отсутствии доступа к сети интернет будет получено сообщение «Отсутствует подключение к сети интернет», в данном случае необходимо проверить подключение к сети интернет и повторить попытку.

#### 4.4.2 Изменение пароля пользователя

Для изменения пароля пользователя необходимо нажать на текстовое поле «Изменить пароль» (рис. П5.10), откроется диалоговое окно для изменения пароля (рис. П5.12).

В диалоговом окне необходимо ввести значение пароля в тестовом поле «Введите пароль», повторить введенное значение в текстовом поле «Подтверждение пароля» и нажать на кнопку «Ок» для подтверждения операции или нажать на кнопку «Отмена» для отмены операции.

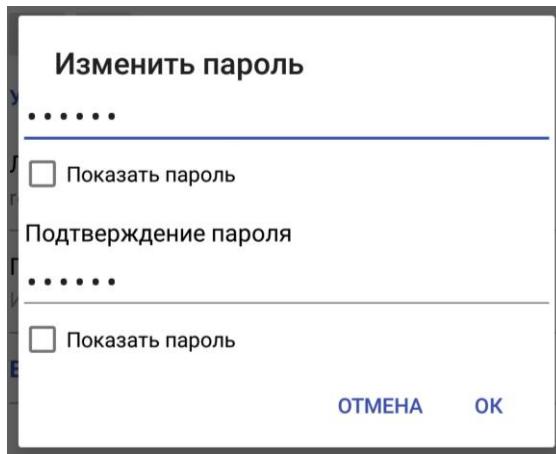


Рис. П5.12. Изменение пароля пользователя

В случае успешного изменения пароля пользователя будет получено сообщение «Пароль пользователя изменен успешно».

В случае отсутствия подключения к сети интернет будет получено соответствующее сообщение. В данном случае необходимо проверить подключение к сети интернет и повторить изменение пароля.

Если пароль не удалось изменить, то могут быть получены следующие сообщения об ошибках:

- «Пароль должен содержать не менее 4 символов» - если введенный пользователем пароль меньше 4 символов;
- «Значения полей “Пароль”, “Подтверждение пароля не совпадают”» - если значения указанных полей не совпадают.

В случае получения указанных сообщений необходимо проверить введенные данные и повторить попытку.

#### 4.4.3 Загрузка нового изображения пользователя

Для загрузки нового изображения пользователя необходимо в окне учетной записи нажать на изображение пользователя (рис. П5.10) и в открывшемся диалоговом окне (рис. П5.13) выбрать способ добавления изображения: с камеры («Снять на камеру») или из галереи устройства («Из альбома»).

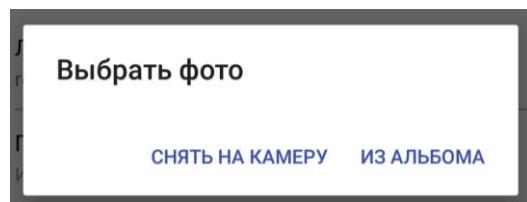


Рис. П5.13. Диалоговое окно для выбора способа добавления изображения

##### 4.4.3.1 Выбор изображения из галереи устройства

При выборе пункта «Из альбома» откроется окно галереи устройства, из которой необходимо выбрать изображение, нажав на картинку (рис. П5.14). В левом верхнем углу галереи имеется кнопка для отмены выбора изображения.

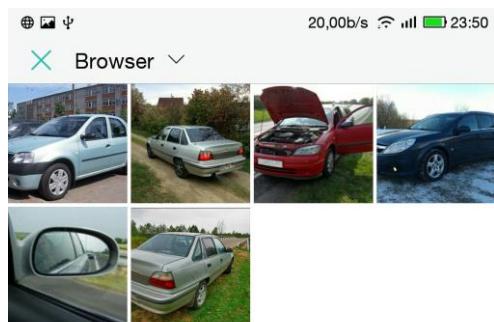


Рис. 5.14. Выбор изображения из галереи устройства

Если изображение было выбрано, то оно установится в качестве фотографии пользователя, и откроется диалоговое окно для выбора программы обрезки изображения (рис. П5.15).

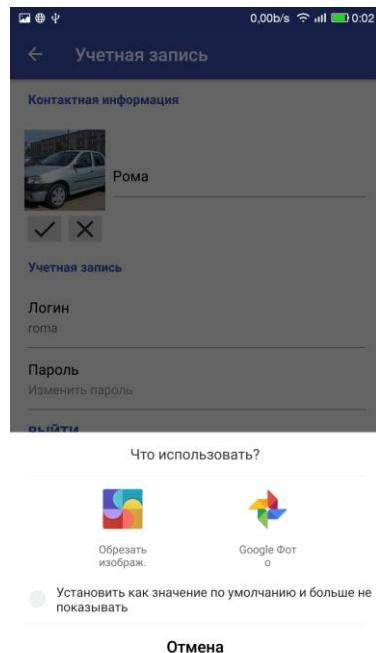


Рис. П5.15. Диалог выбора программы для обрезки изображения

### Обрезка изображения

При выборе пункта «Обрезать изображение» в диалоговом окне (рис. П5.15, нижняя часть) запустится программа системы Android, позволяющая обрезать изображение (рис. П5.16). При выборе в диалоговом окне пункта «Отмена» обрезка изображения выполнится не будет.

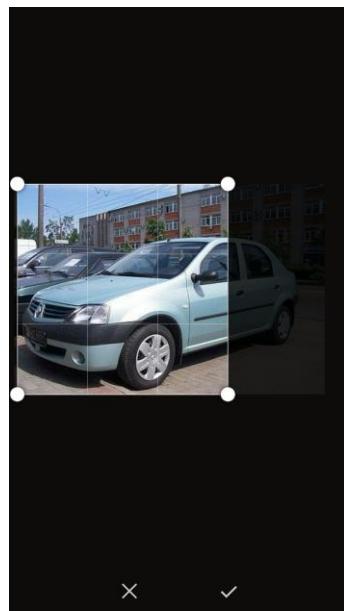


Рис. П5.16. Обрезка изображения

Для обрезки изображения (рис. П5.16) необходимо выделить обрезаемую область изображения и нажать на «галочку» для подтверждения операции, для отмены операции обрезки необходимо нажать на «крестик».

Примечание 1: если на устройстве нет программы, позволяющей обрезать изображения, то будет получено сообщение «Ваше устройство не поддерживает операцию кадрирования».

Примечание 2: в зависимости от модели устройства внешний вид окна галереи устройства, диалоговых окон может отличаться от окон, изображенных на рис. П5.14 – П5.16).

#### 4.4.3.2 Съемка изображения на камеру

При выборе пункта «Из альбома» откроется стандартное окно камеры устройства, в котором необходимо выполнить съемку изображения (рис. П5.17).

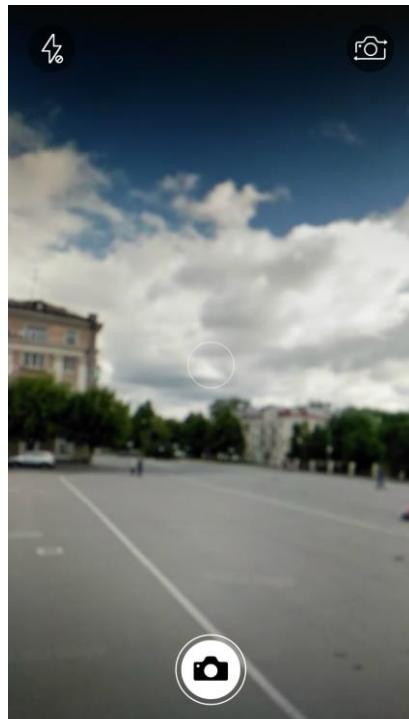


Рис. П5.17. Съемка изображения на камеру

После снятия изображения необходимо подтвердить установку изображения в качестве фотографии пользователя, нажав на «галочку», или отменить, нажав на «крестик» (рис. П5.18).

