МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

РАБОТА ПРОВЕРЕНА
Рецензент
_____
«___»_____ 2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___» _____ 2019 г.

Разработка программно–аппаратного комплекса для отслеживания местоположения домашнего питомца

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____ П.О. Шабуров
«___» _____2019 г.

Автор работы,
студент группы КЭ-452
_____ А. С. Лашманова
«___»_____2019 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С. В. Сяськов
«___»_____2019 г.

Челябинск 2019

АННОТАЦИЯ

Лашманова А. С. Разработка программно–аппаратного комплекса для отслеживания местоположения домашнего питомца. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2019, 140 с., 29 ил., библиогр. список – 15 наим.

В данной выпускной квалификационной работе выполнена разработка программно–аппаратного комплекса для отслеживания местоположения домашних питомцев. В ходе работы был произведен обзор существующих аналогов и выявлены их преимущества и недостатки. Также было разработано техническое задание, на основе которого реализован основной функционал веб-приложения.

Пояснительная записка состоит из введения, оглавления, основной части из 5 разделов, заключения, библиографического списка и 38 приложений.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В последнее время на улицах городов можно увидеть сотни бездомных кошек и собак или других питомцев, которые когда-то были домашними. Большинство из них потерялись или сбежали от своих хозяев. Причин, по которым они убегают из дома, множество: одни пугаются громких звуков, другие могут побежать за другими животными или за человеком, кто-то из них реагирует на новые запахи. В праздники число таких питомцев увеличиваются в разы. Многие из них пугаются салютов, криков или большого количества людей. Из-за всех этих причин питомцы оказываются на улице. Чаще всего они убегают далеко и не всем из них удается найти дорогу домой. На улице их поджидает много опасностей, поэтому человеку необходимо как можно быстрее найти своего любимца. Из-за долгого нахождения на улице многие питомцы становятся агрессивными и несут опасность для окружающих их людей или для других животных. Иногда они могут сбиваться в стаи и нападать на человека. К сожалению, большинство из этих животных, которое сейчас оказались на улице, в свое время не смогли найти их хозяева. Из этого можно сделать вывод, что существующие методы поиска не всегда являются эффективными, а существующие устройства не до конца отвечают требованиям большинства владельцев домашних питомцев. Поэтому необходимо создать устройство, которое будет отвечать желаниям потребителей. Создание такого устройства поможет не только находить людям своих питомцев, но и сократит число бездомных животных и сделает улицы городов более безопасными для нахождения людей.

1.АНАЛИЗ РЕШАЕМОЙ ЗАДАЧИ

1.1 Актуальность задачи

Ежедневно пропадают десятки домашних питомцев. Чаще всего питомец, который потерялся не может самостоятельно найти дорогу домой. Также известны случаи кражи животных, особенно дорогих и породистых. В таких случаях вероятность благоприятного, самостоятельного возращения питомца домой еще меньше. Тогда человеку приходится самому производить поиск своего домашнего животного. Обычно поиск питомцев основывается на информации об особенностях его поведения и характеристиках окружающей территории. Поиск, основанный на интеллектуальном анализе поведения питомца и характеристик местности, занимает много времени и не всегда приносит положительный результат. На улице питомцев поджидает большое количество опасностей, поэтому владелец должен найти своего друга как можно быстрее.

Н5а дверях подъездов, на досках объявлений или в интернете можно увидеть множество сообщений о пропаже домашних питомцев. Чтобы сократить количество таких объявлений необходимо создать трекер для домашних питомцев. Каждый сможет приобрести для своего любимца это устройство и тем самым сократит число потерявшихся питомцев.

Конечно, существует множество подобных устройств. Но модели, которые сейчас представлены на рынке отличаются высокой стоимостью или недостаточно большим функционалом. В связи с этим, было принято решение разработать программно-аппаратный комплекс для отслеживания местоположения домашних питомцев, который будет обладать преимуществами перед своими аналогами и отвечать основным требованиям потребителей.

1.2 Цель выпускной квалификационной работы

Целью выпускной квалификационной работы является разработка

программно-аппаратного комплекса, которое использует GPS для определения местоположения, а также создание приложения на Android, которое будет выводить данные о вашем местоположении и местоположении питомца на карту.

Вышеуказанная цель будет достигнута путем решения следующих задач:

1. Исследование принципа работы проектируемой системы.
2. Разработка программной части.
3. Разработка аппаратной части.
4. Выбор и закупка аппаратных модулей.
5. Проектирование готового изделия, с учетом требования минимального энергопотребления готового устройства.
6. Изготовление, сборка и отладка макетного образца трекера.

1.3 Обзор аналогов

На сегодняшний день существует достаточно большое разнообразие устройств с возможностью отслеживания местоположения. Они во многом облегчают жизнь людей, помогая не только определять свое местоположение или путь до необходимого места, но и местоположение детей, машины или домашних питомцев. К таким устройствам можно отнести различные «умные» браслеты для взрослых и для детей, навигаторы для машин, а также трекеры для животных. Так как «умные» браслеты и навигаторы для машин имеют множество функций, которые в трекерах для животных не будут приносить пользы, а будут только повышать энергопотребление устройства, мы сравним между собой только различные трекеры для питомцев.

Для выявления преимуществ и недостатков были выбраны для рассмотрения 3 наиболее популярных устройства для отслеживания местоположения питомцев: Garmin Astro 320 с ошейником DC50, TK STAR Pet tracker (TK 909), Garmin Delta Smart.

Навигатор Garmin Astro 320 с ошейником DC50 – устройство для отслеживания местоположения собак на охоте на расстоянии до 14,5 км. Ошейник получает сигнал по GPS и ГЛОНАСС далее передает информацию о местонахождении и траекторию движения собак на устройство Garmin Astro 320. Сигнал от ошейника передается каждые 5, 10, 15, 30 или 120 секунд в зависимости от настроек и заряда батареи. Навигатор работает с готовыми картами из интернета, а также с картами, которые может создавать пользователь самостоятельно из растровых изображений карт или снимков со спутника. На карте отображается местоположение собаки и владельца питомца. Также устройство может отображать состояние собаки (на карте будет видно сидит питомец, стоит или бежит). Кроме того, от ошейника поступает сигнал, если собака лает. Устройство может отображать информацию о нескольких питомцах одновременно, траекторию движения и пройденное собакой расстояние [1].

На рисунке 1.1 представлен внешний вид устройства:

Рисунок 1.1 – Навигатор Garmin Astro 320 с ошейником DC50

Устройство имеет ряд преимуществ таких, как:

- время автономной работы устройства до 54 часов в режиме экономии энергии (данные пересылаются лишь раз в две минуты) и до 24 часов в стандартном (радиосигнал с координатами посылается каждые 5 секунд);
- высокий уровень защиты от воды и пыли, класс защиты IP- 68, что позволяет погружать ошейник под воду на глубину до 10 метров;
- большой функционал трекера;
- корпус повышенной прочности, а также надежное крепление устройства к ошейнику защищает устройство от повреждений и понижает вероятность потери устройства.

Из недостатков можно выделить:

- высокую стоимость – около 50000 руб., следовательно, устройство не может быть использовано широким кругом лиц;
- вес ошейника 280 грамм;
- размеры трекера 61x169x36 мм, что не позволяет использовать трекер для маленьких собак, котов и других животных.

TK STAR Pet tracker (TK 909) – недорогой GPS трекер для животных. Для отображения местоположения питомца и выбора режима работы существует приложение для Android и IOS. Также можно получить ссылку на Google maps в СМС сообщениях, после вызова устройства по телефонному номеру SIM-карты с мобильного телефона. Реализована возможность отправки SMS команды на трекер для настроек режимов его работы и получения необходимых данных (координаты, заряд аккумулятора и др.). Устройство оснащено минимальными функциями для отслеживания местоположения животных. Имеет небольшие размеры и вес, следовательно,

подходит для кошек и собак мелких пород. В комплекте идет ошейник, на который крепится устройство. [2]

На рисунке 1.2 представлен внешний вид устройства:

Рисунок 1.2 – TK STAR Pet tracker (TK 909)

Из преимуществ можно выделить:

- низкую стоимость до 5000 р;
- небольшие размеры трекера 70х37х20 мм;
- наличие кроссплатформенного приложения;
- вес 46 грамм.

Из недостатков можно выделить:

- недостаточную защиту от воды и пыли, уровень которой IP-54, что не позволяет погружаться питомцу под воду, а выдерживает лишь небольшое попадание брызг на корпус;
- пластмассовое крепление, которое является недостаточно прочным и не может гарантировать хорошее крепление прибора к ошейнику, что повышает вероятность потери устройства;

– в данном трекере используются устаревшие модули GSM и GPS с высоким энергопотреблением, что не может обеспечить длительную работу устройства, а также недостаточную точность определения местоположения;
– мобильное приложение универсально для всех навигационных устройств фирм TKSTAR. Из недостатков мобильного приложения можно выделить: неполную русификацию приложения, отсутствие возможности отображения местоположения владельца устройства, а также не возможность добавления нескольких питомцев в одном профиле. Рейтинг приложения не высокий в Play Market 3,4 в App Store 3,8.

Garmin Delta Smart – устройство для дрессировки собак, а также для отслеживания местоположения и состояния питомца. С помощью мобильного приложения Garmin Canine можно использовать смартфон в качестве удаленного портативного тренажера, а также просматривать местоположение питомца и его перемещение. С помощью вибраций, звукового сигнала или корректирующей стимуляций от электронного ошейника можно поощрять или наказывать собаку за определенное поведение. Сигнал привлекает внимание собаки и позволяет безопасно и эффективно обеспечивать защиту от нежелательного или раздражающего поведения. Настраиваемые функции обучения позволяют выбрать тип сигнала / коррекции, которая необходима для вашего питомца. В качестве дополнительной меры защиты можно использовать метки «Keep Away» (приобретаются отдельно), которые создают небольшие (30-90см) зоны безопасности, чтобы собака держалась подальше от мусорных баков, цветочных клумб и прочих мест, где присутствие питомца нежелательно. Также существует пульт для удаленного управления устройством. Комплект с пультом стоит дороже. Устройство крепится к ошейнику питомца [3].

На рисунке 1.3 представлен внешний вид устройства:

Рисунок 1.3 – Навигатор Garmin Delta Smart

Преимущества:

- наличие дополнительного функционала, это не только трекер для отслеживания местоположения, но и устройство для дрессировки;
- длительное время работы;
- прочное крепление устройства к ошейнику;
- габариты устройства :63 х 35 х 34 мм;
- вес: 41 г;
- водонепроницаемость на глубине до 10 метров.

Недостатки:

- отсутствие в приложении русского языка;
- высокая стоимость около 16 000 рублей.

Вынесем основные характеристики трекеров в отдельную таблицу 1.1.

Таблица 1.1 – Сравнительная характеристика трекеров

	Garmin Astro 320 с ошейником DC50	**TK STAR Pet tracker (TK 909)**	**Garmin Delta Smart**
Цена	50000	4000	16000
Длительность автономной работы(в режиме экономии энергии)	До 56 часов	До 14 часов	До 20 часов
Вес (г)	280	46	41
Габариты (мм)	61 x 160 x 45	70 x 37 x 20	63 x 35 x 34
Класс зашиты от воды и пыли	IP-68	IP-54	IP-57
Наличие приложения	+	+	+
Поддержка русского языка	+	+/-	-
Определение местоположения владельца устройства	+	-	+

Вывод: из данной таблицы можно сделать вывод, что существующие устройства обладают некоторыми недостатками и не могут в полной мере удовлетворить всем требования владельцев домашних питомцев. В связи с этим, необходимо создать собственное устройство, которое будет иметь превосходство перед старыми, традиционными способами поиска животных, а также, обладать линейкой преимуществ перед своими аналогами.

1.4 Подбор модулей

Основным назначением нашего устройства будет получение координат местоположения устройства и передача их в приложение на Android, где они

будут отображаться на карте. Для разработки собственного устройства необходимо определиться с технологиями, используемыми для определения координат и передачи данных.

Для отслеживания местоположения используются различные технологии, такие как GPS, ГЛОНАСС, GSM и сеть Wi-Fi. Рассмотри каждую из них подробнее:

Самой обширной и развитой инфраструктурой определения местоположения является Глобальная система позиционирования (GPS). Эта система определяет расстояние, а значит, и местоположение путем расчета времени, за которое радиосигнал доходит от спутникового передатчика до наземного приемника. Основа системы GPS — это двадцать четыре спутника, вращающихся вокруг Земли таким образом, что из любого места планеты в любое время видны как минимум пять из них. На каждом из спутников установлены точнейшие часы, а их безошибочные координаты в любой момент времени известны. Спутники передают сигналы, содержащие точное время и координаты по их приборам, и время прохождения сигнала позволяет рассчитать расстояние до передатчика. Сигналов от четырех спутников достаточно, чтобы вычислить местоположение приемника в трёхмерном пространстве [4]. Координаты определяются с погрешность 2–4 метра. Из недостатков GPS можно выделить невозможность определения местоположения в зданиях. Если ваш питомец находится в четырех стенах, датчик в ошейнике вряд ли получит обычный сигнал GPS. Поэтому в современных модулях используется технология A-GPS. Преимущество данной технологии заключается в том, что A-GPS зависит не только от спутников, но и от сетевых ресурсов. Поэтому, A-GPS способен получать координаты даже в замкнутых помещениях, даже межэтажные перекрытия не являются помехой. Также данный тип технологии сокращает время, необходимое для фактического поиска вашего питомца после включения

устройства. Наличие данной технологии является важным параметром при выборе модуля.

ГЛОНАСС (Глобальная Навигационная Спутниковая Система) – российская спутниковая система навигации. Функционирует с 1982 года. Основой системы являются 24 спутника, движущихся над поверхностью Земли в трёх орбитальных плоскостях с наклоном орбитальных плоскостей 64,8° и высотой орбит 19 100 км. Принцип измерения аналогичен GPS. Основное отличие от системы GPS в том, что спутники ГЛОНАСС в своём орбитальном движении не имеют резонанса (синхронности) с вращением Земли, что обеспечивает им большую стабильность. Также срок службы спутников ГЛОНАСС меньше, чем у GPS. Местоположение определяется с погрешностью 2-6 метров.

Вычисление местоположения на основе сигналов GSM тоже широко распространено. Суть технологии в том, чтобы анализировать информацию о тех базовых станциях сотовой связи, которые находятся поблизости. Однако точность заметно меньше, чем у GPS, погрешность увеличивается до 50–100 метров, а если измерения местоположения производятся в сельской местности или в небольших городах, где в любой точке имеется сигнал только от одной, максимум — двух базовых станций одного оператора, то возможная погрешность увеличивается до 10–15 км [5]. Использование данной технологии не подходит в нашем случае, так как питомец может оказаться вдалеке от базовых станций.

Также существует метод позиционирования с помощью сетей Wi-Fi. Этот метод уже несколько лет используется компанией Cisco и заключается он в том, чтобы определить силу сигнала от клиента на 3х-4х точках доступа Wi-Fi и в зоне пересечения возможного расположения клиента относительно каждой точки спозиционировать устройство [6]. Данный метод является довольно информативным. При правильном развесе точек доступа он позволяет с высокой вероятностью определить координату клиента с

точностью 5-7 м. Проблема заключается в том, что сеть Wi-Fi существуют не везде и поэтому данный метод не может обеспечить успешный исход поиска питомца вдалеке от точек доступа Wi-Fi. Использование данной технологии не подходит в нашем случае, так как питомец может оказаться вдалеке от точек доступа Wi-Fi.

Таким образом, из представленных решений определение местоположения с помощью GPS и ГЛОНАСС является наиболее оптимальным. В таблице 1.2 приведено сравнение представленных на рынке модулей, использующих данные технологии между собой для выявления, наиболее подходящего [7].

Таблица 1.2 – Сравнение GPS/ГЛОНАСС модулей

	Quectel L76-L	**GlobalTop FireFly-X1**	**Tracker GNS1316**	**Geostar Navigation GS501E**
Чипсет	MTK3333	MTK3333	STA8088	SiRFstrV 5e
Навигационные системы	ГЛОНАСС/GPS	ГЛОНАСС/GPS	ГЛОНАСС/GPS	ГЛОНАСС/GPS
Чувствительность при слежении (дБ)	-165	-165	-160	-164
Чувствительность при обнаружении (дБ)	-148	-145	-145	-145
Энергопотребление при слежении (мА)	22	30	26	26
Энергопотребление при обнаружении (мА)	29	33	30	28
Поддержка технологии A-	+	+	+	+

GPS				

Продолжение таблицы 1.2

	Quectel L76-L	**GlobalTop FireFly-X1**	**Tracker GNS1316**	**Geostar Navigation GS501E**
Время холодного старта (с)	35	35	35	35
Время горячего старта (с)	1	1	1	1
Напряжение питания (В)	2,8 ~ 4,5	3 ~ 4,3	3 ~ 3,6	3,0 ~ 3,6
Тип антенны	Внешняя	Внешняя	Внешняя	Внешняя
Габариты (мм)	10,1x9,7 x 2,5	9,0 x 9,5 x 2,1	13x15,9x2,8	15,2x13x2,2
Цена (руб.)	780	870	750	750

Вывод: в таблице представлено сравнение модулей по основным характеристикам. Стоит отметить, что время холодного старта – это время первого включения модуля или время включения после перемещения модуля на большие расстояния. За это время скачивается альманах, содержащий параметры орбит всех спутников. Время горячего старта – это время запуска после кратковременного выключения устройства. Данные альманаха не устарели, поэтому приемник сразу может использовать сигнал всех ранее найденных спутников. Из представленных модулей в соответствии с требованиями к минимальному энергопотреблению и необходимостью обеспечения минимальных размеров готового устройства следует выбрать модуль Quectel L76-L.