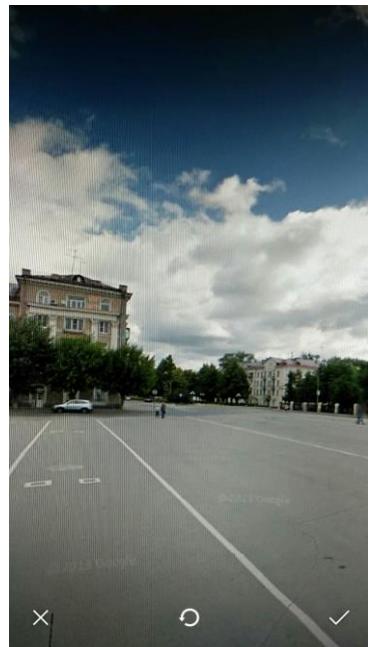


Рис. П5.18. Подтверждение операции съемки

После подтверждения установки снятого на камеру изображения, оно будет установлено в качестве изображения пользователя, и откроется диалоговое окно для выбора программы обрезки изображения (см. п. 4.4.3.1, «Обрезка изображения»).

Примечание: в зависимости от модели устройства внешний вид окон для работы с камерой может отличаться от окон, представленных на рис. П5.17 – П5.18.

#### 4.4.3.3 Сохранение изображения

После выбора изображения пользователя из галереи или съемки на камеру устройства новое изображение (рис. П5.19) отобразится в окне учетной записи пользователя.

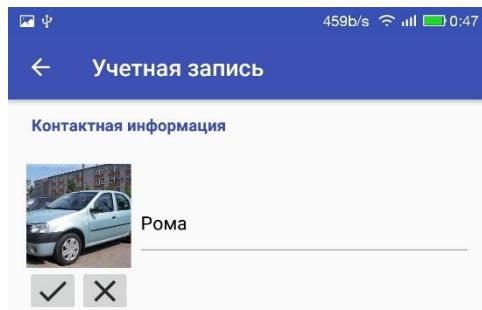


Рис. П5.19. Новое изображение пользователя

Для сохранения выбранного изображения необходимо нажать на кнопку  . Если сохранение изображения выполнено успешно, то будет получено сообщение «Изображение пользователя изменено успешно».

Если отсутствует подключение к сети интернет, то будет получено сообщение о необходимости проверки подключения и повтора операции сохранения.

Если получено сообщение «Не выбрано изображение пользователя», то необходимо выбрать изображение (см. п. 4.4.3.1 - 4.4.3.2).

#### 4.4.4 Удаление изображения пользователя

Для удаления изображения пользователя необходимо нажать на кнопку  . Если удаление изображения выполнено успешно (рис. П5.20), то будет получено сообщение «Изображение пользователя удалено успешно».

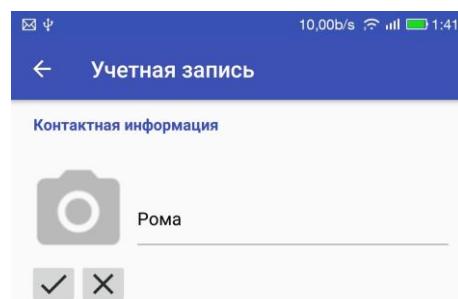


Рис. П5.20. Результат удаления изображения

Если отсутствует подключение к сети интернет, то будет получено сообщение о необходимости проверки подключения и повтора операции сохранения.

Если у пользователя не задано изображение, то будет получено сообщение «Не задано изображение пользователя».

#### 4.4.5 Выход из учетной записи

Для выхода из учетной записи необходимо нажать «ВЫЙТИ» в окне учетной записи (рис. П5.10) и в открывшемся диалоговом окне (рис. П5.21) нажать на «OK».



Рис. П5.21. Диалоговое окно для выхода из учетной записи

После выхода из учетной записи будет получено сообщение «Вы вышли из учетной записи».

#### 4.5 Создание новой заявки на помощь

Для добавления новой заявки пользователю приложения необходимо авторизоваться или зарегистрироваться в приложении (см. п. 4.1. «Регистрация в приложении», п. 4.2. «Авторизация в приложении»).

Для добавления новой заявки на помощь необходимо в окне приложения «Заявки в системе» (рис. П5.4) нажать на кнопку «+», расположенную в нижнем правом углу окна. В открывшемся окне для добавления новой заявки (рис. П5.22) нужно ввести текст сообщения заявки (обязательное для заполнения поле).

Если требуется выбрать фотографию заявки (рис. П5.22), то нужно нажать на кнопку «Добавить» и выбрать изображение с камеры или галереи устройства

(Порядок действий при добавлении изображения к заявке аналогичный тому, который описан в п. 4.4.3 «Изменение изображения пользователя»).

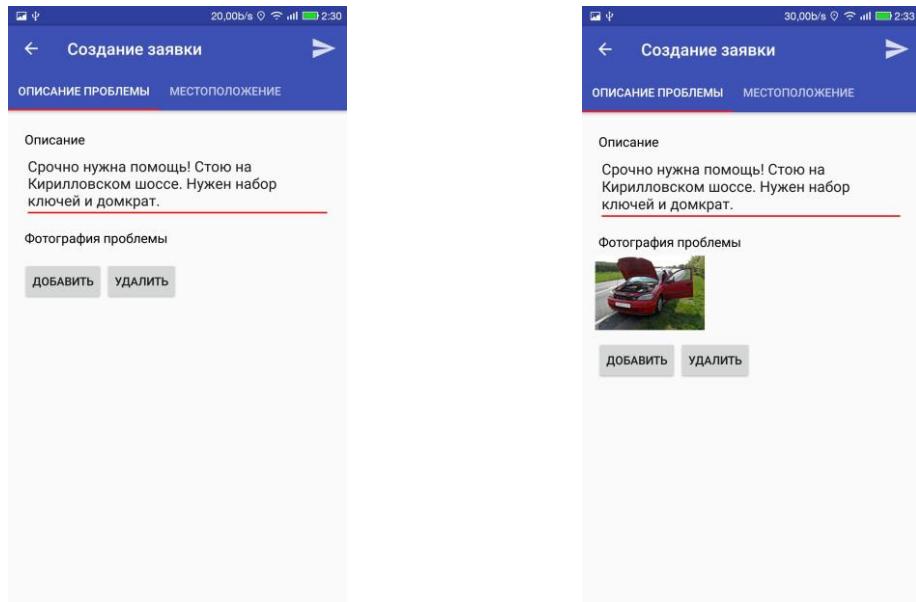


Рис. П5.22. Создание заявки: добавление сообщения и фотографии заявки

Для уточнения местоположения заявки необходимо перейти к вкладке «Местоположение». По умолчанию маркер показывает текущее местоположение пользователя, определенное устройством автоматически.

Для уточнения местоположения необходимо переместить маркер нажатием на нужное место (рис. П5.23).

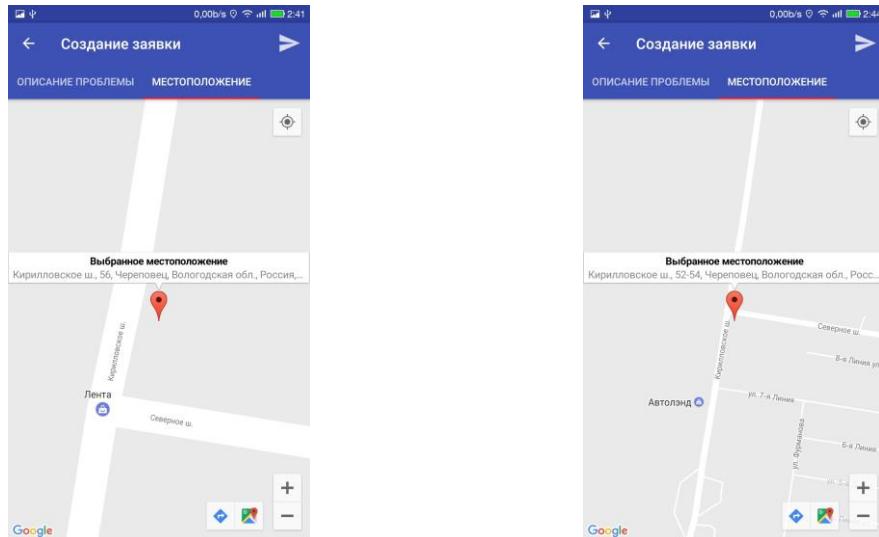


Рис. П5.23. Создание заявки: автоматически определенное и уточненное пользователем местоположение

Для сохранения заявки в приложении необходимо нажать на кнопку ➤. Если заявка добавлена успешно, то будет получено сообщение «Заявка отправлена успешно» и заявка появится в списке заявок (рис. П5.24).



Рис. П5.24. Просмотр добавленной заявки

Если отсутствует подключение к сети интернет, то будет получено сообщение о необходимости проверки подключения и повтора операции сохранения заявки.

Если не введено сообщение заявки, то будет получено сообщение «Не задано сообщение заявки». В этом случае необходимо ввести сообщение заявки и повторить сохранение заявки в приложении.

Если пользователь не авторизован в приложении, то откроется диалоговое окно, в котором пользователю будет предложено пройти авторизацию в приложении.

## 4.6 Просмотр заявок в системе

### 4.6.1 Просмотр списка заявок

Просмотр списка актуальных заявок осуществляется при запуске приложения в окне «Заявки в системе» (рис. П5.25).

Для перехода к окну «Заявки в системе» также можно выбрать пункт «Заявки в системе» в главном меню приложения (рис. П5.5).

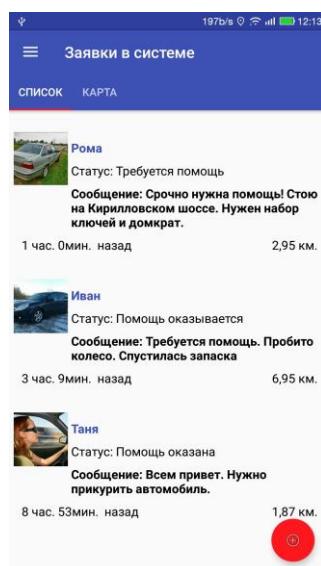


Рис. П5.25. Просмотр списка актуальных заявок на помощь

В списке отображаются заявки на помощь, которые добавлены в течение суток на данный момент и которые попадают в указанный в настройках приложения радиус получения заявок (см. п. 4.8). Для каждой заявки в списке отображается имя и фотография пользователя, который ее добавил, сообщение заявки, время добавления заявки (например, 8 час. 53 мин. назад), расстояние до пользователя, который добавил заявку в приложение (например, 1,87 км.) (рис. П5.26).

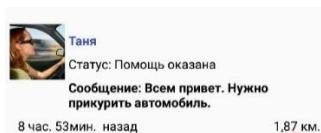


Рис. П5.26. Заявка

Также каждая заявка имеет статус: «требуется помощь», «помощь оказывается», «помощь оказана», который показывает получена ли пользователем заявки требуемая помощь.

Обновление списка заявок осуществляется свайпом вниз (рис. П5.27).

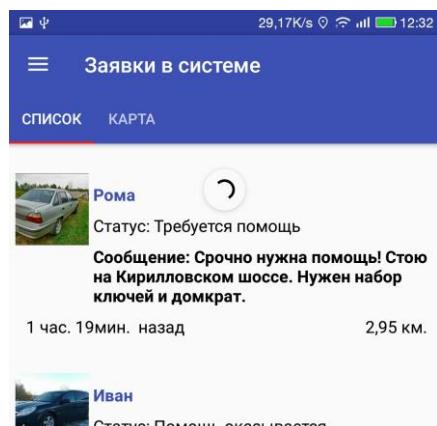


Рис. П5.27. Обновление списка заявок

В случае, если данный момент в приложении нет заявок, то на экране отобразится сообщение «В этой категории нет заявок»

Если при загрузке списка заявок произошла ошибка, то будет получено сообщение «При загрузке списка заявок произошла ошибка», при возникновении которой необходимо повторить обновление списка заявок (рис. П5.27).

Для просмотра деталей заявки (см. п. 4.6.3) необходимо выбрать соответствующий элемент списка нажатием на него.

#### 4.6.2 Просмотр карты заявок

Для просмотра отметок заявок на карте необходимо перейти на вкладку «Карта» окна «Список заявок». На карте отображаются маркеры заявок на помощь, которые добавлены в течение суток на данный момент и которые попадают в указанный в настройках приложения радиус получения заявок (см. п. 4.8). Каждый маркер имеет цвет в зависимости от статуса заявки и показывает местоположение заявки:

- желтым маркером показываются заявки со статусом «требуется помощь»;

- синим маркером показываются заявки со статусом «помощь оказывается»;
- зеленым маркером показываются заявки со статусом «помощь оказана».

Иключение составляет красный маркер, который показывает текущее местоположение пользователя.

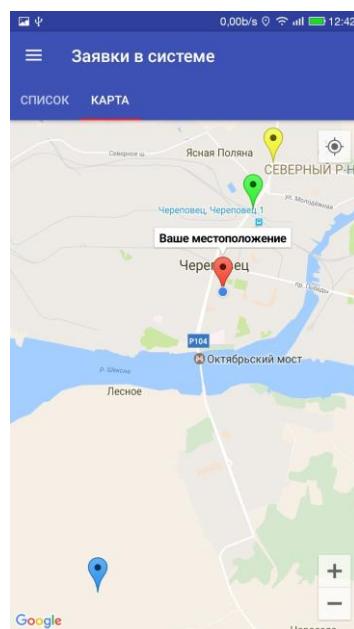


Рис. П5.28. Карта заявок

При нажатии на маркер открывается диалоговое окно, в котором при нажатии: на кнопку «OK» осуществляется переход к деталям заявки (см. п 4.6.3), на кнопку «ОТМЕНА» - отмена указанной операции.

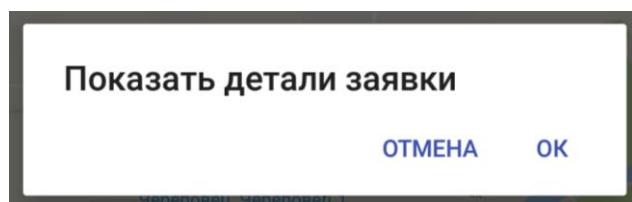


Рис. П5.29. Просмотр деталей заявки

#### 4.6.3 Просмотр деталей заявки

В данном режиме показываются детали выбранной в п. 4.6.1, 4.6.2 заявки.

#### 4.6.3.1 Просмотр описания заявки

Вкладка окна «Детали заявки» содержит описание заявки: имя, фотографию, пользователя, который добавил заявку, сообщение и фотографию заявки, сведения о дате добавления и расстоянии до пользователя, который добавил заявку (рис. П5.30).



Рис. П5.30. Просмотр деталей заявки

После описания заявки содержится список сообщений пользователей, который показывает переписку пользователей с водителем, которому требуется помощь. Список автоматически обновляется при появлении новых сообщений для данной заявки.

Для добавления нового сообщения пользователю необходимо быть авторизованным в приложении и ввести текст в текстовое поле «Введите текст сообщения...» внизу окна и нажать на кнопку кнопку ► для отправки

сообщения. В момент добавления сообщения выполняется обновление списка сообщений (рис. П5.31).