Для передачи данных используются такие технологии как: GPRS/GSM, Bluetooth, 3G, Lora.

GPRS надстройка над технологией мобильной связи GSM, осуществляющая пакетную передачу данных. GPRS позволяет модулю производить обмен данными с другими устройствами в сети GSM и с внешними сетями, в том числе Интернет. GPRS предполагает тарификацию по объёму переданной/полученной информации, а не времени, проведённому онлайн. Весь поток данных отправителя разбивается на отдельные пакеты и затем доставляется получателю через неиспользуемые в данный момент голосовые каналы, где пакеты собираются воедино, и совсем необязательно, что все пакеты пойдут одним маршрутом.

Bluetooth — это беспроводной интерфейс с небольшим радиусом действия, созданный в 1994 году инженерами шведской компании Ericsson. Основным назначением Bluetooth является создание так называемых персональных сетей, которые обеспечивают возможность обмена данными между расположенными поблизости (внутри одного дома, помещения, транспортного средства и т.д.) настольными и портативными ПК, периферийными и мобильными устройствами и пр. [8]. Недостатком Bluetooth является невозможность отправления данных на большие расстояние, следовательно, данная технология не подходит для нашего модуля.

3G (технологии мобильной связи 3 поколения) — набор услуг, который объединяет как высокоскоростной мобильный доступ с услугами сети Интернет, так и технологию радиосвязи, которая создаёт канал передачи данных [9]. 3G отличается высокой скоростью, но так как в нашем устройстве необходимо передавать только координаты использование данной технологии не имеет смысла, так как это повышает энергопотребление и стоимость модуля.

LoRa – это технология беспроводной передачи данных, использующая метод модуляции радиоволн, который может осуществляться чипами-

трансиверами LoRa от компании Semtech [10]. Этот метод модуляции позволяет передавать маленькие порции данных (это значит, что у него небольшая пропускная способность) на большие расстояния. Кроме того, у него высокая стойкость к помехам и пониженное энергопотребление. Использование технологии LoRa для взаимодействия модуля и приложения является перспективным решением. Но так как покрытие сетей LoRaWAN на данный момент не является достаточно обширным использование этой технологии на существующем этапе разработки не имеет смысла.

Таким образом, из представленных решений передача данных с помощью GSM/GPRS является наиболее оптимальной. Сравним в таблице 1.3 представленные на рынке модули, которые используют данную технологию между собой для выявления, наиболее подходящего [11].

Таблица 1.3 – Сравнение GSM/GPRS модулей

	Quectel M66	**Quectel M95**	**Quectel M10**	**Quectel M80**
Технология	GSM/GPRS	GSM/GPRS	GSM/GPRS	GSM/GPRS
Частоты (МГц)	850/900/1800 /1900	850/900/1800 /1900	850/900/1800 /1900	850/900/1800 /1900
Энергопотребление в режиме ожидания (мА)	1,2	1,5	1,3	1,1
Напряжение питания (В)	3,3 ~ 4,6	3,3 ~ 4,6	3,3 ~ 4,6	3,4~4,5
Габариты (мм)	17,7 × 15,8 × 2,3	19,9 × 23,6 × 2,6	29 × 29 × 3,6	23 × 25 × 2,6
Цена (руб.)	700	650	680	700

Вывод: из представленных модулей в соответствии с требованиями к минимальному энергопотреблению и необходимостью обеспечения минимальных размеров готового устройства следует выбрать модуль Quectel M66.

Также на рынке предоставлены модули, которые совмещают в себе GPS/ГЛОНАСС для определения местоположения и GSM/GPRS для передачи данных. Один из таких модулей это Quectel MC60. В таблице 1.4 сравним его характеристики с характеристиками выбранных раннее модулями.

Таблица 1.4 – Сравнение модулей

	Quectel L76-L + Quectel M66	**Quectel MC60**
	GNSS часть	
Чипсет	MTK3333	MTK3333
Навигационные системы	ГЛОНАСС/GPS	ГЛОНАСС/GPS
Чувствительность при слежении (дБ)	-165	-167
Чувствительность при обнаружении (дБ)	-148	-152
Энергопотребление при слежении (мАч)	22	22
Энергопотребление при обнаружении (мАч)	29	29
Поддержка технологии A-GPS	+	+
Время холодного старта (с)	35	35
Время горячего старта (с)	1	1
Напряжение питания (В)	2,8 ~ 4,5	2,8~4,3
Тип антенны	Внешняя	Внешняя
	GPRS часть	
Технология	GSM/GPRS	GSM/GPRS
Частоты (МГц)	850/900/1800/1900	850/900/1800/1900
Энергопотребление в режиме ожидания (мАч)	1,2	1,2
Напряжение питания (В)	3,3 ~ 4,6	3,3~4,6
	Общее	

Габариты (мм)	10,1 × 9,7 × 2,5 17,7 × 15,8 × 2,3	18,7 × 16 × 2,1
Цена (руб)	1480	1100

Вывод: Из таблицы видно, что использование модуля, который объединяет в себе 2 технологии является более выгодным так как цена на один такой модуль меньше, чем цена на 2 отдельных модуля, а также это решение уменьшает окончательные размеры устройства. А также выбор модуля Quectel MC60 упрощает работу над передачей данных. Нет необходимости взаимодействовать между GSM и GPS модулями. Также данный модуль поддерживает технологию OpenCPU. Эта технология позволяет программировать модуль без подключения внешнего микроконтроллера.

1.5 Подбор аккумулятора

В настоящее время наиболее распространены Li-Ion и Li-Pol аккумуляторы [12].

Li-Ion – это сокращение от литий-ионный, а Li-Pol – от литий-полимерный. Окончание «ионный» и «полимерный» — это указание на катод. Литий-полимерный аккумулятор состоит из полимерного катода и твердого электролита, а литий-ионный аккумулятор – из углерода и жидкого электролита. Оба аккумулятора перезаряжаемые, и потом, в том или ином смысле, они оба выполняют одну и ту же функцию.

При сравнении двух типов аккумуляторов можно выделить следующие преимущества литий-полимерных аккумуляторов:

- Меньший вес и размеры при идентичных характеристиках;
- Меньшее время, необходимое для зарядки;
- Более высокая (почти в два раза) емкость при одинаковом общем объеме батареи;

- Меньшая взрывоопасность батареи.

Исходя из этого лучшим решением будет использование литий-полимерного аккумулятора.

Сравним представленные на рынке модели аккумуляторов в таблице 1.5. Основными параметрами при выборе аккумулятора являются не большие размеры и вес:

Таблица 1.5 – Сравнение аккумуляторов

	Li-Pol 018265	**Li-Pol 018251**	**Li-Pol 018251**	**UK 4004050**
Тип аккумулятора	Li-Pol	Li-Pol	Li-Pol	Li-Pol
Емкость (мАч)	1500	900	550	1200
Габариты (мм)	2 × 55 × 85	3 × 50 × 57	4 × 40 × 45	4 × 40 × 50
Цена (руб)	310	245	245	240

Вывод: Исходя из информации, представленной в таблице был выбран литий-полимерный аккумулятор UK 404050 На рисунке 1. 4 представлено изображение выбранного аккумулятора. Необходимо приобрести данный аккумулятор в количестве двух штук, для возможности замены во время зарядки. Также для данного аккумулятора было подобрано зарядное устройство.

Рисунок 1.4 – Аккумулятор UK 4004050

1.6 Подбор программных компонентов

В данном пункте описан выбор программных компонентов системы.

1.6.1 Выбор СУБД

Система управления базами данных (СУБД) – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

В нашей системе нет необходимости в хранении и записи большого объема данных. А как одно из главных требований к БД можно выделить мобильность базы данных, так как важна возможность создания в будущем кроссплатформенного приложения. Исходя из этого оптимальным будет использование БД SQLite [13]. Это наиболее распространенная СУБД для мобильных приложений. Эта система базируется на файлах, что предоставляет довольно широкий набор инструментов для работы с ней, по сравнению с сетевыми СУБД. При работе с БД обращения происходят

напрямую к файлам и благодаря этому, SQLite легко встраивается в приложения и является мобильной.

1.6.2 Выбор языков программирования

Программная часть в данной работе состоит из двух частей:

1. Серверная часть.
2. Клиентская часть.

Для разработки серверной части приложения необходимо выбрать технологию. На данный момент самыми популярными считаются технологии PHP (язык для написания серверных сценариев (скриптов)) и ASP.NET (средство для разработки веб-приложений от Microsoft. ASP.Net – это не язык программирования, это технология, включающая в себя множество компонент).

При их сравнении можно заметить, что ASP.NET проще PHP в плане реализации одних и тех же задач. Это объясняется тем, что большинство функций являются встроенными в ASP.NET (например, авторизация, сохранение состояния перегружаемой страницы, гриды с автоматической привязкой данных) [11]. Ещё одним плюсом ASP.NET является то, что программа пишется на строго типизированных компилируемых .NET языках и поэтому технология существенно упрощает отладку по сравнению PHP. Исходя из этого для разработки нашего приложения, мы будем использовать технологию ASP.NET, а в частности язык C#.

C# - объектно-ориентированный язык программирования. Данный язык используется для разработки веб-приложений на платформе ASP.NET.

Также выбор ASP.NET можно обосновать необходимостью масштабировать систему, так как число зарегистрированных в системе пользователей будет постоянно увеличиваться.

Наиболее популярные языки для разработки приложений на Android это Java и Kotlin:

Java – это наиболее распространенный язык для разработки мобильных приложений на Android. Является сильно типизированным объектно-ориентированным языком программирования.

Kotlin – это современный статически типизированный объектно-ориентированный язык программирования, компилируемый для платформ Java и JavaScript.

Для разработки клиентской части приложения был выбран язык Java так как язык имеет множество готовых библиотек и SDK, что ускоряет процесс разработки.

1.6.3 Выбор среды разработки

Необходимо выбрать среду разработки для серверной и клиентской части.

1. Для разработки серверной части будет использована Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом. Является наиболее распространенной средой программирования. Обладает рядом преимуществ таких, как: высокая скорость разработки, поддержка различных языков программирования при разработке, лучшие инструменты отладки, наличие встроенного Web-сервера.

2. Для разработки клиентской части будет использована Android Studio – интегрированная среда разработки для работы с Android. Находится в свободном доступе, поддерживается такими ОС как Windows, OS X и Linux. Используемыми языками программирования являются C++, Java и Kotlin.

Является наиболее распространенной средой разработки для разработки приложений на Android. Имеет ряд преимуществ таких, как: гибкость среды разработки, большой набор функций, возможность тестирования корректности работы приложений на различных системах можно производить в эмуляторе, возможность разработки под различные версии Android, а также достаточно большая библиотека с готовыми шаблонами и компонентами для разработки.

1.6.4 Выбор хостинг провайдера

В наше время на рынке представлено множество сервисов, предоставляющие услуги хостинга приложений. Далее представлены наиболее популярные хостинг-провайдеры на территории Российской Федерации это REG.ru и Nic.ru.

REG.ru – является аккредитованным регистратором доменных имен в нескольких сотнях доменных зон. Является одним из крупнейших регистраторов в России. Высокий уровень надежности. Предоставляет множество дополнительных услуг и сервисов. Поддержка работает на высоком уровне.

Стоимость аренды хостинга с необходимыми характеристиками: 179 руб/мес

Nic.ru – является одним из старейших и надежных регистраторов доменных имен в нескольких сотнях доменных зон. Является одним из крупнейших регистраторов в России. Высокий уровень надежности.

Стоимость аренды хостинга с необходимыми характеристиками: 200 руб/мес

В представленном ниже проекте был выбран хостинг-провайдер Reg.ru. Выбор хостинг-провайдера Reg.ru обусловлен тем, что данный хостинг

предоставляет наиболее экономически-выгодные условия для аренды VPS сервера.

1.6.5 Выбор карты

На рынке картографических и справочных сервисов можно выделить двух основных конкурентов:

- Яндекс.Карты;
- Google Maps.

Данные карты достаточно похожи, но Google Maps имеет более широкое покрытие и функцию. Именно поэтому для разработки системы были выбраны эти карты.

1.7 Вывод

Для разработки нашего проекта были выбраны следующие программные компоненты:

– СУБД: SQLite;
– Языки программирования: C# и Java;
– Среда разработки серверной части: Visual Studio;
– Среда разработки клиентской части: Android Studio;
– Карты – Google maps;
– Хостинг провайдер: Reg.ru.

В аппаратной части был выбран модуль GNSS/GPRS модуль Quectel MC60 и аккумулятор UK 404050.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

Для реализации данной системы необходим следующий набор подсистем:

1. Клиентское приложение на Android. Приложение обеспечивает доступ пользователя к системе, а также возможность вывода результата в приложение на карту.
2. Графический интерфейс клиентского приложения. Визуальное отображение регистрации и авторизации в системе, отображение карты и отметок местонахождения.
3. Серверное приложение. Обеспечивает взаимодействие аппаратной реализации системы и клиентского приложения. Необходимо расположить на хостинге.
4. База данных SQLite. База данных обеспечивает хранение данных о пользователях и питомцах, а также о их координатах.
5. Аппаратная реализация системы. Устройство, позволяющее определять местоположение питомца и отправлять полученные координаты пользователю.

2.1 Функциональные требования

Для того чтобы выделить функциональные требования к системе построим диаграмму прецедентов она отображает события, возникающие в системе с точки зрения пользователя, изображены с помощью диаграммы прецедентов в нотации UML [14]. На рисунке 2.1 изображена диаграмма прецедентов.

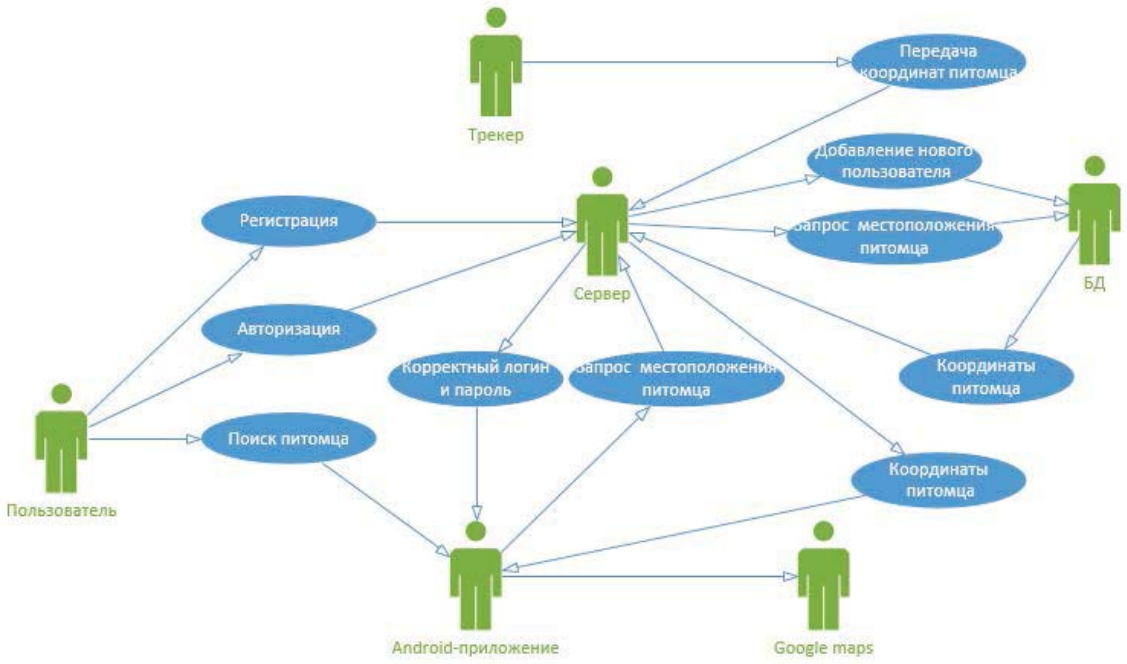

Рисунок 2.1 – Диаграмма вариантов использования системы

На рисунке изображена диаграмма вариантов использования системы. Данная диаграмма используется для отражения прагматики разрабатываемой системы, описания структуры объектов, из которых она состоит, их взаимосвязи и атрибутов. Роли распределены между Android-приложением, пользователем, трекером, базой данных и Google-картами. В овалы, в свою очередь, заключены прецеденты использования разрабатываемой системы.

Если пользователь не был ранее зарегистрирован в системе, то перед началом работы ему необходимо пройти процедуру регистрации для этого необходимо указать в соответствующих полях имя учетной записи, пароль, фамилию, имя, дату рождения, кличку питомца и идентификатор ошейника. Пользователям, ранее проходившим данную процедуру необходимо авторизоваться, чтобы иметь возможность работать в системе: для этого им необходимо ввести в соответствующем интерфейсе системе свои имя учетной записи и пароль.

После этого пользователь может перейти на карту и посмотреть местоположение питомца или свое собственное. Трекер в режиме реального

времени получает со спутников свои координаты. После этого они могут быть отражены в приложении на карте.