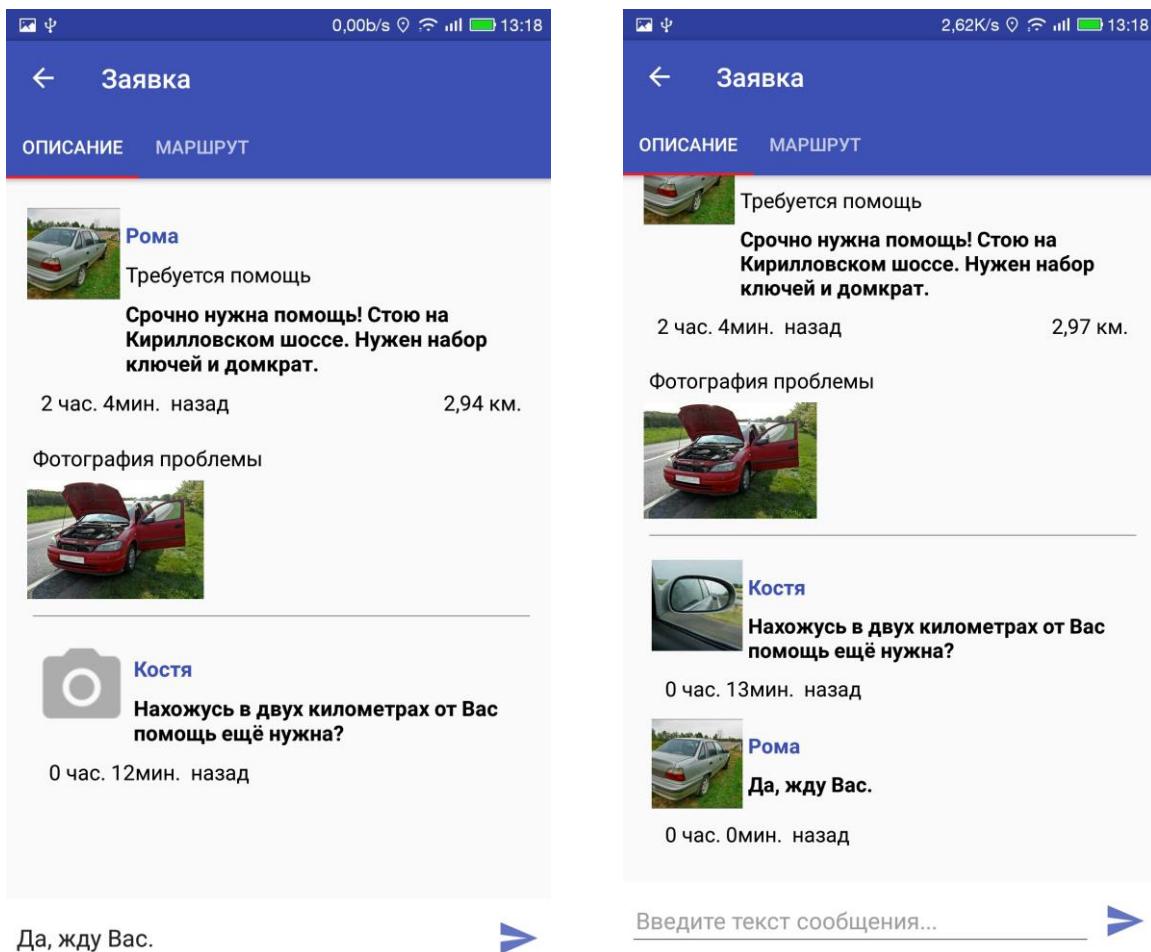


Рис. П5.31. Добавление сообщения к заявке

#### 4.6.3.2 Просмотр маршрута заявки

При переходе на вкладку «Маршрут» окна «Детали заявки» показывается маршрут между местоположением пользователя приложения (отображается красным маркером) и местоположением пользователя заявки. Для построения маршрута необходимо, чтобы мобильное устройство было подключено к сети интернет (рис. П5.32).

Построенный маршрут является приближенным и при больших расстояниях возможны отклонения от реального маршрута.

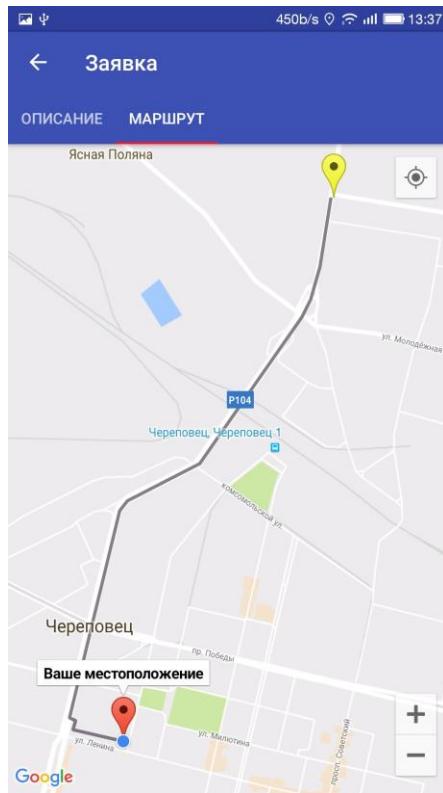


Рис. П5.32. Маршрут заявки

При нажатии на маркер заявки отображаются сведения о пользователе и адресе местоположения заявки, на которое указывает данный маркер (рис. П5.33).

При нажатии на маркер местоположения пользователя отображаются сведения об адресе местоположения пользователя.

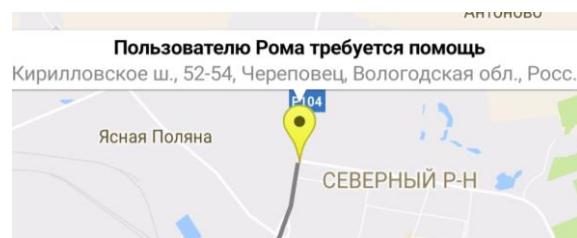


Рис. П5.33. Просмотр сведений маркера

В случае, если не удалось построить маршрут, пользователю выдается сообщение том, что не удается построить маршрут.

## 4.7 Просмотр заявок пользователя приложения

Если пользователь авторизован в приложении, то при нажатии в главном меню приложения (см. рис. П5.6) в окне «Заявки в системе» (см. п. 4.6) показываются только актуальные заявки (заявки, которые добавлены в течении суток на момент просмотра списка) авторизованного пользователя приложения (рис. П5.34).

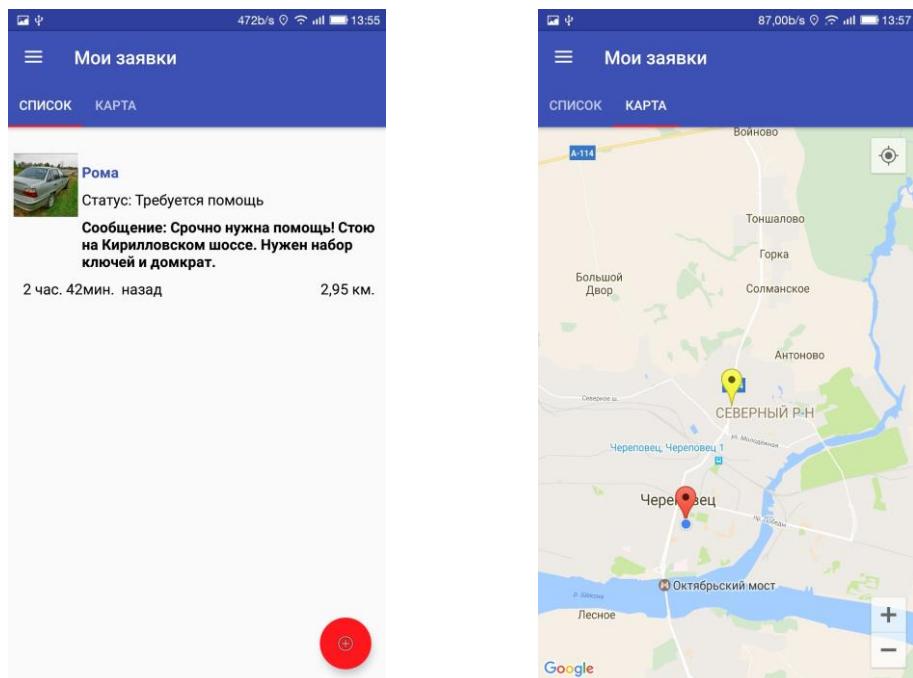


Рис. П5.34. Просмотр заявок авторизованного пользователя приложения

При нажатии на элемент списка или нажатии на маркер заявки на карте заявок осуществляется переход к сведениям детальной заявки (см. п. 4.6.3).

### 4.7.1 Изменение статуса заявки

В режиме просмотра заявок пользователя приложения пользователь при просмотре сведений детальной заявки может изменить статус заявки, выбрав в выпадающем списке новое значение статуса из значений (рис. П5.35):

- статус «требуется помощь» показывает, что пользователь заявки ждет помощь от других пользователей приложения;

- статус «оказывается помощь» показывает, что пользователь заявки уже получает помощь от других пользователей приложения;
- статус «помощь оказана» показывает, что пользователь заявки получил помощь и помощь ему больше не требуется.

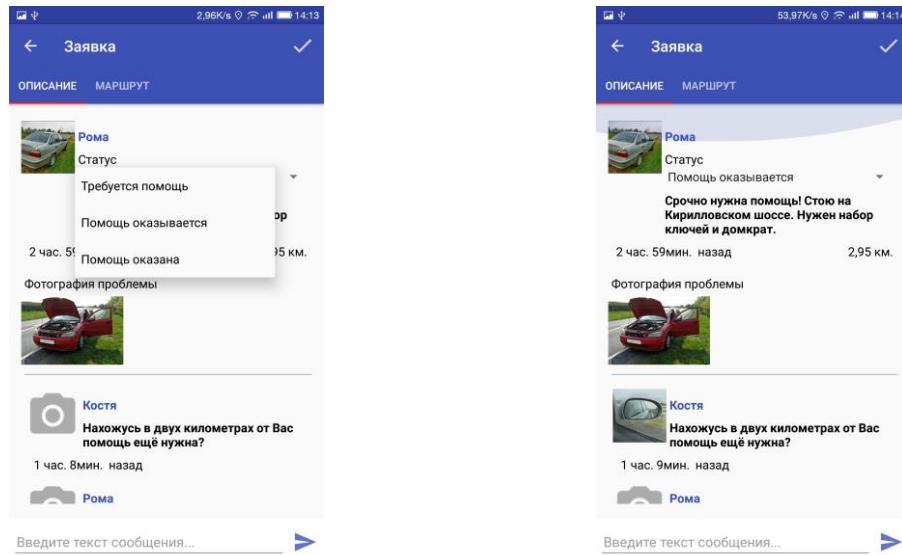


Рис. П5.35. Изменение статуса заявки

Для сохранения нового статуса необходимо нажать на  , расположенную на панели заголовка окна. Если статус изменен успешно, то будет получено сообщение об изменении заявки и обновленная заявка будет видна в списке заявок (рис. П5.36).

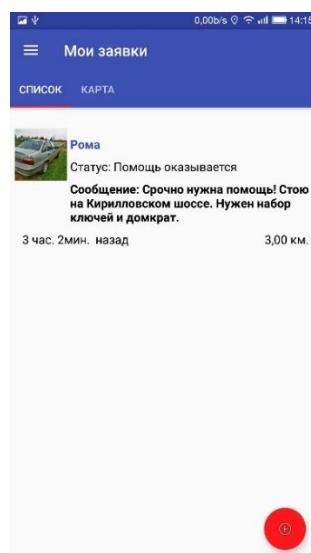


Рис. П5.36. Заявка пользователя с измененным статусом

## 4.8 Изменение настроек приложения

Для перехода к изменению настроек приложения необходимо в главном меню приложения (рис. П5.6) выбрать пункт «Настройки»: будет осуществлен переход к окну для изменения настроек приложения (рис. П5.37).

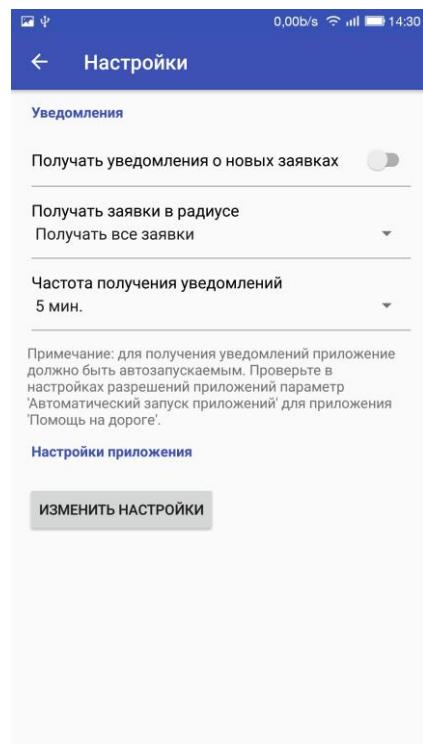


Рис. П5.37. Изменение настроек приложения

### 4.8.1 Задание параметров получения заявок и настроек уведомлений

В приложении доступна функция, которая позволяет получать всплывающие уведомления о новых заявках в приложении.

#### 4.8.1.1 Задание радиуса получения заявок

Для настройки получения уведомлений и фильтрации списка заявок необходимо задать радиус получения заявок, который показывает расстояние от пользователя до местоположения заявки, путем выбора его из значений выпадающего списка «Получать заявки в радиусе» (рис. П5.38). По умолчанию в приложении задано получение всех заявок.

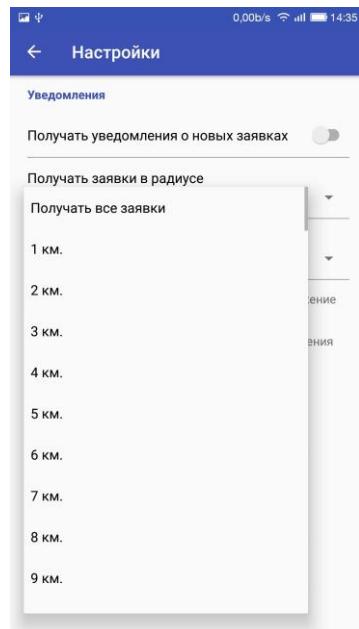


Рис. П5.38. Задание радиуса получения заявок

#### 4.8.1.2 Задание частоты получения уведомлений

Пользователь приложения имеет возможность задать частоту получения уведомлений, которая показывает, как часто выполняется проверка наличия новых заявок в приложении. По умолчанию частота получения уведомлений равна 5 мин. Пользователь может задать частоту получения заявок в пределах от 1 до 60 мин. (см. рис. П5.39).

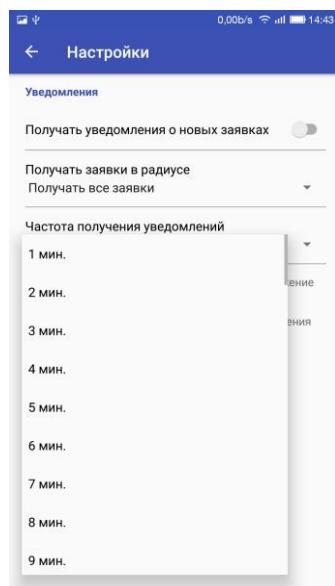


Рис. П5.39. Задание частоты получения уведомлений

## 4.8.2 Получение уведомлений о новых заявках

### 4.8.2.1 Проверка, является ли приложение автозапускаемым

Для того, чтобы уведомления о новых заявках приходили, когда приложение не является запущенным, необходимо проверить является ли приложение автозапускаемым. Это необходимо для запуска фонового сервиса, который проверяет наличие новых заявок и посыпает уведомления о них пользователю устройства.

Для этого нужно в стандартном приложении системы Android «Настройки» найти пункт «Разрешения», в котором необходимо разрешить автоматический запуск для приложения «Помощь на дороге». Пример установки данного параметра представлен на рис. П5.40.