2.2 Нефункциональные требования

2.2.1 Общие требования

- Приложение не должно зависеть от аппаратной части. Приложение должно иметь возможность масштабироваться и не зацикливаться на одном устройстве;
- Данные должны передаваться в зашифрованном виде. Т.к. система будет хранить email пользователей и их пароли, должна обеспечиваться защищенная передача этих данных;
- Невозможность прохождения процедуры регистрации без заполнения всех полей, также вводимая информация должна соответствовать требованиям поля;
- Наличие идентификатора или QR-кода для обеспечения уникальности устройства;
- Доступ к системе возможен только для зарегистрированных пользователей, после прохождения процедуры регистрации;
- Время работы устройства определяется характеристиками устройства питания, устройство должно находиться в спящем режиме и включаться по нажатию кнопки, также необходимо наличие съемного аккумулятора;
- Размеры устройства не должны превышать следующих параметров: 15*60*60 мм;
- Вес устройства не должен превышать 60 грамм

2.2.2 Требования к системе

Система должна работать с любым Android – устройством. Версия Android должна быть не менее 4.4.

2.2.3 Требования к документации

В документации на устройство должны содержаться технические характеристики устройства, которые включают следующие требования: вес, размеры, тип источника питания, класс пыле- и влаго- защищенности, а также правила регистрации питомца в системе. Документация должна предоставляться Заказчику в печатном виде на бумажных носителях, а также в электронном виде, в виде файлов формата MS Word (DOC) или Adobe Acrobat Reader (PDF), пригодных для печати. Документация должна быть на русском языке.

3. ПРОЕКТИРОВАНИЕ

3.1 Функциональный состав

Для описания функционального состава системы можно представить функциональную схему. Данная схема поясняет отдельные виды процессов, протекающих в целостных функциональных блоках. На рисунке 3.1 представлена функциональная схема системы.

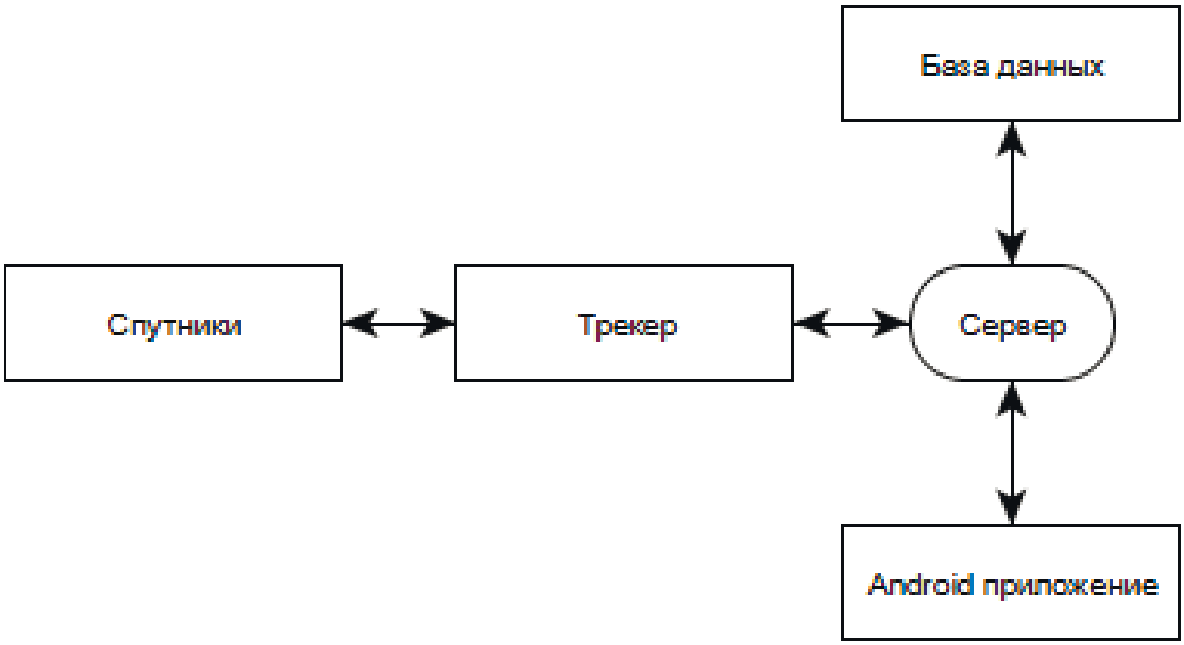

Рисунок 3.1 – Функциональная схема системы

Вся система будет взаимодействовать через сервер. Трекер будет получать информацию о своем местоположении со спутников и передавать их на сервер, который будет обрабатывать полученные координаты и хранить их. В приложении, с помощью запросов, будет реализована возможность получения необходимой информации из БД, а также отправка данных в базу при регистрации новых пользователей. В БД будет содержаться информация о пользователях, питомцах и их координатах. Также в приложении будет отображаться карта, с полученными координатами.

3.2 Структурный состав

Для описания структурного состава программной части системы, необходимо построить структурную схему.

Структурная схема – это совокупность элементарных звеньев объекта и связей между ними. Под элементарным звеном подразумевается часть объекта, которая реализует элементарную функцию.

Внутренние компоненты отвечают за логику функционирования и хранение данных, а внешние за визуальное взаимодействие с системой.

Структурная схема системы представлена на рисунке 3.2.

Рисунок 3.2 – Структурная схема Android-приложения

Структурная схема Android-приложения включает в себя внутренние и внешние компоненты. Состав внешних компонентов: интерфейс приложения

и графические элементы на карте. Во внутренние компоненты входят: база данных, функции ввода/вывода данных и сервер.

3.3 Описание данных

Составим схему БД для нашего проекта. Схема базы данных— её структура, описанная на формальном языке, поддерживаемом СУБД. Структурная схема базы данных системы представлена на рисунке 3.3.

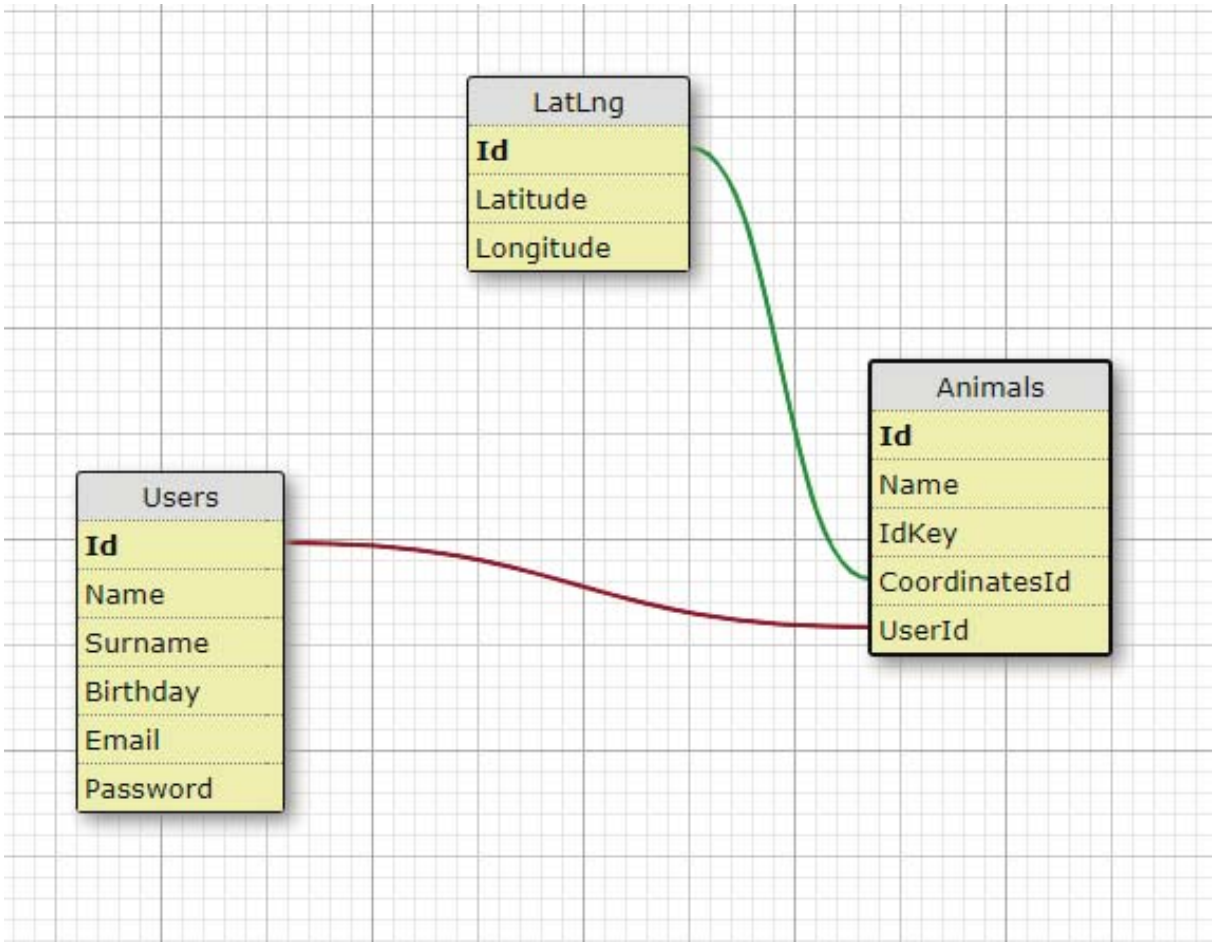

Рисунок 3.3 – Схема базы данных

Так как в качестве СУБД была использована SQLite, данная система поддерживает не все типы данных, поэтому нам необходимо указать какие типы данных мы используем:

– INTEGER – значение представляет собой целое число со знаком;
– TEXT – значение представляет собой текстовую строку, хранящуюся с использованием кодировки базы данных. Так как в SQLite не

существует специального типа, для даты и времени для соответствующих полей можно использовать тип данных text;

– REAL – значение представляет собой значение с плавающей запятой.

База данных состоит из трех таблиц. В таблице 3.1 представлено их краткое описание.

Таблица 3.1 – Описание таблиц базы данных

Таблица	**Описание**
User	содержит информацию о пользователе
Animal	содержит информацию о питомце
LatLng	содержит координаты питомца

В таблице 3.2 представлено описание полей таблицы Animal.

Таблица 3.2 – Описание полей таблицы User

Поле	**Тип данных**	**Описание**
Id (первичный ключ)	INTEGER	уникальный идентификатор пользователя
Name	TEXT	имя
Surname	TEXT	фамилия
Birthday	TEXT	дата рождения
Email	TEXT	email пользователя, указанный при регистрации
Password	TEXT	пароль, указанный при регистрации

В таблице 3.3 представлено описание полей таблицы Animal.

Таблица 3.3 – Описание полей таблицы Animal

Поле	Тип данных	Описание
Id (первичный ключ)	INTEGER	уникальный идентификатор питомца
Name	TEXT	кличка питомца
IdKey	INTEGER	идентификатор ошейника, который будет указан в документации к ошейнику
CoordinatesID	INTEGER	(внешний ключ, ссылается на таблицу LatLng) идентификатор координат питомца.
UserId	INTEGER	(внешний ключ, ссылается на таблицу User) идентификатор хозяина питомца.

В таблице 3.4 представлено описание полей таблицы LatLng.

Таблица 3.4 – Описание полей таблицы LatLng

Поле	Тип данных	Описание
Id (первичный ключ)	INTEGER	уникальный идентификатор координат питомца
Latitude	REAL	широта
Longitude	REAL	долгота

3.4 Функциональная схема аппаратной части системы

.Рассмотрим характеристики и возможности датчиков и модулей, которые используются для проектирования аппаратной части комплекса для отслеживания местоположения питомца.

Функциональная схема взаимодействия элементов системы представлена на рисунке 3.4.

Рисунок 3.4 – Функциональная схема трекера

Трекер будет состоять из:

– микроконтроллера;
– GNSS/GSM модуля;
– источника питания.

На контроллер поступают данные с модуля GPS в которых содержится информация о местоположении питомца. Посредством GSM модуля, контроллер взаимодействует с сервером, как на отправку данных, так и на прием. Для возможности передачи данных необходимо подключить SIM-карту и антенны. С помощью источника питания (аккумулятора) устройство получает необходимое напряжение.

4. РЕАЛИЗАЦИЯ

4.1 Программная реализация серверной части

Для постоянного доступа к системе сервер был расположен на хостинге, IP адрес которого 80.78.248.202.

В основе серверной части нашего клиент-серверного приложения лежит фреймворк ASP.NET MVC. Фреймворк – программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта. ASP.NET MVC – одна из самых популярных и надежных основ для построения веб-приложений любой сложности. Архитектура MVC переводится как «модель-представление-контроллер». Суть этой архитектуры сводится к разделению модели данных, пользовательского интерфейса и управляющей логики контроллера. Программа серверной части состоит из:

- Model (модель) – это часть, которая содержит в себе функциональную бизнес-логику приложения. Содержит или представляет данные, с которыми работают пользователи;
- Controller (контроллер) – обрабатывает поступающие запросы, выполняет операции с моделью и выбирает представления для визуализации пользователю.

Так как на сервере нет необходимости в визуальном представлении на данный момент View не был реализован.

Взаимодействие между ними происходит с помощью HTTP запросов. Hyper Text Transfer Protocol (протокол передачи гипертекста) — протокол прикладного уровня передачи данных изначально — в виде гипертекстовых документов в формате «HTML», в настоящий момент используется для передачи произвольных данных.

В корне приложения находятся четыре основные папки:

1. Controllers – данная папка содержит классы, которые используются для обработки поступающих запросов. Контроллер является центральным компонентом в архитектуре MVC. Она используются для организации логики поступающих запросов. По соглашениям об именовании названия контроллеров должны оканчиваться на суффикс "Controller", остальная же часть до этого суффикса считается именем контроллера. Данная папка содержит 2 класса: AccountController, AnimalLocationController. Контроллеры в данных папках реагируют на входящие HTTP запросы и обрабатывают их. Нужно отметить, что каждый метод контроллера отвечает за один HTTP запрос. Адрес запроса, на который должен реагировать каждый метод прописывается в атрибуте “Route”. С помощью атрибутов [HttpGet], [HttpPost], [HttpDelete] или [HttpPut] определяется тип обрабатываемого запроса. Описание разработанных контроллеров:

– AccountController – данный контроллер описывает реакцию сервера на входящие запросы при работе с учетными записями пользователя. Исходный код данного контроллера представлен в листинге А.1. приложения А;
– AnimalLocationController – данный контроллер описывает реакцию сервера на входящие запросы при работе с координатами питомца Исходный код данного контроллера представлен в листинге А.2. приложения А.

В таблице 4.1 представлено описание методов класса AccountController.

Таблица 4.1 – Описание методов класса AccountController»

Метод	**Описание**
Login	этот метод используется, для обработки информации о пользователе при авторизации.

Продолжение таблицы 4.1

Метод	Описание
Registration	этот метод используется, для обработки информации о пользователе при регистрации.
GetIdentity	этот метод вызывается если пользователь был найден, то создается объект ClaimsIdentity

В таблице 4.2 представлено описание методов класса AnimalLocationController.

Таблица 4.2 – Описание методов класса AnimalLocationController

Метод	Описание
Put	этот метод используется для отправки запроса на получение координат питомца
Get	этот метод используется для получения координат питомца
FindAnimal	этот метод используется при определении местоположения питомца

2. Models – содержатся классы описания данных, которые используются в приложении. Модели хранят только состояние в виде свойств. Данная папка содержит следующе папки:

- AnimalLocation – данная папка используется для работы с координатами питомца. Содержит два класса:
  - ModelForGettingAnimalLocation – данный класс определяет ряд свойств, которые описывают информацию для получения координат питомца. Содержит поля, содержащие email пользователя и идентификатор устройства. Исходный код данного класса представлен в листинге Б.1. приложения Б;

- ModelForPutAnimalLocation – данный класс определяет ряд свойств, которые описывают информацию для отправки запроса на получение координат питомца. Содержит поля, содержащие email пользователя, идентификатор устройства и координаты питомца. Исходный код данного класса представлен в листинге Б.2. приложения Б.

- Database – данная папка содержит информацию о том, что хранится в базе данных, а также функциональную бизнес-логику приложения. Содержит следующие классы:
  - Animal – данный класс определяет ряд свойств, которые описывают информацию о питомце: уникальный идентификатор, имя, идентификатор ошейника и идентификатор координат питомца;
  - DatabaseContext – определяет контекст данных, используемый для взаимодействия с базой данных. Класс DatabaseContext наследуется от абстрактного класса DbContext, который определен внутри Entity Framework и предоставляет широкие возможности по работе с базой данных. Также в классе DataContext имеются два поля: Users, Animals, которые интерпретируются в базу данных как таблицы, также Entity Framework берет на себя работу с созданием связей между таблицами. Исходный код данного класса представлен в листинге В.1. приложения В;
  - LatLng – данный класс определяет ряд свойств, которые описывают координаты: широта и долгота. Также данный класс содержит метод Update для обновления координат питомца;
  - User – данный файл определяет ряд свойств, которые описывают информацию о пользователе: уникальный идентификатор, имя, фамилию, день рождения, логин и

пароль и уникальный идентификатор питомца. Также данный класс содержит класс UserComparer, который применяется для сравнения пользователей. Исходный код данного класса представлен в листинге Г.1. приложения Г.

Также существует 3 класса:

– AccountInfo – данный класс определяет ряд свойств, которые описывают информацию об аккаунте пользователя: имя, фамилию, маркер доступа, информацию о питомце. Исходный код данного класса представлен в листинге Д.1. приложения Д;
– LoginModel – данный класс определяет ряд свойств, которые описывают информацию необходимую при авторизации пользователя: логин и пароль. Данная модель хранит только состояние в виде свойств. Исходный код данного класса представлен в листинге Е.2. приложения Е;
– RegistrationModel – данный класс определяет ряд свойств, которые описывают информацию необходимую при регистрации пользователя: уникальный идентификатор, имя, фамилия, день рождения, логин и пароль, а также информацию о питомце. Данная модель хранит только состояние в виде свойств. Также существует метод ToUser(), который используется для добавления пользователя в БД. Исходный код данного класса представлен в листинге Е.1. приложения Е.

3. Authenticate – содержатся классы, которые описывают методы для авторизации. Механизм авторизации использует JWT-токены. JWT (или JSON Web Token) представляет собой веб-стандарт, который определяет способ передачи данных о пользователе в формате JSON в зашифрованном виде. Данная папка содержит 2 класса:

– AuthOptions – класс описывает характеристики токена для авторизации. Содержит 4 константы. Константа ISSUER представляет издателя

токена. Здесь можно определить любое название. AUDIENCE представляет потребителя токена - опять же может быть любая строка, но в данном случае указан адрес текущего приложения. Константа LIFETIME определяет время жизни токена. Константа KEY хранит ключ, который будет применяться для создания токена. Исходный код данного класса представлен в листинге Й.1. приложения Й;

– TokenGeneration – класс, который генерирует специальный токен для авторизации. Содержит метод GetToken, который генерирует токен. Тут создается JWT-токен и происходит инициализация констант из класса AuthOptions. Исходный код данного класса представлен в листинге Й.2. приложения Й.

4. Migration – позволяет вносить изменения в базу данных при изменениях моделей и контекста данных. Папка содержит два файла: Configration.cs и файл начальной миграции. Первый файл содержит объявление одноименного класса Configuration, который устанавливает настройки конфигурации Файл начальной миграции устанавливает, как база данных определяется на данный момент. Исходные коды данных классов представлены в листинге Ж.1. и Ж.2 приложения Ж.

5. Также реализовано 2 класса Startup и Program.

– Startup – класс, которые определяет настройки веб сервиса. Является входной точкой в приложение ASP.NET Core. В данном классе есть конструктор и два метода: ConfigureServices и Configure. Метод ConfigureServices регистрирует серверы, которые используются приложением и отвечает за получение строки подключения к базе данных из конфигурационного файла. В методе Configure задаются настройки выдачи окна ошибок, также в данном методе настраивается фильтр запросов, на которые веб-сервер должен отвечать. Все запросы, не попадающие в этот фильтр, будут отклонятся сервером. Исходный код данного класса представлен в листинге Л.1. приложения Л;

– Program – в данном классе создается объект IWebHost в рамках которого развертывается веб приложение. Для создания IWebHost применяется объект IWebHostBuilder. Этот объект создается с помощью метода CreateWebHostBuilder(args). Данный метод выполняет ряд задач, таких как настройка узла, настройка веб-сервиса Kestrel, задание имени хоста и определение главного файла приложения. После это в методе Main происходит создание хоста IWebHost с помощью метода Build(), а затем происходит запуск с помощью метода Run. Исходный код данного класса представлен в листинге К.1. приложения К.

4.2 Программная реализация клиентской части

В основе клиентской части лежит представление MVP. На рисунке 4.1 показана схема взаимодействия частей в MVP.

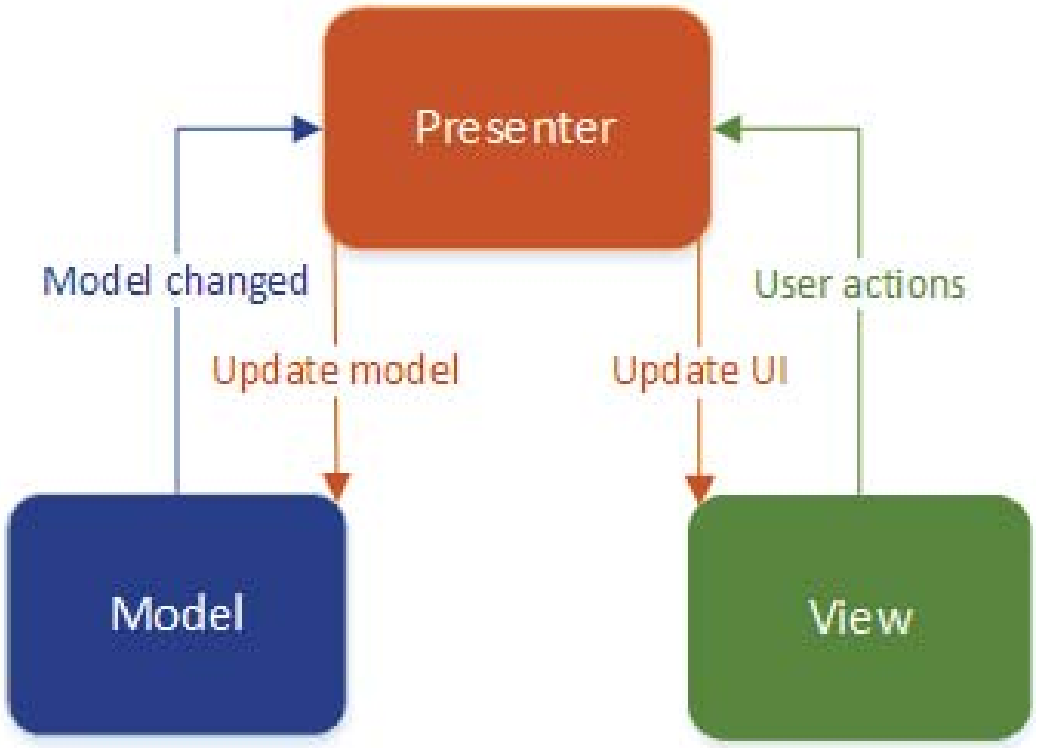

Рисунок 4.1 – Представление MVP

MVP расшифровывается как Model-View-Presenter.

- Модель (Model) — хранит в себе всю бизнес-логику, при необходимости получает данные из хранилища;
- Вид (View) — реализует отображение данных (из Модели) и обращается к Presenter за обновлениями. В качестве View может выступать Activity;
- Представитель (Presenter) — реализует взаимодействие между Моделью и Видом.

Важно помнить, что напрямую View и Model не взаимодействуют, только через Presenter.

В корне приложения содержатся следующие папки:

1. Activities – содержит классы, которые используются для информации. Содержит 2 класса:

– MapActivity – класс для работы с картами. Содержит методы, которые применяются для работы с картами. Исходный код данного класса представлен в листинге М.1. приложения М;

– RootActivity – данный класс содержит общие методы для всех активностей. Исходный код данного класса представлен в листинге Н.1. приложения Н.

В таблице 4.3 представлено описание методов класса MapActivity.

Таблица 4.3 – Описание методов класса MapActivity

Метод	**Описание**
OnCreate	данный метод задает начальную установку параметров при инициализации активности
OnStart	в данном методе происходит инициализация действий видимых пользователю
OnDestroy	в данном методе происходит отчистка всех ресурсов

Продолжение таблицы 4.3

Метод	Описание
OnMapReady	данный метод запускается, когда карта готова к работе
onRequestPermissionsResult	данный метод используется для обработки результатов разрешений приложения
checkPermission	данный метод используется для получения разрешений приложения
initialMap	в данном методе происходит инициализация карты
setInitialFocusOnMap	данный метод используется для фокусировки карты в определенном месте
startUserLocationUpdate	данный метод используется для определения местоположения пользователя
getAnimalCoordinates	метод, который используется для получения координат питомца
showMessage	метод для вывода сообщений

В таблице 4.4 представлено описание методов класса RootActivity.

Таблица 4.4 – Описание методов класса RootActivity

Метод	Описание
onCreate	данный метод задает начальную установку параметров при инициализации активности
showProgress	данный метод отображает индикатор выполнения
hideProgress	данный метод скрывает индикатор выполнения

2. Fragment – используется для обработки информации. Содержит папку:

- Presenters – данная папка содержит классы, которые обрабатывают информацию введенную пользователем. Содержит классы:
  - LoginFragmentPresenter – данный класс содержит методы для обработки вводимой информации о пользователе при авторизации. Исходный код данного класса представлен в листинге С.1. приложения С;
  - RegistrationFragmentPresenter – данный класс содержит методы для обработки вводимой информации о пользователе при регистрации. Исходный код данного класса представлен в листинге Т.1. приложения Т;
  - Presenter – данный класс используется для обработки общих методов. Исходный код данного класса представлен в листинге Р.1. приложения Р.

В таблице 4.5 представлено описание методов класса LoginFragmentPresenter.

Таблица 4.5 – Описание методов класса LoginFragmentPresenter

Метод	**Описание метода**
CheckAuthentication	этот метод используется аутентификации пользователя
redirectToRegistration	этот метод используется для перенаправления пользователя на страницу регистрации, если пользователь не был найден
cancelAuth	этот метод вызывается, если авторизация не была пройдена
Login	этот метод используется для авторизации
onComplete	этот метод вызывается при успешном завершении авторизации

onFail	этот метод вызывается если авторизация завершена с ошибкой
onFailRequest	этот метод вызывается при ошибке подключения к серверу

В таблице 4.6 представлено описание методов класса RegistrationFragmentPresenter.

Таблица 4.6 – Описание методов класса RegistrationFragmentPresenter

Метод	**Описание метода**
Registration	этот метод используется для регистрации
validateFields	этот метод используется для проверки ввода данных и их корректности
onComplete	этот метод вызывается при успешном завершении регистрации
onFail	этот метод вызывается если регистрация завершена с ошибкой
onFailRequest	этот метод вызывается при ошибке подключения к серверу

В таблице 4.7 представлено описание методов класса Presenter.

Таблица 4.7 – Описание методов класса Presenter

Метод	**Описание метода**
startUpdateMap	в данном методе задаются условия для обновления карты
cancelUpdate	этот метод используется при отмене обновления карты
getInstanceTimerTask	метод для получения экземпляра класса TimerTask
updateMap	метод, который используется для обновления карты

Также содержит 3 класса:

- LoginFragment – данный класс используется для обработки информации при авторизации пользователя. Исходный код данного класса представлен в листинге У.1. приложения У;
- RegistrationFragment – данный класс используется для обработки информации при регистрации пользователя. Данный класс содержит метод initialViews, который используется для описания реакции на нажатие на кнопку, методы для получения информации о пользователе и питомце, а также обработчики ошибок. Исходный код данного класса представлен в листинге Ф.1. приложения Ф;
- RootFragment – данный класс используется для обработки общих методов. Содержит метод для перехода на карту. Исходный код данного класса представлен в листинге Х.1. приложения Х.

В таблице 4.8 представлено описание методов класса LoginFragment.

Таблица 4.8 – Описание методов класса LoginFragment

Метод	**Описание метода**
OnCreate	данный метод задает начальную установку параметров при инициализации активности
getEmail	метод для получение email
getPassword	метод для получения пароля
initialViews	данный метод используется для описания реакции на нажатие на кнопку
showProgress	данный метод отображает индикатор выполнения
hideProgress	данный метод скрывает индикатор выполнения

3. Models – папка содержит классы для работы с данными. Содержит папку:
    - Apiclient – используется для обработки запросов. Содержит 4 папки:
        – animal.location – содержит классы UpdateAnimalLocation и UpdateAnimalLocationModel, которые используются для получения логина и идентификатора ошейника, используемых при обновление местоположения. А также класс UpdateAnimalLocationTask, который используется для отправки запросов. Код класса UpdateAnimalLocationTask представлен в листинге Ц.1. приложения Ц;
        – authentication – содержат содержит классы Authenticator и AuthResponse, которые используются для получения данных, используемых для авторизации. А также класс LoginTask, который используется для отправки запросов. Код класса LoginTask представлен в листинге Ц.2. приложения Ц;
        – Enities – содержит существующие сущности. Имеется 2 класса User и Animal. Данные классы содержат методы, которые используются для получения информации о пользователе и питомце;
        – Registration – содержат содержит класс ApiRegistration, который используется для получения данных, используемых для регистрации. А также класс RegistrationTask, который используется для отправки запросов. Код класса RegistrationTask представлен в листинге Ц.3. приложения Ц;

    Также данная папка содержит 3 класса:

– AccountInfo – данный класс содержит методы, которые используются для получения информации о пользователе;
– ApiClient – данный класс создан для связи клиентского приложения с сервером, здесь происходит настройка параметров для подключения к хостингу и реализованы методы, которые необходимы для оправки HTTP запросов к веб-серверу. Код данного класса представлен в листинге Щ.1. приложения Щ;
– AuthResult – данный класс используется для получения результатов авторизации пользователя. Код данного класса представлен в Ш.1. приложения Ш5.

Также содержит 2 класса:

1. Setting – данный класс содержат методы, которые определяют методы для настроек. Код данного класса представлен в листинге Э.1. приложения Э.
2. Validator – данный класс содержит методы для валидации данных, вводимых пользователем (логин, пароль и дата рождения). Код данного класса представлен в листинге Ю.1. приложения Ю.

В таблице 4.9 представлено описание методов класса Setting.

Таблица 4.9 – Описание методов класса Setting

Метод	**Описание метода**
saveAccountInfo	данный метод применяется для сохранения информации о пользователе
getAccessToken	данный метод используется для получения состояния подключения
isLogin	метод, который вызывается при успешной авторизации пользователя

С помощью XML разметки реализован интерфейс страниц авторизации, регистрации и карты клиентского приложения. XML — это расширяемый язык разметки. Расширение XML — это конкретная грамматика, созданная на базе XML и представленная словарём тегов и их атрибутов, а также набором правил, определяющих какие атрибуты и элементы, могут входить в состав других элементов. Данный способ помогает создать удобный интерфейс с хорошо обслуживаемым кодом. Коды XML представлены в приложениях A, B, C, D.

4.3 Аппаратная часть

В первую очередь необходимо подсоединить батарею к 1 и 2 ножке модуля, а PWKEY подсоединить к земле. GSM антенна была подключена в разъём J102, а GNSS антенна в разъём J101. Переключатели S101 и S102 используемые для включения GSM и GPS должны быть переведены в положение ON. Для удобства была припаяна кнопка, которая будет использоваться для включения и выключения устройства. На рисунке 4.2 показан макетный образец трекера. На рисунке М.1. представлена электрическая схема обвязки модуля.

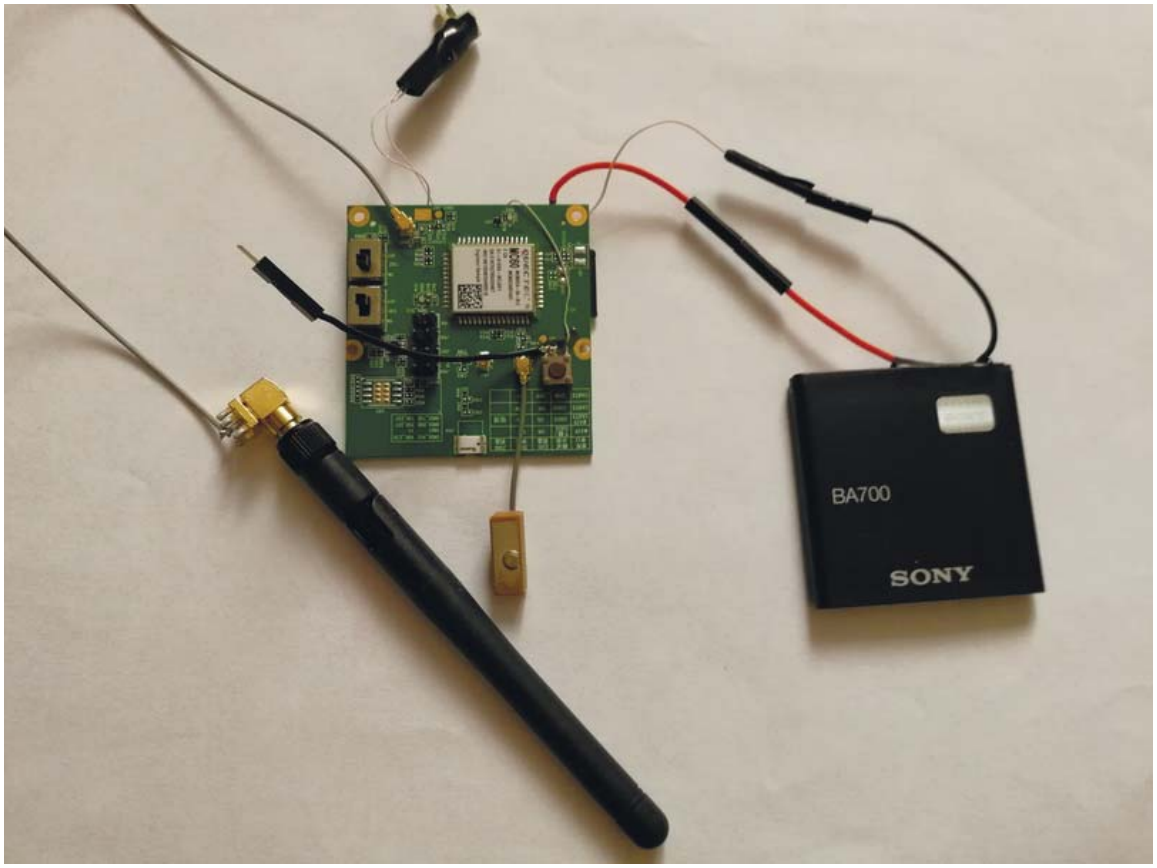

Рисунок 4.2 – Макетный образец трекера.