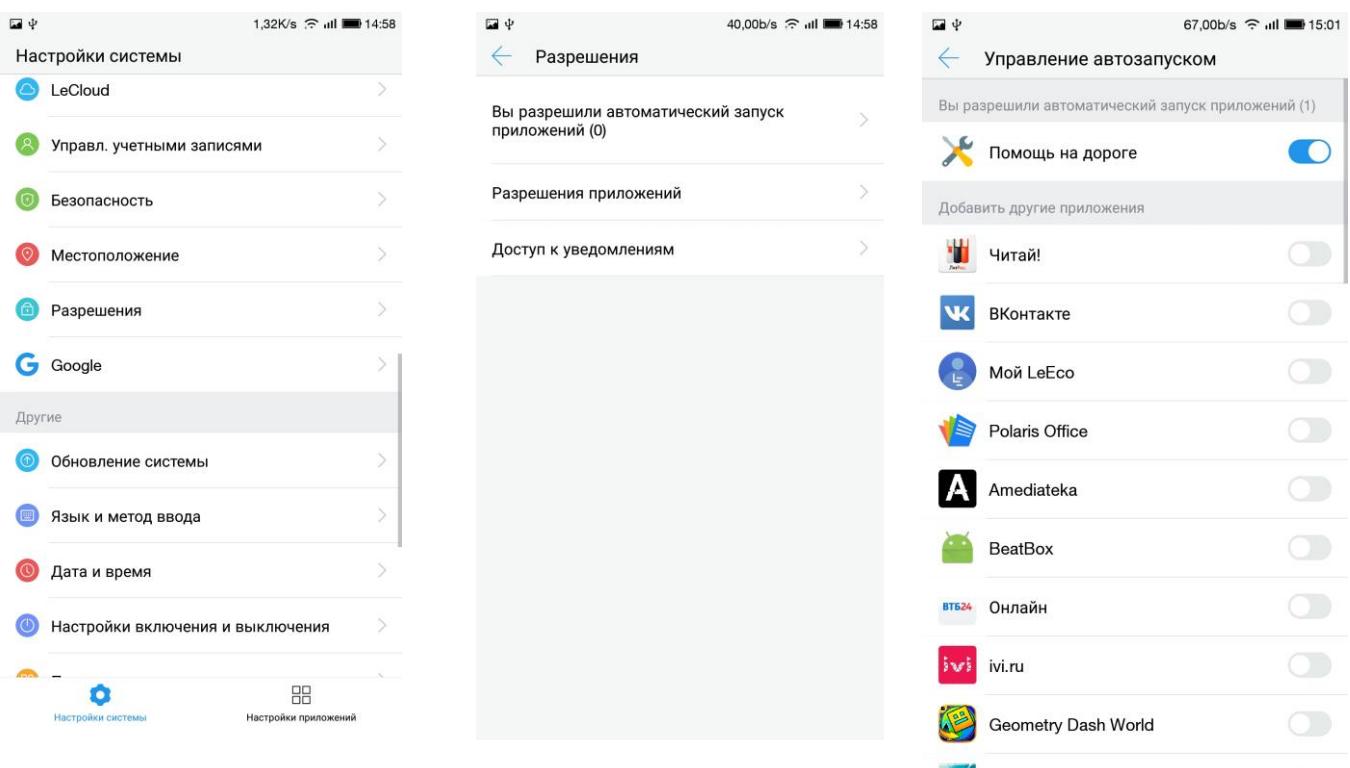


Рис. П5.40. Пример установки автозапуска для приложения «Помощь на дороге»

Примечание: в зависимости от модели устройства и версии системы Android установка данного параметра может отличаться от примера, представленного на

рис. П5.40. Для некоторых моделей устройств установка данного параметра не требуется.

#### 4.8.2.2 Включение и отключение уведомлений

Для включения/отключения получения уведомлений о новых заявках необходимо установить переключатель «Получать уведомления» о новых заявках во включенное/выключенное состояние, соответственно (рис. П5.41).

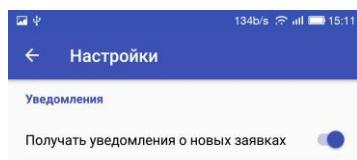


Рис. П5.41. Включение получения уведомлений о новых заявках

После включения уведомлений будут приходить всплывающие уведомления о новых заявках в системе со статусами: «Требуется помочь», «Помощь оказывается», которые еще не были просмотрены пользователем приложения и которые не добавлены авторизованным пользователем приложения. При получении уведомления устройством будет издан сигнал, и уведомление будет отображено на экране блокировки приложения (рис. П5.42).

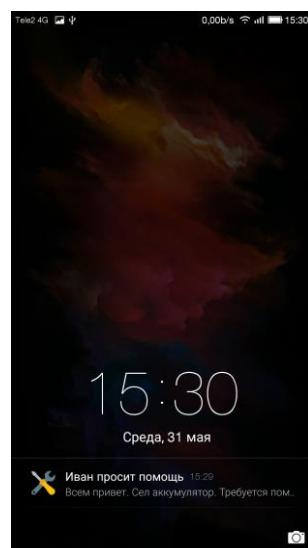


Рис. П5.42. Уведомление о новой заявке

При нажатии на уведомление будет запущено приложение для просмотра списка уведомлений (рис. П5.43).



Рис. П5.43. Запуск приложения из уведомления

#### 4.8.3 Изменение настроек приложения

В окне «Настройки» (рис. П5.37) имеется кнопка «Изменить настройки», при нажатии на которую запускается системное окно настроек (рис. П5.44). Внешний вид и набор пунктов которого зависит от версии системы Android и модели устройства.

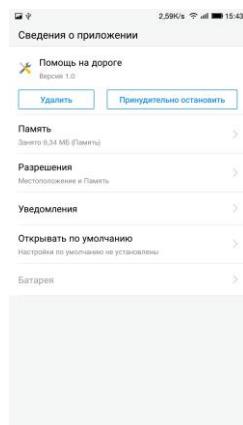


Рис. П5.44. Системное меню настроек приложения

Пункт «Разрешения» системных настроек приложения позволяет управлять разрешениями приложения (рис. П5.45).

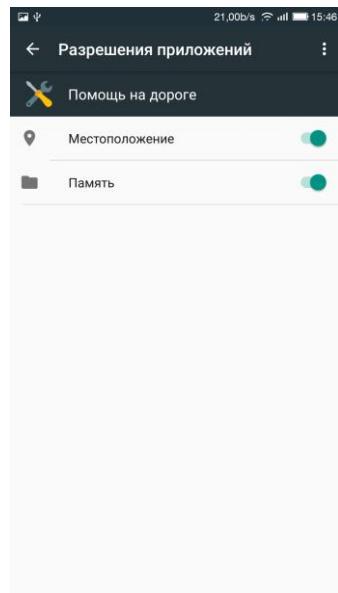


Рис. П5.45. Управление разрешениями приложения

Пункт «Уведомления» системных настроек приложения позволяет управлять уведомлениями, которые получает приложение (рис. П5.46).

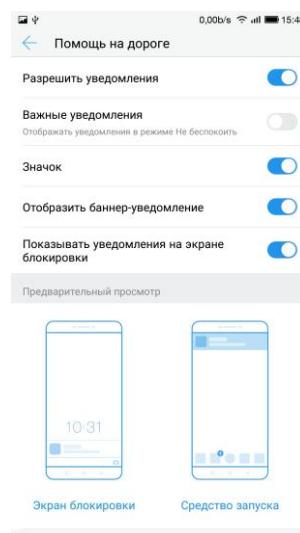


Рис. П5.46. Управление уведомлениями приложения

Также в окне настроек приложения (рис. П5.47) содержится кнопка «Удалить» для удаления приложения с устройства.

## Наборы тестовых данных и результатов тестирования

### 1 Наборы тестовых данных

#### 1.1 Тестовые данные для проверки скрипта users.changeName.php

users.changeName.php - изменение имени пользователя. Параметры: name - значение имени пользователя, id\_user – идентификатор пользователя. Тестовые данные – табл. П6.1, П6.2.

Таблица П6.1

#### Описание тестовых данных скрипта users.changeName.php

Параметр	Допустимые значения	Тестовые данные	
		Правильные	Неправильные
id_user	Имеющиеся значения поля id –идентификатор пользователя в таблице User БД MySQL	Наименьшее существующее значение id, наибольшее существующее значение id	Наименьшее существующее значение id, уменьшенное на 1, наибольшее существующее значение id, увеличенное на 1
name	Строка текста длиной от 1 до 50 символов	Строка текста длиной 1 символ, строка текста длиной 50 символов	Пустая строка текста, строка текста длиной 51 символ

Таблица П6.2

#### Пример тестовых данных скрипта users.changeName.php

Параметр	Тестовые данные		Дата составления
	Правильные	Неправильные	29.04.2017 18:34
id_user	1,22	0, 23	
name	Строка текста: а Строка текста: a....a (50 символов)	Строка текста: пустая строка Строка текста: a.....a (51 символ)	

#### 1.2 Тестовые данные для проверки скрипта users.changeName.php

users.changePassword.php - изменение пароля пользователя. Параметры: name - значение имени пользователя, id\_user – идентификатор пользователя. Тестовые данные – табл. П6.3, П6.4.