Также для функционирования GSM части необходимо вставить SIM-карту. На рисунке 4.3 показана обратная сторона модуля, на которой видно карту, а также видно к каким ножкам подсоединены используемые провода.

Рисунок 4.3 – Обратная сторона модуля

Для управления модулем и настройки режима работы будут использоваться АТ-команды. АТ-команды (от англ. attention – «внимание») – это набор команд, которые состоят из коротких текстовых строк, которые объединяются вместе, чтобы сформировать полные команды операций, таких как набор номера, начала соединения, изменения параметров подключения и передачи данных. Для ввода АТ-команд используется терминальная программа.

Необходимо соединить модуль с UART используя TX и RX. На рисунке 4.4 изображена схема подключения модуля к UART.

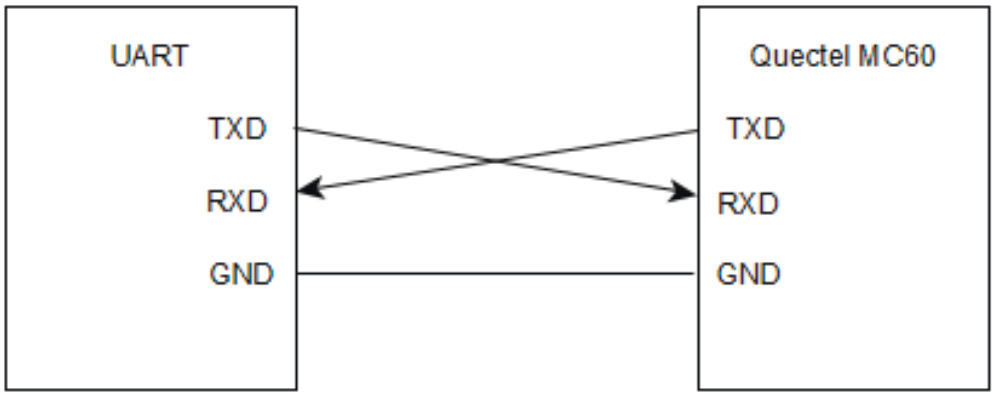

Рисунок 4.4 – Схема подключения модуля

На рисунке 4.5 показано подключение модуля Quectel MC60 к UART.

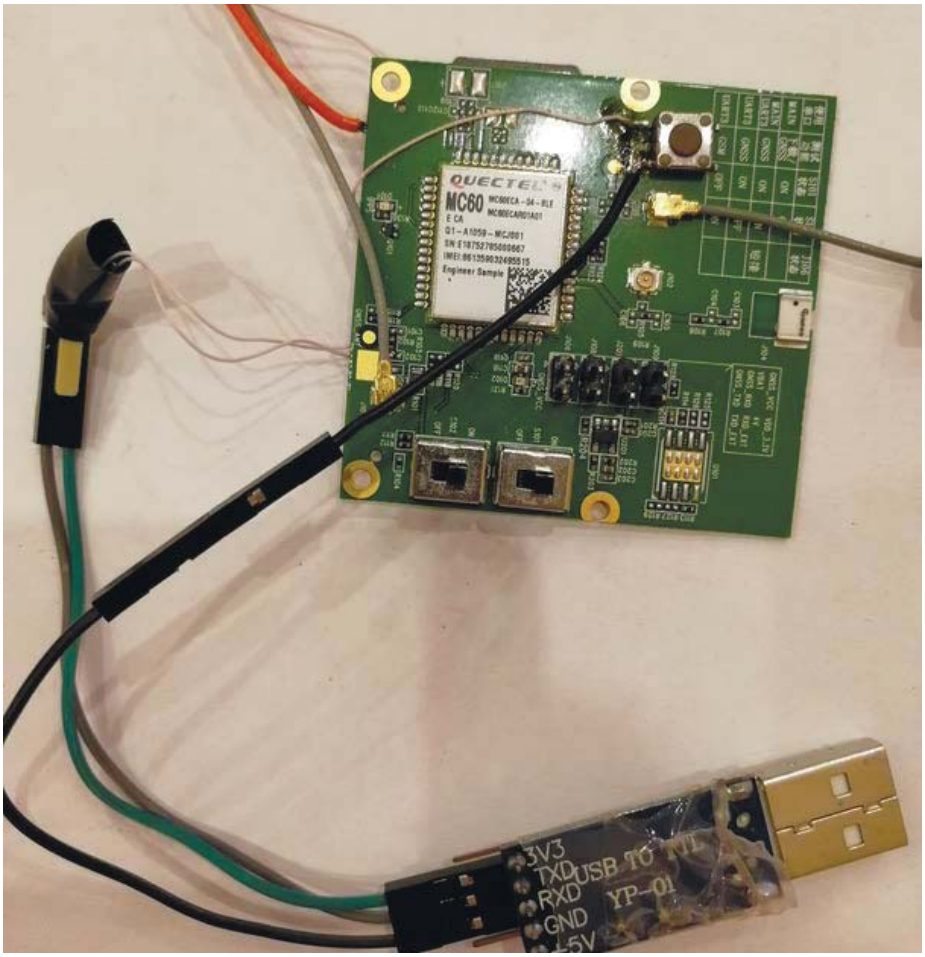

Рисунок 4.5 – Подключение модуля

Для начала необходимо настроить скорость работы модуля, она равна 115200. Далее нужно отправить тестовую команду AT, если модуль ответил

на нее ОК, то он готов к работе. Ниже рассмотрим подробнее основные команды, которые были использованы при работе с данным модулем.

1. Общие команды

– AT+GMI – эта команда вернет три строки. Первая строка – это информация о компании изготовителе, вторая – это идентификатор модуля, а третья содержит версию прошивки;
– AT+GSN – эта команда возвращает IMEI модуля;
– AT+GCAP – эта команда возвращает возможности модуля;
– AT+CCLK – эта команда возвращает текущие дату и время устройства;
– AT+CBC – эта команда используется для мониторинга напряжения питания модуля. Возвращает три параметра. Первый показывает заряжается ли модуль, второй параметр – это уровень заряда, а третий напряжение питания модуля.

Полный перечень команд с полученными результатами представлен в приложении Е.

2. GSM команды модуля

– AT+QSIMDET – эта команда используется для включения режима детектирования наличия SIM-карты. Возвращает три параметра. Первый означает, что режим детектирования карт выключен;
– AT+CPIN=XXXX – при необходимости при помощи этой команды необходимо ввести PIN код при включении модуля;
– AT+ COPS – эта команда, которая выводит информацию об операторе;
– AT+GSN – эта команда используется для получения IMEI;
– AT+CPAS – эта команда содержит информацию о состоянии модуля;

– AT+CREG – эта команда содержит информацию о типе регистрации в сети. Возвращает 2 параметра. Показывают наличие кода регистрации и тип регистрации;
– AT+CSQ – это команда содержит информацию о уровне сигнала;

Также с помощью AT-команд можно звонить и посылать СМС сообщения, рассмотрим основные команды для этого:

– ATD +79xxxxxxx;- это команда применятся для звонка. Номер вводится без пробелов и точек;
– ATA – эта команда используется для ответа на входящий звонок;
– ATH – эта команда используется для отключения вызова;
– AT+CMGS – эта команда используется для отправки СМС сообщений. Необходимо указать номер в кавычках, а после этого ввести тест сообщения. После окончания ввода необходимо ввести ctrl+z для отправки сообщения или esc для отмены;
– AT+CMGR – эта команда используется для чтения входящих СМС сообщений.

В приложении F представлен полный перечень используемых команд и ответов модуля.

3. GNSS команды модуля

– AT+QGNSSC – эта команда используется для управления питанием GNSS;
– AT+QGNSSRD – эта команда используется для получения навигационной информации GNSS;
– AT+QGNSSTS – эта команда используется для получения статуса синхронизации времени для модуля GNSS;
– AT+QGNSSEPO – эта команда используется для включения или отключения функции EPO (A-GPS).
– AT+QGEPOAID – эта команда используется для запуска функции EPO.

Полный перечень команд с полученными результатами представлен в приложении G.

4. GPRS команды модуля

– AT+CGATT – эта команда используется для подключения модуля к GPRS;
– AT+CGPADDR – эта команда возвращает IP адрес контекста PDP (PDP – это протокол пакетных данных);
– AT+QICSGP – эта команда используется для настройки APN (APN — идентификатор сети пакетной передачи данных);
– AT+QIFGCNT – эта команда используется для активации GPRS;
– AT+QGSMLOC – эта команда возвращает 2 параметра. Первый содержит информацию о местоположении устройства, полученную с помощью GSM, а второй – это точные дата и время сети.

Полный перечень команд с полученными результатами представлен в приложении H.

5. Команды для передачи данных

– AT+QIOPEN – данная команда используется для открытия TCP/UDP подключения. На вход подаётся 3 параметра: тип подключения, IP адрес и порт. В ответ возвращаются информация об успешности подключения;
– AT+QISEND – данная команда используется для передачи сообщений. После окончания ввода необходимо ввести ctrl+z для отправки сообщения или esc для отмены;

– AT+QICLOSE – данная команда используется для закрытия TCP/UDP подключения. Полный перечень команд с полученными результатами представлен в приложении I.

4.4 Визуальное представление реализации

1. Форма регистрации пользователя

В форме регистрации пользователя необходимо заполнить все поля. Необходимо указать имя, фамилию, дату рождения, кличку питомца и идентификатор ошейника питомца, а также придумать логин и пароль. Дата рождения и идентификатор устройства должны быть цифровыми значениями. Также необходимо обеспечить подключение телефона к интернету для отправки координат на сервер. Скриншот формы регистрации пользователя представлен на рисунке 4.6.

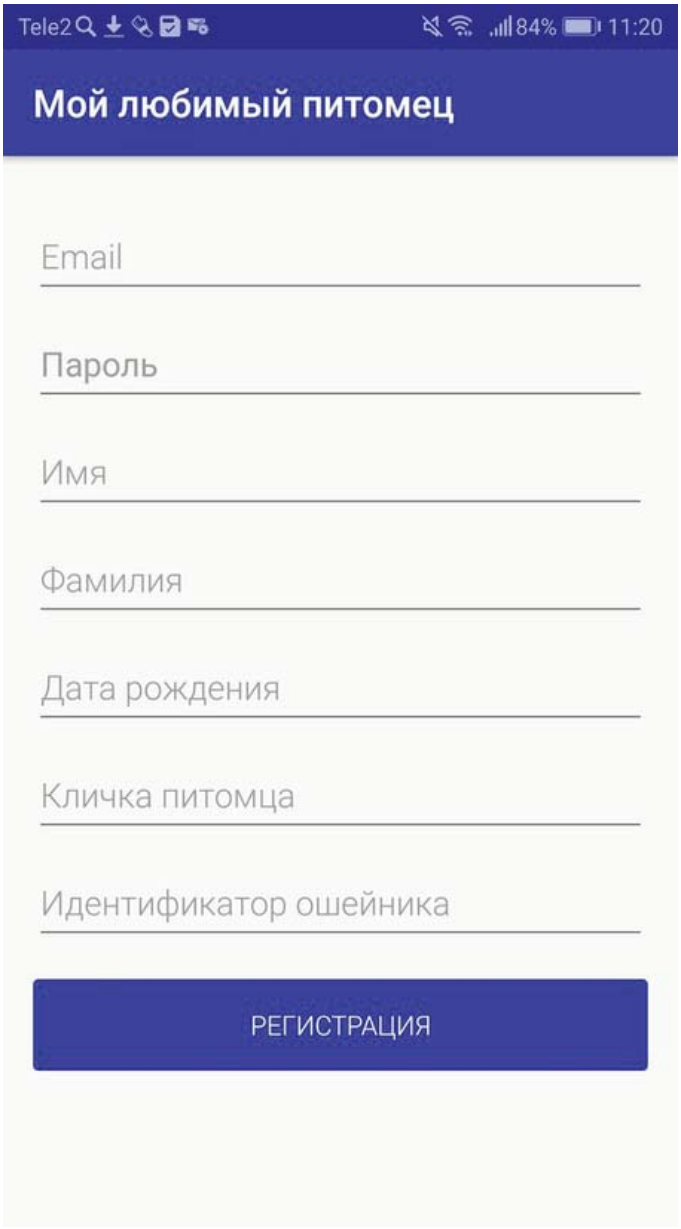

Рисунок 4.6 – Форма регистрация пользователя

2. Форма авторизации пользователя

В форме авторизации пользователя необходимо заполнить все поля. Указать имя пользователя и пароль, которые были использованы раннее при регистрации. Если пользователь не был зарегистрирован необходимо вернуться на страницу регистрации. Также необходимо обеспечить подключения телефона к интернету для сверки данных. Скриншот формы авторизации пользователя представлен на рисунке 4.7.

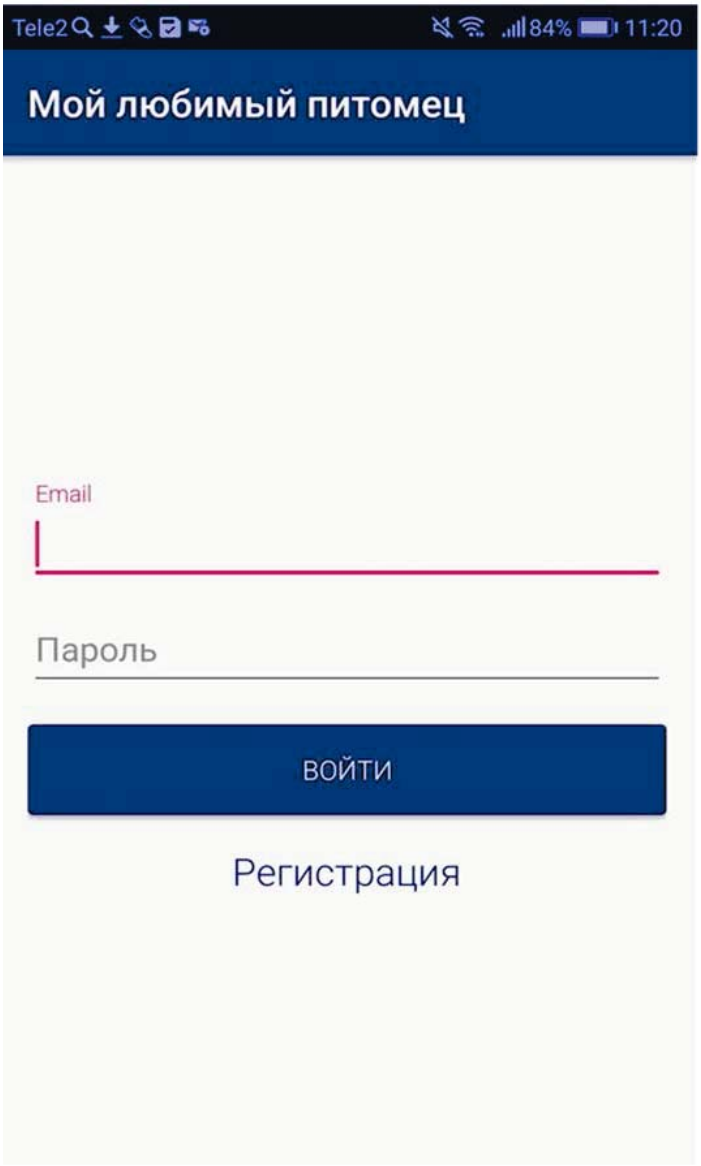

Рисунок 4.7 – Форма авторизации пользователя

3. Карта

После перехода на карту запрашивается разрешение на доступ приложения к данным о местоположение устройства и на карте отображается местоположение пользователя и питомца. На рисунке 4.8 синий маркер отображает местоположение пользователя, а красный местоположение питомца.

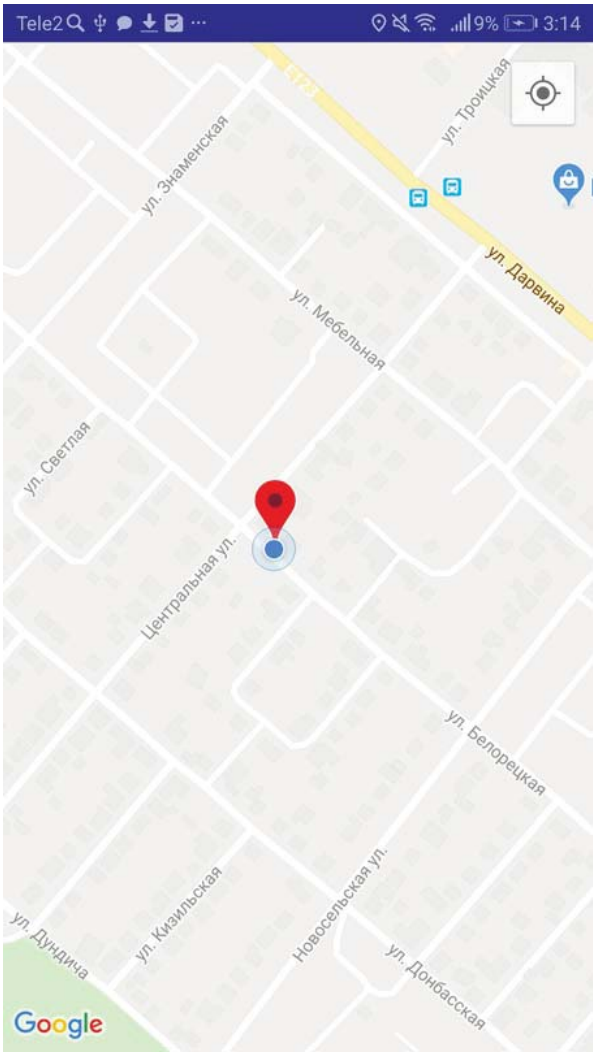

Рисунок 4.8 – Отображение местоположения

5. ТЕСТИРОВАНИЕ

При вводе данных необходимо убедиться в заполнение всех полей. Если какие-то поля не были заполнены, то пользователь видит предупреждение о необходимости заполнения поля и не может пройти регистрацию. На рисунке 5.1 отображено предупреждение.

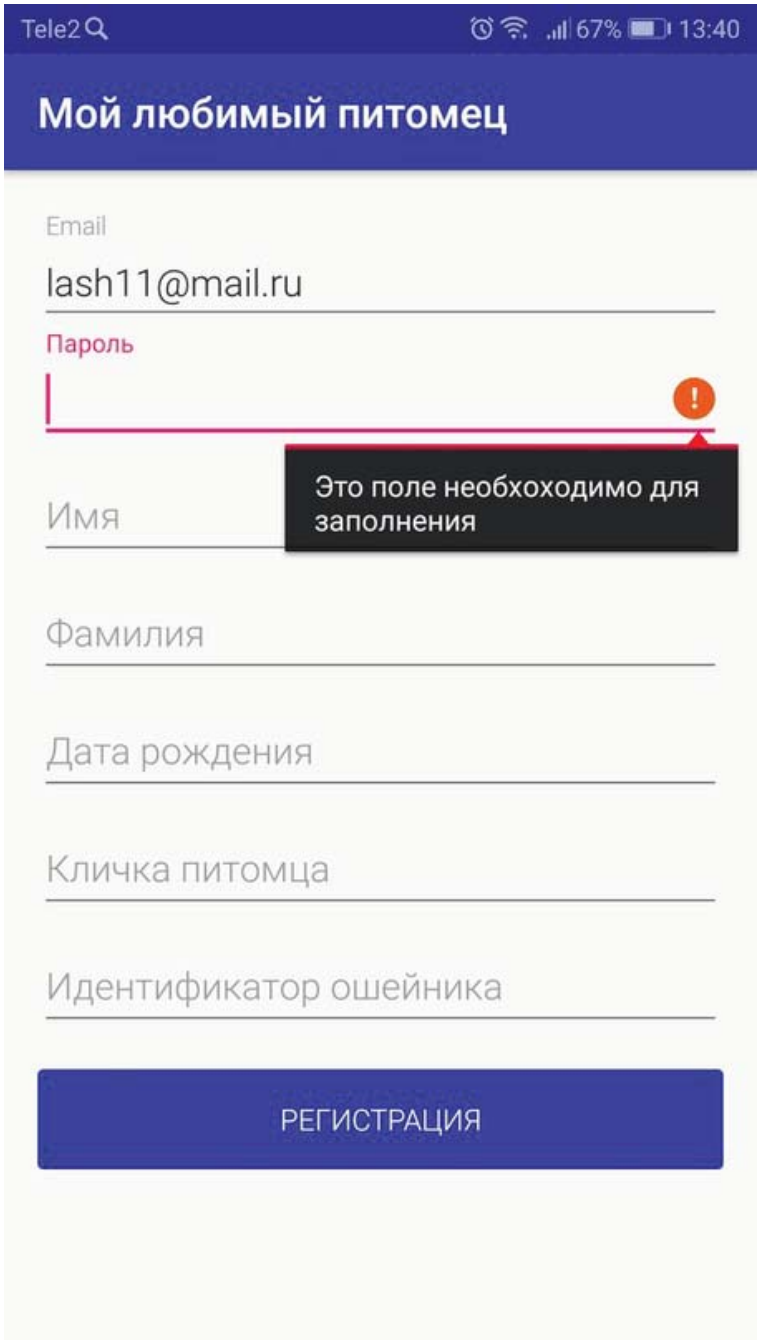

Рисунок 5.1 – Предупреждение о необходимости заполнения всех полей

Идентификатор устройства и дата рождения должны быть числовыми значениями. При вводе отображается только клавиатура. На рисунке 5.2 отображено поле ввода идентификатора.

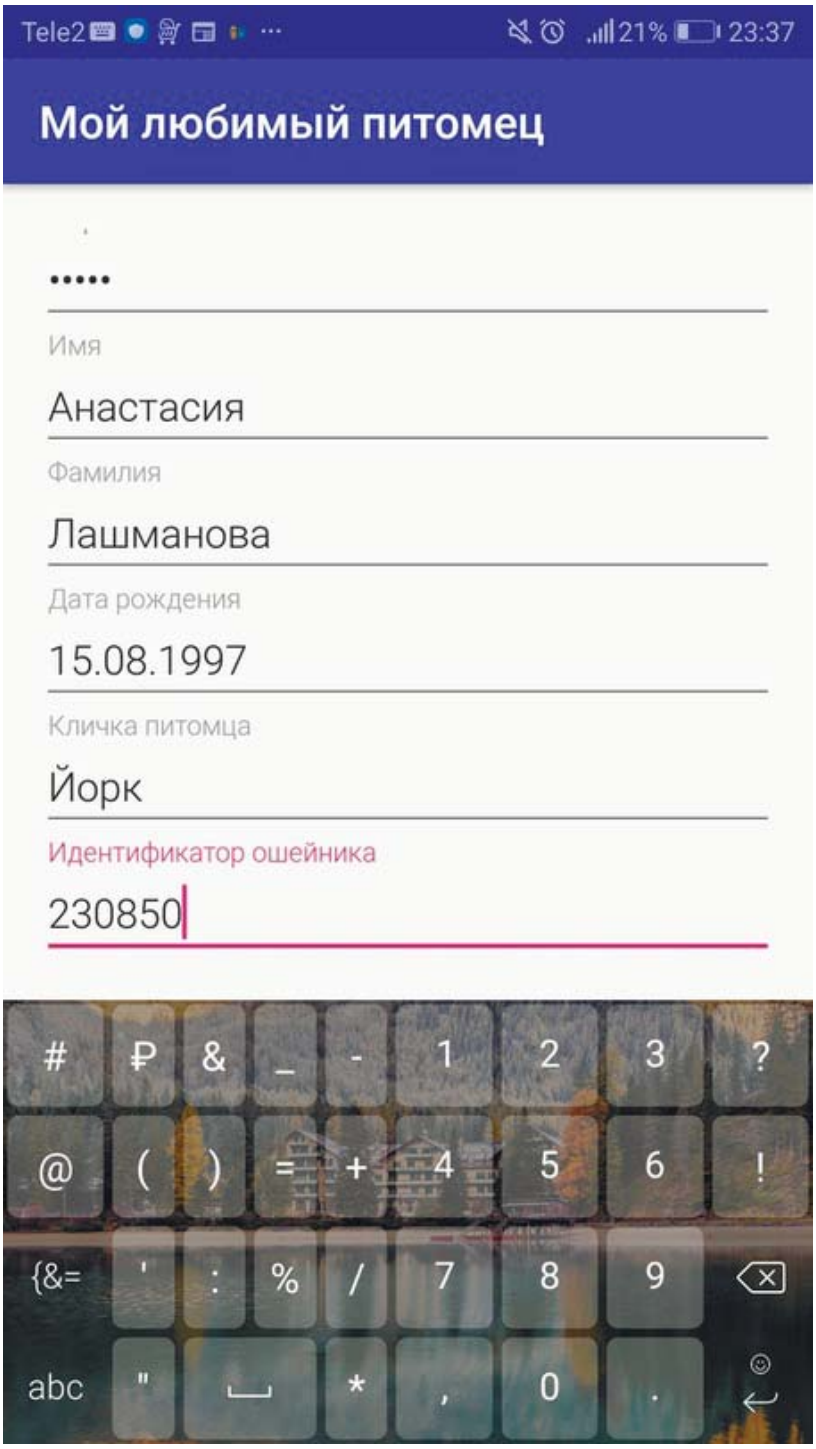

Рисунок 5.2 – Поле ввода идентификатора

После завершения ввода данных необходимо убедиться в подключении телефона к интернету, для отправки приложением данных о регистрации пользователя на сервер. На рисунке 5.3 показан вывод ошибки при отсутствии подключения к сети интернет.

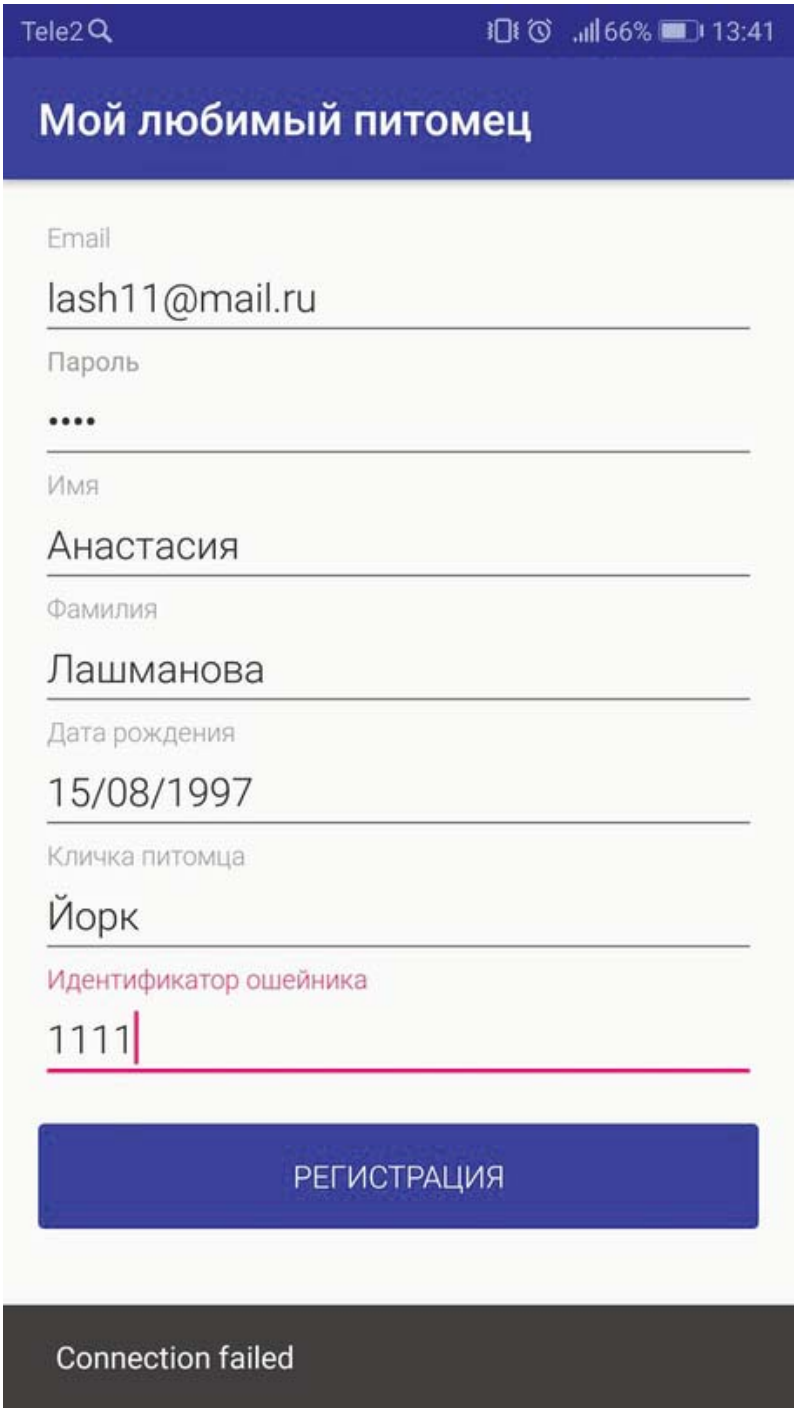

Рисунок 5.3 – Вывод сообщения об ошибке

После отправки данных на сервер необходимо убедиться в правильности добавления данных в бд. На рисунке 5.4 предсталена таблица User, а на рисунке 5.5 таблица Animal.

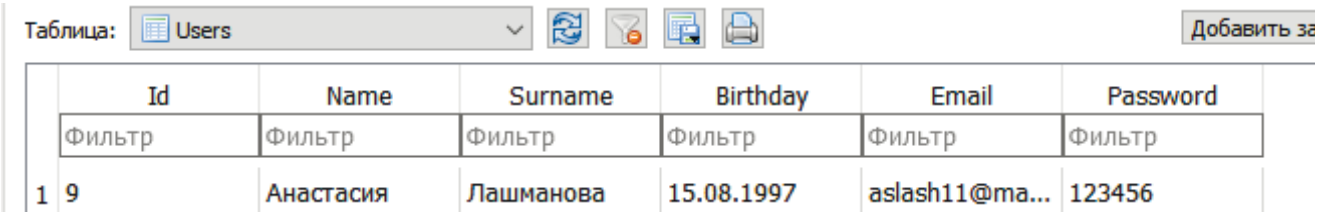
Таблица: Users

Добавить за

	Id	Name	Surname	Birthday	Email	Password
	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	9	Анастасия	Лашманова	15.08.1997	aslash11@ma...	123456

Рисунок 5.4 – Демонстрация добавления данных в таблицу User

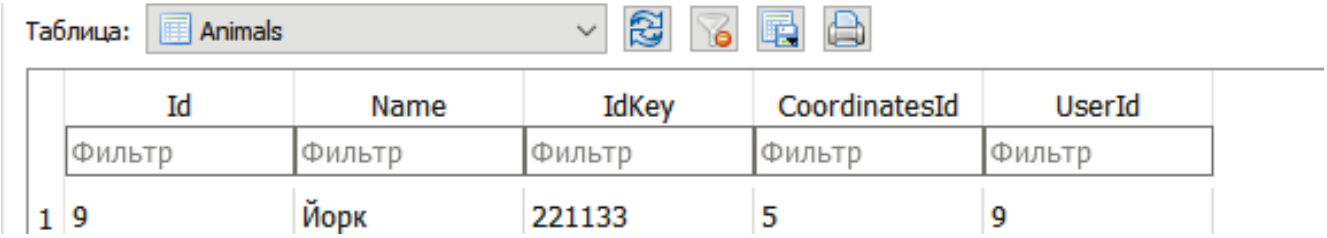
Таблица: Animals

	Id	Name	IdKey	CoordinatesId	UserId
	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	9	Йорк	221133	5	9

Рисунок 5.5– Демонстрация добавления данных в таблицу User

На рисунках видно, что данные о пользователе и питомце были успешно добавленны в таблицу.

Для определения координат устройства используются AT-команды QGREFLOC и QGNSSRD. На листингах 5.1, 5.2 представлены результат выполнения данной команды.

Листинг 5.1 «Определение местоположения по GSM»

```
AT+QGSMLOC=1

+QGSMLOC: 0,61.373521,55.121447,2019/06/07,15:39:33
```

Листинг 5.2 «Определение местоположения по GPS»

```
AT+QGNSSRD?
+QGNSSRD:
$GNRMC,144641.000,A,5507.2813,N,06122.3096,E,0.77,221.28,170619,,,A*75
$GNVTG,221.28,T,,M,0.77,N,1.43,K,A*2E
$GNGGA,144641.000,5507.2813,N,06122.3096,E,1,4,3.16,240.1,M,-11.6,M,,*68
$GPGSA,A,3,15,13,27,,,,,,,,,3.31,3.16,0.98*05
$GLGSA,A,3,68,,,,,,,,,,,3.31,3.16,0.98*14
$GPGSV,3,1,09,20,79,219,,21,66,128,,10,50,237,,27,47,291,17*74
$GPGSV,3,2,09,15,43,071,30,13,21,041,20,08,19,319,,16,14,253,*7C
$GPGSV,3,3,09,30,01,356,*42
```

```
$GLGSV,3,1,10,84,86,211,,69,67,199,,68,55,049,17,85,31,317,*66
$GLGSV,3,2,10,83,27,143,,76,14,339,,77,10,037,,70,10,216,*66
$GLGSV,3,3,10,67,04,039,,75,03,300,*69
$GNGLL,5507.2813,N,06122.3096,E,144641.000,A,A*41
```

Помимо координат были определены точные дата и время сети.После получения данных о местоположении неообходимо отправить эти данные на сервер используя 2 команды: QIOPEN для подключения модуля к серверу и QISEND для отправки запроса на сервер. В листинге 5.3 показан результат выполнения данных команд.

Листинг 5.3 «Отправка сообщени на сервер»

```
AT+QIOPEN="TCP","80.78.248.202",80

OK

CONNECT OK

AT+QISEND

> POST /api/animallocation/put HTTP/1.1

HOST: 80.78.248.202

Accept: application/json

Content-Type: application/json

Content-Length: 104

{"email":"aslash11@mail.ru","idKey":"221133","Coordinates":{"latitude":55.116
226196,"longitude":61.357231140}}
```

Необходимо убедиться в добавлении координат в таблицу, содержащию координаты устройства. На рисунке 5. 6 показана таблица координат.