Таблица П6.3

## Описание тестовых данных скрипта users.changeName.php

Параметр	Допустимые значения	Тестовые данные	
		Правильные	Неправильные
id_user	Имеющиеся значения поля id – идентификатор пользователя в таблице User БД MySQL	Наименьшее существующее значение id, наибольшее существующее значение id	Наименьшее существующее значение id, уменьшенное на 1, наибольшее существующее значение id, увеличенное на 1
name	Строка текста длиной от 4 до 50 символов	Строка текста длиной 4 символа, строка текста длиной 50 символов	Строка текста длиной 3 символа, строка текста длиной 51 символ

Таблица П6.4

## Пример тестовых данных скрипта users.changeName.php

Параметр	Тестовые данные		Дата составления
	Правильные	Неправильные	
id_user	1,22	0, 23	29.04.2017 18:36
password	Строка текста: aaaa Строка текста: a....a (50 символов)	Строка текста: aaa Строка текста: a.....a (51 символ)	

## 1.3 Тестовые данные для проверки скрипта users.loadUserImage.php

users.loadUserImage.php – изменение изображения пользователя. Параметры: id\_user – идентификатор пользователя. Тестовые данные – табл. П6.5, П6.6.

Таблица П6.5

## Описание тестовых данных скрипта users.loadUserImage.php

Параметр	Допустимые значения	Тестовые данные	
		Правильные	Неправильные
id_user	Имеющиеся значения поля id – идентификатор пользователя	Наименьшее значение id, наибольшее существующее значение id	Наименьшее значение id, уменьшенное на 1, наибольшее значение id+1

Таблица П6.6

## Пример тестовых данных скрипта users.loadUserImage.php

Параметр	Тестовые данные		Дата составления
	Правильные	Неправильные	
id_user	1,22	0, 23	29.04.2017 18:37

## 1.4 Тестовые данные для проверки работы скрипта users.changeImage.php

users.changeImage.php – изменение изображения пользователя. Параметры: id\_user – идентификатор пользователя. Тестовые данные – табл. П6.7, П6.8.

Таблица П6.7

### Описание тестовых данных скрипта users.changeImage.php

Параметр	Допустимые значения	Тестовые данные	
		Правильные	Неправильные
id_user	Имеющиеся значения поля id –идентификатор пользователя	Наименьшее значение id, наибольшее значение id	Наименьшее значение id, уменьшенное на 1, наибольшее значение id, увеличенное на 1

Таблица П6.8

### Пример тестовых данных скрипта users.changeImage.php

Параметр	Тестовые данные		Дата составления
	Правильные	Неправильные	29.04.2017 18:37
id_user	1,22	0, 23	

## 1.5 Тестовые данные для проверки работы скрипта users.deleteImage.php

users.deleteImage.php – удаление изображения пользователя. Параметры: id\_user – идентификатор пользователя. Тестовые данные – табл. П6.9, П6.10.

Таблица П6.9

### Описание тестовых данных скрипта users.deleteImage.php

Параметр	Допустимые значения	Тестовые данные	
		Правильные	Неправильные
id_user	Имеющиеся значения поля id –идентификатор пользователя	Наименьшее значение id, наибольшее значение id	Наименьшее значение id, уменьшенное на 1, наибольшее значение id, увеличенное на 1

Таблица П6.10

### Пример тестовых данных скрипта users.deleteImage.php

Параметр	Тестовые данные		Дата составления
	Правильные	Неправильные	29.04.2017 18:37
id_user	1,22	0, 23	

## 1.6 Тестовые данные для проверки скрипта request.updateRequestStatus.php

request.updateRequestStatus.php – изменение статуса заявки. Параметры: id\_request – идентификатор заявки. Тестовые данные – табл. П6.11, П6.12.

Таблица П6.11

### Описание тестовых данных для скрипта request.updateRequestStatus.php

Параметр	Допустимые значения	Тестовые данные	
	1	2	3
		Правильные	Неправильные
id_request	Имеющиеся значения поля id –идентификатор заявки в таблице Request	Наименьшее значение id, наибольшее значение id	Наименьшее значение id, уменьшенное на 1, наибольшее значение id, увеличенное на 1
id_status	0,1,2	0,2	-1,3

Таблица П6.12

### Пример тестовые данных для скрипта request.updateRequestStatus.php

Параметр	Тестовые данные		Дата составления
	Правильные	Неправильные	
id_request	5,50	4, 51	29.04.2017 18:38
id_status	0,2	-1,3	

## 1.7 Тестовые данные для проверки работы скрипта requests.getImage.php

Скрипт requests.getImage.php – получение изображения заявки. Параметры: id\_request – идентификатор заявки. Тестовые данные – табл. П6.13.

Таблица П6.13

### Описание тестовые данных скрипта requests.getImage.php

Параметр	Допустимые значения	Тестовые данные	
		Правильные	Неправильные
id_request	Имеющиеся значения поля id –идентификатор заявки в таблице User	Наименьшее значение id, наибольшее существующее значение id	Наименьшее значение id, уменьшенное на 1, наибольшее значение id, увеличенное на 1

## 2 Результаты тестирования

Таблица П6.14

### Тестирование модулей

Дата тестирования	Тестируемые модули	Способ тестирования	Описание теста	Тестолог	Результат тестирования
1	2		4	5	5
20.04.2017	LocationListenerGPSServices.java LocationPermission.java	Ручной	Проверка доступа к GPS сервису	разработчик	Приложение остановлено
20.04.2017	LocationListenerGPSServices.java LocationPermission.java	Ручной	Проверка доступа к GPS сервису	разработчик	Успех
21.04.2017	AddNewRequestMapFragment.java	Ручной	Отображение карты	разработчик	Успех
21.04.2017	LocationListenerGPSServices.java AddNewRequestMapFragment.java	Ручной	Отображение местоположения на карте	разработчик	Местоположение не отображается. Не происходит изменение окна карты
21.04.2017	LocationListenerGPSServices.java AddNewRequestMapFragment.java	Ручной	Изменение местоположения пользователем	разработчик	Не удается изменить местоположение, не отображается текстовое описание местоположения
22.04.2017	LocationListenerGPSServices.java AddNewRequestActivity.java AddNewRequestMapFragment.java	Ручной	Изменение местоположения на карте	Чистякова Ю.А.	Успех
22.04.2017	RequestManager.java AddNewRequestFragment.java AddNewRequestMapFragment.java	Ручной	Отправка заявки в базу	Аксенов А.С.	Заявка не отправляется серверной части
22.04.2017	AddNewRequestActivity.java AddNewMapFragment.java RequestManager.java	Ручной	Отправка заявки в базу	разработчик	Успех
22.04.2017	AddNewRequestFragment.java PhotoManager.java FileManager.java	Ручной	Добавление фотографии к заявке	Аксенов А.С.	Успех
24.04.2017	users.registerUser.java	Ручной	Проверка работы скрипта регистрации	разработчик	Успех