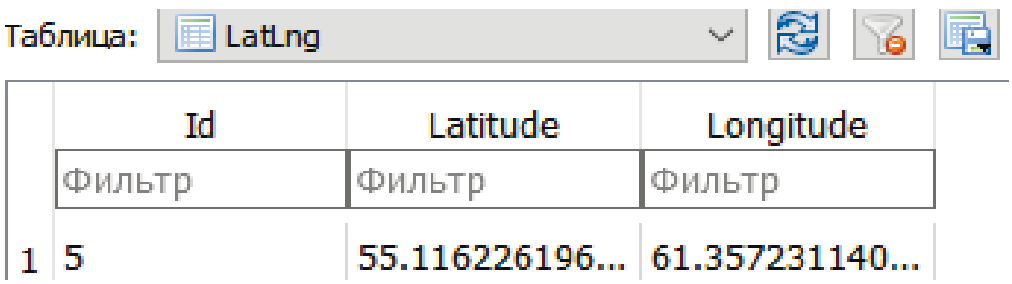

Рисунок 5.6 – Добавление координат в таблицу

После этого необходимо убедиться в правильности отображения местоположения на карте. При переходе на карту запрашивается разрешение на доступ приложения к данным о местоположении устройства. На рисунке 5.7 показано отображение запроса.

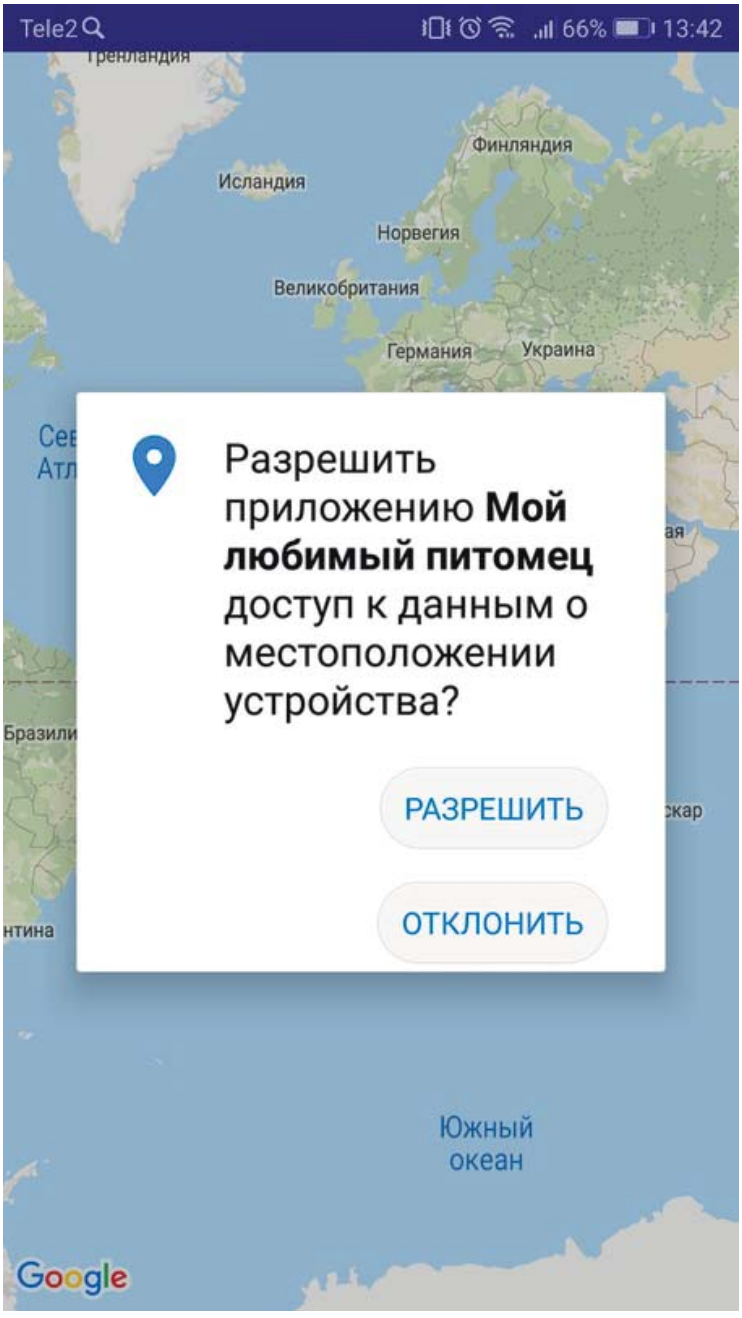

Рисунок 5.7 – Запрос разрешений

На рисунке 5.8 показано отображение местоположения питомца (красная метка) и пользователя (синяя метка).

Рисунок 5.8 – Отображение меток

При нажатии на красную метку Google карты дают возможность построение пути до точки назначение и переноса на карту, где можно посмотреть фотографии окружающей территории.

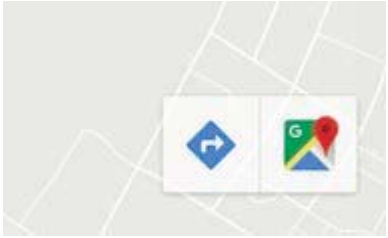

Рисунок 5.9 – Кнопки отображающие дополнительные возможности

На рисунке 5.10 показано посторенние минимального пути до точки назначения и варианты способов наиболее быстрого преодоления этого расстояния.

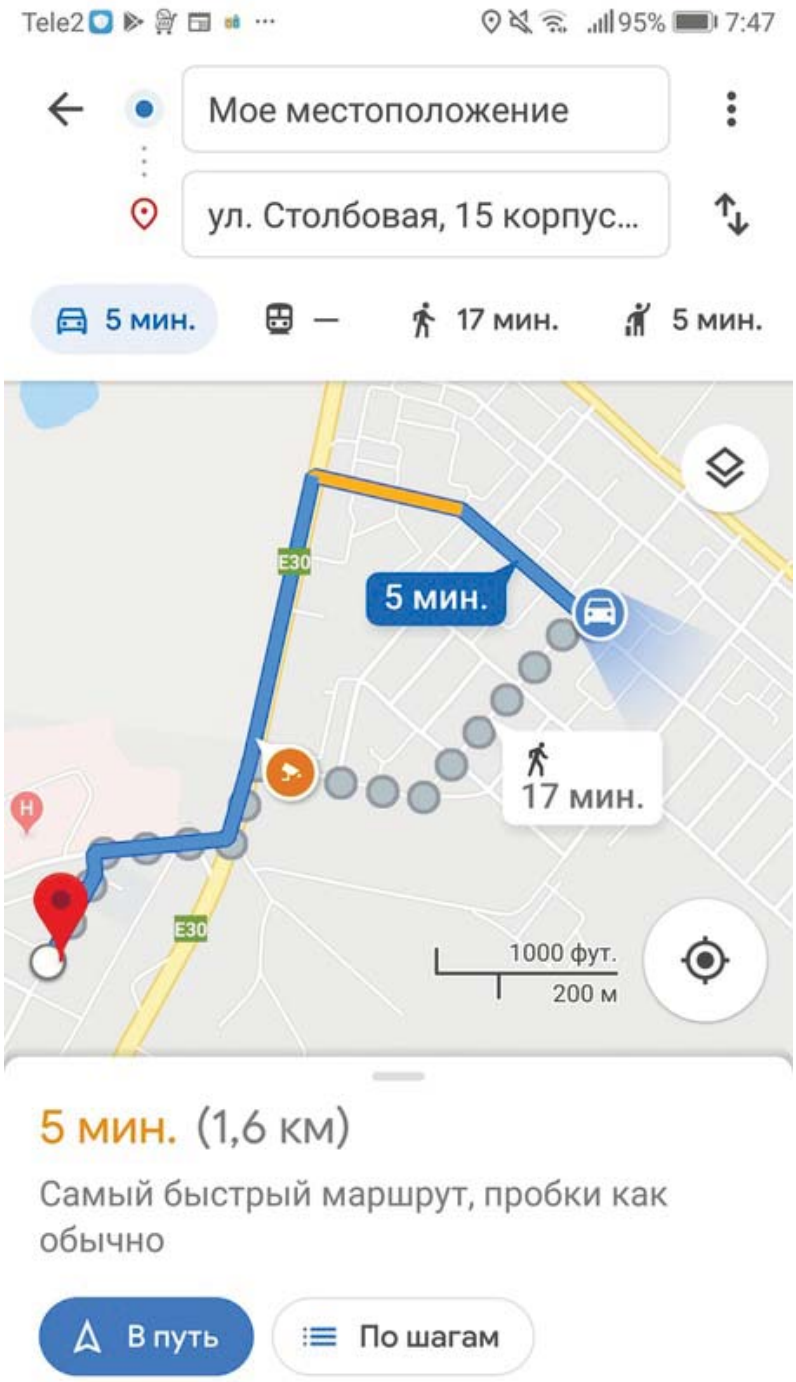

Рисунок 5.10 – Построение минимального пути

На рисунке 5.11 показано отображение фотографий территории вблизи точки местоположения питомца.

Рисунок 5.11 – Окружающая территория

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы было проделано следующее:

- проведен анализ существующих устройств для отслеживания местоположения домашних питомцев, были выявлены их преимущества и недостатки;
- определен набор основных функций и требований к разрабатываемой системе;
- разработана архитектура и функциональный состав системы;
- выбраны средства реализации и платформа для функционирования системы;
- спроектирована структура базы данных;
- разработана серверная часть системы;
- разработана клиентская часть системы;
- произведена сборка макетного образца трекера;
- проведено тестирование работы каждой части отдельно и их взаимодействия.

Результатом выполненной работы является спроектированная система для отслеживания местоположения домашних питомцев.

В настоящее время разработка продолжается, с целью расширения возможностей данной системы. Основные функции, заявленные в данной работе, подготовлены к опытной эксплуатации.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Garmin. – https://www.garmin.ru/portativnye-turisticheskie-navigatory-cat/astro-320-s-osheynikom-dc50.html. Дата обращения: 20 февраля 2019 года.
2. DIGIPULSE. – https://digipulse.ru/gps-treker-s-oshejnikom-dlya-zhivotnyix,-sobak-i-koshek-tk-star-pet-tracker. Дата обращения: 20 февраля 2019 года.
3. Garmin. – https://www.garmin.ru/electroosheynik/catalog/delta/delta-smart/. Дата обращения: 20 февраля 2019 года.
4. Studfiles. – https://studfiles.net/preview/5914493/page:45/. Дата обращения: 5 марта 2019 года.
5. Беспроводные технологии». – https://www.wireless-e.ru/articles/gsm/2010_03_16.php. Дата обращения: 5 марта 2019 года.
6. Habr. – https://habr.com/ru/company/cisco/blog/270779/. Дата обращения: 23 марта 2019 года.
7. Aurora. – http://www.auroramobile.ru/category_15.html. Дата обращения: 25 марта 2019 года.
8. Compress. – https://compress.ru/article.aspx?id=22313. Дата обращения: 25 марта 2019 года.
9. Chip. – https://ichip.ru/prosto-o-slozhnom-2g-3g-4g-i-5g.html. Дата 30 марта 2019 года.
10. Quest. – http://www.i cquest.ru/?section=4&id=85/. Дата обращения: 30 марта 2019 года.
11. Aurora. – http://www.auroramobile.ru/category_13.html. (Дата обращения: 30 марта 2019 года.
12. Itdistrict. – http://itdistrict.ru/li-ion-ili-li-po-v-chem-razlichie-i-chto-vyibrat/. Дата обращения: 30 марта 2019 года.
13. Habr. – https://habr.com/ru/post/149356/. Дата обращения: 13 апреля 2019 года.

14. Питер софт. – https://piter-soft.ru/knowledge/glossary/process/notatsiya-UML.html. Дата обращения: 20 апреля 2019 года.
15. Unetway. – http://unetway.com/tutorial/sqlite-type-data/. Дата обращения: 20 апреля 2019 года.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЕ КОДЫ КОНТРОЛЛЕРОВ

Листинг А.1. – Исходный код “AccountController”

```
using System.Collections.Generic;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ServerAPI.Authenticate;
using ServerAPI.Models;
using ServerAPI.Models.Database;
using static ServerAPI.Models.Database.User;

namespace ServerAPI.Controllers
{
    [Route("api/account")]
    [ApiController]
    public class AccountController : Controller
    {
        private readonly DatabaseContext database;
        public AccountController(DatabaseContext database)
        {
            this.database = database;
        }

        [HttpPost]
        [Route("login")]
        public async Task<IActionResult> Login([FromBody] LoginModel model) // метод
используется, для обработки информации о пользователе при авторизации
        {
            User user = await database.Users
                .Include(u => u.Animals)
                    .ThenInclude(u => u.Coordinates)
                .FirstOrDefaultAsync(u => u.Email.Equals(model.Email) &&
u.Password.Equals(model.Password));

            if (user is null) return StatusCode(400, "incorrect email or password");

            ClaimsIdentity identity = GetIdentity(model.Email, model.Password);

            string accessToken = TokenGenerator.GetToken(identity);
            AccountInfo accountInfo = new AccountInfo
            {
                Name = user.Name,
                Surname = user.Surname,
                AccessToken = accessToken,
                Animals = user.Animals
            };

            return Ok(accountInfo);
        }

        [HttpPost]
        [Route("registration")] // этот метод используется, для обработки информации
о пользователе при регистрации
        public IActionResult Registration([FromBody] RegistrationModel model)
        {
```

Продолжение листинга А.1

```
         if (model is null) return StatusCode(204, "data is empty");

            User user = model.ToUser();
            if (database.Users.ToList().Contains(user, new UserComparer()))
                return StatusCode(400, new { Message = "user contains in database"
});

            database.Users.Add(user);
            database.SaveChanges();

            ClaimsIdentity identity = GetIdentity(model.Email, model.Password);
            string accessToken = TokenGenerator.GetToken(identity);

            AccountInfo accountInfo = new AccountInfo
            {
                Name = model.Name,
                Surname = model.Surname,
                AccessToken = accessToken,
                Animals = model.Animals
            };

            return Ok(accountInfo);
        }

        private ClaimsIdentity GetIdentity(string email, string password) // метод
вызывается если пользователь был найден, то создается объект ClaimsIdentity
        {
            List<Claim> claims = new List<Claim>
            {
                new Claim(ClaimsIdentity.DefaultNameClaimType, email)
            };

           return new ClaimsIdentity(claims, "Token",
ClaimsIdentity.DefaultNameClaimType,
                ClaimsIdentity.DefaultRoleClaimType);
        }
    }
}
```

Листинг А.2. – Исходный код “AnimalLocationController”

```
using System;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ServerAPI.Models.AnimalLocation;
using ServerAPI.Models.Database;

namespace ServerAPI.Controllers
{
    [Route("api/animallocation")]
    [ApiController]
    public class AnimalLocationController : ControllerBase
```

Продолжение листинга А.2

```
    {
        private readonly DatabaseContext database;

        public AnimalLocationController(DatabaseContext database) => this.database =
database;

        [HttpPost]
        [Route("put")]
        public async Task<IActionResult> Put([FromBody] ModelForPutAnimalLocation
model) //отправка запроса на получение координат питомца

            if (model is null) return BadRequest("data is empty");

            Animal animal = await FindAnimal(model.Email, model.IdKey);

            if (animal is null) return BadRequest("animal not found");

            if (animal.Coordinates is null)
                animal.Coordinates = model.Coordinates;
            else
                animal.Coordinates.Update(model.Coordinates);

            database
                .Animals
                .Update(animal);

            await database
                .SaveChangesAsync();

            return Ok();
        }

        [HttpPost]
        [Route("get")]
        public async Task<IActionResult> Get([FromBody] ModelForGettingAnimalLocation
model) // получение координат питомца
        {
            Animal animal = await FindAnimal(model.Email, model.IdKey);

            if (animal is null) return BadRequest("animal not found");

            if (animal.Coordinates is null) BadRequest("location of animal is null");

return Ok(animal.Coordinates);

        }

        private async Task<Animal> FindAnimal(string email, string idKey)
//определение местоположения питомца
        {
            User user = await database
                .Users
                .Include(u => u.Animals)
                    .ThenInclude(a => a.Coordinates)
                .FirstOrDefaultAsync(u => u.Email.Equals(email));

            if (user is null) return null;
```

Продолжение листинга А.2

```
            return user
                .Animals
                .Find(a => a.IdKey.Equals(idKey));
        }
    }
}
```

ПРИЛОЖЕНИЕ Б
ИСХОДНЫЕ КОДЫ ANIMALLOCATION

Листинг Б.1. – Исходный код “ModelForGettingAnimalLocation”

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

namespace ServerAPI.Models.AnimalLocation
{
    public class ModelForGettingAnimalLocation
    {
        public string Email { get; set; }
        public string IdKey { get; set; } //идентификатор ошейника
    }
}
```

Листинг Б.2. – Исходный код “ModelForPutAnimalLocation”

```
using ServerAPI.Models.Database;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ServerAPI.Models.AnimalLocation
{
    public class ModelForPutAnimalLocation
    {
        public string Email { get; set; }
        public string IdKey { get; set; } // идентификатор ошейника
        public LatLng Coordinates { get; set; } //координаты
    }
}
```

ПРИЛОЖЕНИЕ В

Листинг В.1. – Исходный код “DatabaseContext”

```
using Microsoft.EntityFrameworkCore;

namespace ServerAPI.Models.Database
{
    public class DatabaseContext:DbContext //определяет контекст данных, для
взаимодействия с базой данных