## Продолжение табл. П6.14

1	2		4	5	5
24.04.2017	UserManager.java RegistrationFragment.java	Ручной	Регистрация пользователя в приложении	Аксенов А.С.	Успех
25.04.2017	users.authorizeUser.java	Ручной	Проверка работы скрипта авторизации	разработчик	Успех
25.04.2017	UserManager.java EnterFragment.java	Ручной	Авторизация в приложении	Чистякова Ю.А.	Успех
26.04.2017	users.changeName.php	Функциональный	Изменение имени пользователя	разработчик	Успех
26.04.2017	UserManager.java UserAccountFragment.java SetUserNameDialog.java	Ручной	Изменение имени пользователя	Аксенов А.С	Успех
26.04.2017	users.changePassword.php	Функциональный	Изменение пароля пользователя	разработчик	Успех
26.04.2017	UserManager.java UserAccountFragment.java SetUserPasswordDialog.java	Ручной	Изменение пароля пользователя	разработчик	Успех
27.04.2017	users.changeImage.php	Функциональный	Изменение изображения пользователя	разработчик	Размер загруженного на сервер изображения равен 0 кБ
27.04.2017	users.changeImage.php	Функциональный	Изменение изображения пользователя	разработчик	Успех
28.04.2017	users.Image.php	Функциональный	Загрузка изображения пользователя	разработчик	Успех
28.04.2017	UserManager.java UserAccountFragment.java	Ручной	Загрузка изображения пользователя	Аксенов А.С.	Успех
29.04.2017	users.deleteImage.php	Функциональный	Удаление изображения пользователя	разработчик	При удалении изображения не происходит удаления имени изображения из базы
29.04.2017	users.deleteImage.php	Функциональный	Удаление изображения пользователя	разработчик	Успех

## Продолжение табл. П6.14

1	2		4	5	5
29.04.2017	UserManager.java UserAccountFragment.java	Ручной	Удаление изображения пользователя	разработчик	Успех
30.04.2017	requests.getActualRequest.java	Ручной	Получение списка заявок	разработчик	Список заявок пуст
02.05.2017	requests.getActualRequest.java	Ручной	Получение списка заявок	разработчик	Успех
02.05.2017	RequestManager.java RequestListFragment.java	Ручной	Загрузка списка заявок на устройство	разработчик	Не вычисляется расстояние до заявки. Не отображается дата добавления
02.05.2017	RequestManager.java RequestListFragment.java	Ручной	Загрузка списка заявок на устройство	разработчик	Не загружаются фотографии пользователя
02.05.2017	RequestManager.java RequestListFragment.java	Ручной	Загрузка списка заявок на устройство	разработчик	При прокрутке списка приложение останавливается
03.05.2017	RequestManager.java RequestListFragment.java	Ручной	Загрузка списка заявок на устройство	Аксенов А.С	Успех
03.05.2017	RequestList.java RequestListMapFragment.java	Ручной	Отображение маркеров заявок на карте	разработчик	Не отображается адрес заявки
04.05.2017	RequestList.java RequestListMapFragment.java	Ручной	Отображение маркеров заявок на карте	Филатьева Д.Е.	Успех
04.05.2017	RequestList.java RequestListFragment.java RequestDetailFragment.java	Ручной	Переход к сведениям детальной заявки	Филатьева Д.Е.	Успех
04.05.2017	RequestListMapFragment.java RequestDetailFragment.java	Ручной	Обработка нажатия маркера	разработчик	Успех
05.05.2017	request.updateRequestStatus.php	Функциональный	Изменение статуса заявки	разработчик	Не изменяется статус заявки
05.05.2017	request.updateRequestStatus.php	Функциональный	Изменение статуса заявки	разработчик	Успех
05.05.2017	RequestDetailFragment.java RequestManager.java	Ручной	Изменение статуса заявки	разработчик	Успех
06.05.2017	RequestManager.java RouteParser.java RequestDetailMapFragment.java	Ручной	Построение маршрута заявки	Филатьева Д.Е.	Успех
07.05.2017	offers.getOffers.php	Ручной	Получение списка предложений помощи	разработчик	Список предложений помощи пуст

## Продолжение табл. П6.14

1	2		4	5	5
07.05.2017	offers.getOffers.php	Ручной	Получение списка предложений помощи	разработчик	Успех
08.05.2017	offers.addOffer.php	Ручной	Добавление предложения помощи к заявке	разработчик	Успех
09.05.2017	OfferManager.java RequestDetailFragment.java	Ручной	Получение списка предложений помощи	разработчик	Список содержит не все сообщения
09.05.2017	OfferManager.java RequestDetailFragment.java	Ручной	Получение списка предложений помощи	разработчик	Список содержит не все сообщения
10.05.2017	OfferManager.java RequestDetailFragment.java	Ручной	Добавление предложения помощи к заявке	разработчик	Успех
10.05.2017	requests.getImage.php	Функциональный	Получение изображения заявки	разработчик	Успех
11.05.2017	NewRequestFetcherService.java	Ручной	Получение уведомлений о новых заявках	разработчик	Уведомления не приходят, когда приложение не запущено
11.05.2017	NewRequestFetcherService.java	Ручной	Получение уведомлений о новых заявках	разработчик	После получения уведомлений, сервис не запускается повторно через заданный интервал
12.05.2017	NewRequestFetcherService.java	Ручной	Получение уведомлений о новых заявках	Аксенов А.С.	Успех
13.05.2017	UsersSettingFragment.java UsersPreferences.java	Ручной	Изменение периода получения уведомлений	разработчик	Приложение остановлено
13.05.2017	UsersSettingFragment.java UsersPreferences.java	Ручной	Изменение периода получения уведомлений	разработчик	Успех
14.05.2017	UsersSettingFragment.java UsersPreferences.java	Ручной	Изменение радиуса получения уведомлений	разработчик	Успех
14.05.2017	RequestListMapFragment.java RequestDetailFragment.java	Ручной	Отображение заявок пользователя приложения	разработчик	Успех

Таблица П6.15

## Системное тестирование

Дата тестирования	Проверяемое требование	Описание теста	Результат тестирования
1	2	4	5
25.04.2017	Регистрация пользователя в системе	Проверка правильности ввода логина и пароля, сохранение указанных данных в базе	Успех
25.04.2017	Авторизация пользователя в системе по логину и паролю	Проверка статуса авторизации пользователя в системе, переход на форму для управления учетной записью	Ошибка. Пользователь не проходит процедуру авторизации
25.04.2017	Авторизация пользователя в системе по логину и паролю	Проверка статуса авторизации пользователя в системе, переход на форму для управления учетной записью	Успех
25.04.2017	Заполнение пользователем контактной информации	Изменение имени пользователя, добавление фотографии	Ошибка. Не возможно изменить фотографию пользователя
26.04.2017	Заполнение пользователем контактной информации	Изменение имени пользователя, добавление фотографии, сохранение данных в базе	Успех
27.04.2017	Добавление пользователем данных заявки на оказания техпомощи	Заполнение сообщения заявки	Успех
27.04.2017	Добавление пользователем данных заявки на оказания техпомощи	Определение местоположения, выбор местоположения на карте Google	На карте не определяется новое местоположение, а показывается ранее определённое значение местоположения
28.04.2017	Добавление пользователем данных заявки на оказания техпомощи	Определение местоположения, выбор местоположения на карте Google	Успех
28.04.2017	Редактирование данных заявки на оказание техпомощи пользователем в процессе заполнения	Изменение сообщения заявки, уточнение местоположения пользователя на карте	Успех
28.04.2017	Сохранение заявки в базе	Отправка запроса на сохранение данных заявки в базу с получением текстового сообщения, подтверждающим сохранение	Успех

## Продолжение табл. П6.15

1	2	4	5
05.05.2017	Просмотр списка заявок	Загрузка списка заявок на помощь из базы	Успех
05.05.2017	Просмотр списка заявок пользователя	Загрузка списка заявок пользователя приложения	Успех
05.05.2017	Просмотр списка заявок на карте	Отображение местоположения заявок на карте	Успех
10.05.2017	Просмотр сведений детальной заявки	Загрузка сведений детальной заявки. Добавление сообщений к заявке	Сообщение не добавляется
10.05.2017	Просмотр сведений детальной заявки	Загрузка сведений детальной заявки. Добавление сообщений к заявке	Успех
15.05.2017	Изменение статуса заявки	Изменение статуса заявки пользователя приложения	Успех
15.05.2017	Получение уведомлений о новой заявке	Получение уведомлений о новой заявке на экране блокировки	Успех
15.05.2017	Задание радиуса получения заявки	Изменение радиуса получаемых заявок пользователя в зависимости от его местоположения	Успех

Заключение: все выявленные в процессе тестирования ошибки работы программного обеспечения для мобильных устройств «Помощь на дороге» устранены.