    {
        public DatabaseContext(DbContextOptions<DatabaseContext> options) :
base(options) { }

        public DbSet<User> Users { get; set; }
        public DbSet<Animal> Animals { get; set; }
    }
}
```

ПРИЛОЖЕНИЕ Г

Листинг Г.1. – Исходный код “User”

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

namespace ServerAPI.Models.Database
{
    public class User
    {
//данные поля описывают информацию о пользователе
        public int Id { get; set; }
        public string Name { get; set; }
        public string Surname { get; set; }
        public string Birthday { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }
        public List<Animal> Animals { get; set; }

        public User()
        {
            Id = 0;
            Name = Surname = Email = Password = Birthday = string.Empty;
            Animals = new List<Animal>();
        }

        public class UserComparer : IEqualityComparer<User> //применяется для
сравнения пользователей
        {
            public bool Equals(User x, User y)
            {
                if (ReferenceEquals(x, y)) return true;

                if (ReferenceEquals(x, null) || ReferenceEquals(y, null)) return
false;

                try
                {
                    return (x.Id == y.Id && x.Name == y.Name && x.Surname ==
y.Surname &&
                    x.Birthday == y.Birthday && x.Email == y.Email && x.Password ==
y.Password &&
                    x.Animals.SequenceEqual(y.Animals));
                }
                catch { return false; }
            }

            public int GetHashCode(User obj)
            {
                throw new NotImplementedException();
            }
        }
    }
}
```

ПРИЛОЖЕНИЕ Д

Листинг Д.1. – Исходный код “AccountInfo”

```
using ServerAPI.Models.Database;
using System.Collections.Generic;

namespace ServerAPI.Models
{
    public class AccountInfo //аккаунт пользователя
    {
        public string Name { get; set; }
        public string Surname { get; set; }
        public string AccessToken { get; set; }
        public List<Animal> Animals { get; set; }
    }
}
```

ПРИЛОЖЕНИЕ Е

Листинг Е.1. – Исходный код “ RegistrationModel”

```
using ServerAPI.Models.Database;
using System.Collections.Generic;

namespace ServerAPI.Models
{
    public class RegistrationModel
    {
//данные необходимые при регистрации
        public string Name { get; set; }
        public string Surname { get; set; }
        public string Birthday { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }
        public List<Animal> Animals { get; set; }

        public User ToUser() => //добавление пользователя в базу данных
            new User
            {
                Name     = Name,
                Surname  = Surname,
                Birthday = Birthday,
                Email    = Email,
                Password = Password,
                Animals  = Animals
            };
    }
}
```

Листинг Е.2. – Исходный код “ LoginModel”

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

namespace ServerAPI.Models
{
    public class LoginModel
    {
        public string Email { get; set; }
        public string Password { get; set; }
    }
}
```

ПРИЛОЖЕНИЕ Ж

Листинг Ж.1. – Исходный код Migration

```
using Microsoft.EntityFrameworkCore.Migrations;

namespace ServerAPI.Migrations
{
    public partial class Initial : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "LatLng",
                columns: table => new
                {
                    Id = table.Column<int>(nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    Latitude = table.Column<float>(nullable: false),
                    Longitude = table.Column<float>(nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_LatLng", x => x.Id);
                });

            migrationBuilder.CreateTable(
                name: "Users",
                columns: table => new
                {
                    Id = table.Column<int>(nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    Name = table.Column<string>(nullable: true),
                    Surname = table.Column<string>(nullable: true),
                    Birthday = table.Column<string>(nullable: true),
                    Email = table.Column<string>(nullable: true),
                    Password = table.Column<string>(nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Users", x => x.Id);
                });
            migrationBuilder.CreateTable(
                name: "Animals",
                columns: table => new
                {
                    Id = table.Column<int>(nullable: false)
                        .Annotation("Sqlite:Autoincrement", true),
                    Name = table.Column<string>(nullable: true),
                    IdKey = table.Column<string>(nullable: true),
                    CoordinatesId = table.Column<int>(nullable: true),
                    UserId = table.Column<int>(nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Animals", x => x.Id);
                    table.ForeignKey(
                        name: "FK_Animals_LatLng_CoordinatesId",
                        column: x => x.CoordinatesId,
                        principalTable: "LatLng",
```

Продолжение листинга Ж.1.

```
                    principalColumn: "Id",
                    onDelete: ReferentialAction.Restrict);
                table.ForeignKey(

                    name: "FK_Animals_Users_UserId",
                    column: x => x.UserId,
                    principalTable: "Users",
                    principalColumn: "Id",
                    onDelete: ReferentialAction.Restrict);
            });

        migrationBuilder.CreateIndex(
            name: "IX_Animals_CoordinatesId",
            table: "Animals",
            column: "CoordinatesId");

        migrationBuilder.CreateIndex(
            name: "IX_Animals_UserId",
            table: "Animals",
            column: "UserId");
    }

    protected override void Down(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.DropTable(
            name: "Animals");

        migrationBuilder.DropTable(
            name: "LatLng");

        migrationBuilder.DropTable(
            name: "Users");
    }
  }
}
```

Листинг Ж.2. – Исходный код Migration

```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Infrastructure;
using Microsoft.EntityFrameworkCore.Storage.ValueConversion;
using ServerAPI.Models.Database;
namespace ServerAPI.Migrations
{
    [DbContext(typeof(DatabaseContext))]
    partial class DatabaseContextModelSnapshot : ModelSnapshot
    {
        protected override void BuildModel(ModelBuilder modelBuilder)
        {
#pragma warning disable 612, 618
            modelBuilder
                .HasAnnotation("ProductVersion", "2.1.4-rtm-31024");
            modelBuilder.Entity("ServerAPI.Models.Database.Animal", b =>
                {
```

Продолжение листинга Ж.2.

```
            b.Property<int>("Id")
                .ValueGeneratedOnAdd();

            b.Property<int?>("CoordinatesId");

            b.Property<string>("IdKey");

            b.Property<string>("Name");

            b.Property<int?>("UserId");

            b.HasKey("Id");

            b.HasIndex("CoordinatesId");

            b.HasIndex("UserId");

            b.ToTable("Animals");
        });

    modelBuilder.Entity("ServerAPI.Models.Database.LatLng", b =>
        {
            b.Property<int>("Id")
                .ValueGeneratedOnAdd();

            b.Property<float>("Latitude");

            b.Property<float>("Longitude");

            b.HasKey("Id");

            b.ToTable("LatLng");
        });
    modelBuilder.Entity("ServerAPI.Models.Database.User", b =>
        {
            b.Property<int>("Id")
                .ValueGeneratedOnAdd();

            b.Property<string>("Birthday");

            b.Property<string>("Email");

            b.Property<string>("Name");

            b.Property<string>("Password");

            b.Property<string>("Surname");

            b.HasKey("Id");

            b.ToTable("Users");
        });

    modelBuilder.Entity("ServerAPI.Models.Database.Animal", b =>
        {
            b.HasOne("ServerAPI.Models.Database.LatLng", "Coordinates")
```

Продолжение листинга Ж.2.

```
                        .WithMany()
                        .HasForeignKey("CoordinatesId");

                    b.HasOne("ServerAPI.Models.Database.User")
                        .WithMany("Animals")
                        .HasForeignKey("UserId");
                });
#pragma warning restore 612, 618
        }
    }
}
```

ПРИЛОЖЕНИЕ Й
ИСХОДНЫЕ КОДЫ AUTHENTICATE

Листинг Й.1. – Исходный код “AuthOptions”

```
using Microsoft.IdentityModel.Tokens;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ServerAPI.Authenticate
{
    public class AuthOptions
    {
        public const string ISSUER = "ServerAPI"; //издатель токена
        public const string AUDIENCE = "ServerAPI"; //потребитель токена
        public const int LIFETIME = 3; //время жизни токена

        private const string PRIVATE_KEY = "6oqFQOL5y5CA8pPnVk2NdBH4e0I0sA"; //ключ
        public static SymmetricSecurityKey SecurityKey =>
            new SymmetricSecurityKey(Encoding.ASCII.GetBytes(PRIVATE_KEY));
    }
```

Листинг Й.2. – Исходный код “TokenGenerator”

```
using Microsoft.IdentityModel.Tokens;
using System;
using System.Collections.Generic;
using System.IdentityModel.Tokens.Jwt;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;

namespace ServerAPI.Authenticate
{
    public static class TokenGenerator
    {
        public static string GetToken(ClaimsIdentity claims) //данный метод
генерирует токен
        {
            DateTime dateNow = DateTime.Now;

            JwtSecurityToken token = new JwtSecurityToken(
                issuer: AuthOptions.ISSUER,
                audience: AuthOptions.AUDIENCE,
                notBefore: dateNow,
                expires: dateNow.Add(TimeSpan.FromHours(AuthOptions.LIFETIME)),
                claims: claims.Claims,
                signingCredentials: new SigningCredentials(AuthOptions.SecurityKey,
SecurityAlgorithms.HmacSha256));

            return new JwtSecurityTokenHandler().WriteToken(token);
        }
    }
}
```

ПРИЛОЖЕНИЕ К

Листинг К.1. – Исходный код “ Program”

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;

namespace ServerAPI
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateWebHostBuilder(args).Build().Run();
        }

        public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .UseKestrel() // настраиваем веб-сервер Kestrel
                .UseUrls("http://192.168.0.104")
                .UseStartup<Startup>(); // устанавливаем главный файл приложения
    }
}
```

ПРИЛОЖЕНИЕ Л

Листинг Л.1. – Исходный код “Startup”

```
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using ServerAPI.Authenticate;
using ServerAPI.Models.Database;

namespace ServerAPI
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }
        public IConfiguration Configuration { get; }


        public void ConfigureServices(IServiceCollection services)//регистрация служб
приложения
        {
            services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
                .AddJwtBearer(authOptions =>
                {
                    authOptions.RequireHttpsMetadata = false;
                    authOptions.TokenValidationParameters = new
Microsoft.IdentityModel.Tokens.TokenValidationParameters
                    {
                        ValidateIssuer = true,
                        ValidateAudience = true,
                        ValidateLifetime = true,
                        ValidateIssuerSigningKey = true,

                        ValidIssuer = AuthOptions.ISSUER,
                        ValidAudience = AuthOptions.AUDIENCE,
                        IssuerSigningKey = AuthOptions.SecurityKey
                    };
                });
            string databaseConnetion =
Configuration.GetConnectionString("DatabaseConnection");
            services.AddDbContext<DatabaseContext>(options =>
options.UseSqlite(databaseConnetion));


services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1)
                .AddJsonOptions(jsonOptions =>
jsonOptions.SerializerSettings.ReferenceLoopHandling =
Newtonsoft.Json.ReferenceLoopHandling.Ignore);
        }
        public void Configure(IApplicationBuilder app, IHostingEnvironment
env)//устанавливает, как приложение будет обрабатывать запрос
        {
```

Продолжение листинга Л.1.

```
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseHsts();
            }

            app.UseAuthentication();
            app.UseCors(corsOptions =>
            {
                corsOptions.AllowAnyHeader();
                corsOptions.AllowAnyMethod();
                corsOptions.AllowAnyOrigin();
                corsOptions.AllowCredentials();
            });
            app.UseMvc();
        }
    }
}
```

ПРИЛОЖЕНИЕ М

Листинг М.1. – Исходный код “ MapActivity”

```
package susu.com.animallocation.activities;

import android.Manifest;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.content.pm.PackageManager;
import android.os.IBinder;
import android.support.annotation.NonNull;
import android.support.design.widget.Snackbar;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import android.support.v4.content.ContextCompat;

import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationCallback;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationResult;
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.Marker;
import com.google.android.gms.maps.model.MarkerOptions;

import susu.com.animallocation.R;
import susu.com.animallocation.models.Settings;
import susu.com.animallocation.fragment.presenters.MapPresenter.Presenter;
import
susu.com.animallocation.services.UpdateAnimalLocation.UpdateAnimalLocationService;

public class MapActivity extends FragmentActivity implements OnMapReadyCallback {

    private UpdateAnimalLocationService.ServiceBinder serviceBinder;
    private boolean isBindService;

    private GoogleMap mMap;
    private Presenter presenter;

    private Settings settings;
    private static final int LOCATION_PERMISSION_CODE = 1001;

    @Override
    protected void onCreate(Bundle savedInstanceState) { //начальная установка
параметров
        super.onCreate(savedInstanceState);
        setContentView(R.layout.map_activity);
        presenter = new Presenter(this);
        settings = new Settings(this);

        if(checkPermissions())
```

Продолжение листинга М.1.

```
      initialMap();

    }

    @Override
    protected void onStart() { // инициализация действий
        super.onStart();

        Intent serviceIntent = new Intent(this, UpdateAnimalLocationService.class);


        String accessToken = settings.getAccessToken();

        serviceIntent
                .putExtra("EMAIL", email)
                .putExtra("ID_KEY", idKey)
                .putExtra("TOKEN", accessToken);

        bindService(serviceIntent, serviceConnection, Context.BIND_AUTO_CREATE);
    }

    @Override
    protected void onDestroy() { // отчистка всех ресурсов
        super.onDestroy();
        if(isBindService) {
            unbindService(serviceConnection);
            isBindService = false;
        }
    }

    @Override
    public void onMapReady(GoogleMap googleMap) { //готовность карты
        mMap = googleMap;
        setInitialFocusOnMap();
        startUserLocationUpdate();
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, //запрос разрешений
                                           @NonNull String[] permissions,
                                           @NonNull int[] grantResults) {
        switch (requestCode) {
            case LOCATION_PERMISSION_CODE:
                if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                    initialMap();
                }
                else checkPermissions();
                break;
        }
    }

    private boolean checkPermissions() {
        if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                != PackageManager.PERMISSION_GRANTED) {
```

Продолжение листинга М.1.

```
            ActivityCompat.requestPermissions(this,
                    new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                    LOCATION_PERMISSION_CODE);

            return false;
        }



         return true;
    }

    private void initialMap() { //инициализация карты
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    private void setInitialFocusOnMap() { //фокусировка карты
        LatLng chelyabinsk = new LatLng(55.0914, 61.2544);
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(chelyabinsk, 12));
    }

    @SuppressWarnings("MissingPermission")
    private void startUserLocationUpdate() { //определение местоположения
пользовалеля
        FusedLocationProviderClient locationProviderClient
                = new FusedLocationProviderClient(this);

        LocationRequest locationRequest = new LocationRequest();
        locationRequest.setInterval(2000);
        locationRequest.setFastestInterval(1000);
        locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

        LocationCallback locationCallback = new LocationCallback() {
            @Override
            public void onLocationResult(LocationResult locationResult) {
                if (locationResult == null) {
                    return;
                }
            }
        };

        locationProviderClient.getLastLocation();
        mMap.setMyLocationEnabled(true);
        locationProviderClient.requestLocationUpdates(locationRequest,
locationCallback,
                null);
    }
    private ServiceConnection serviceConnection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName componentName, IBinder
serviceBinder) {
            MapActivity.this.serviceBinder = (
                    UpdateAnimalLocationService.ServiceBinder) serviceBinder;
            isBindService = true;
```

Продолжение листинга М.1.

```
            presenter.startUpdateMap();
        }
        @Override
        public void onServiceDisconnected(ComponentName componentName) {
            isBindService = false;
        }
    };
    private Marker mapMarker;
//работа с маркерами
    public void setMarker(LatLng coordinates) {
        if(mMap == null) return;
        mapMarker = mMap.addMarker(new MarkerOptions().position(coordinates));
    }

    public void deleteMarker() {
        if(mMap == null) return;
        mapMarker.remove();
    }

    public Marker getMapMarker() {
        return mapMarker;
    }

    public LatLng getAnimalCoordinates() {
        if(serviceBinder == null &&
                !serviceBinder.isSuccessful() && isBindService) return null;
        return serviceBinder.getAnimalCoordinates();
    }

    public void showMessage(String message) {
        Snackbar.make(getCurrentFocus(), message, Snackbar.LENGTH_SHORT);
    }
}
```

ПРИЛОЖЕНИЕ Н

Листинг Н.1. – Исходный код “ RootActivity”

```
package susu.com.animallocation.activities;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AppCompatActivity;
import android.widget.ProgressBar;

import susu.com.animallocation.R;
import susu.com.animallocation.fragment.LoginFragment;

import static android.view.View.*;

public class RootActivity extends AppCompatActivity {

    ProgressBar progressBar;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) { // установка
параметров
        super.onCreate(savedInstanceState);
        setContentView(R.layout.root_activity);

        progressBar = findViewById(R.id.progress);

        getSupportFragmentManager()
                .beginTransaction()
                .add(R.id.fragment_container, new LoginFragment())
                .commit();
    }

    public void showProgress() { //отображение индикатора выполнения
        if(progressBar.getVisibility() != VISIBLE)
            progressBar.setVisibility(VISIBLE);
    }

    public void hideProgress() { //скрытие индикатора выполнения
        if(progressBar.getVisibility() != INVISIBLE) {
            progressBar.setVisibility(INVISIBLE);
        }
    }
}
```

ПРИЛОЖЕНИЕ П

Листинг М.1. – Исходный код “ SuperActivity”

```
package susu.com.animallocation.activities;

import android.app.ProgressDialog;
import android.content.Context;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.view.inputmethod.InputMethodManager;

public class SuperActivity extends AppCompatActivity {

    public final String EMAIL_KEY    = "EMAIL";
    public final String PASSWORD_KEY = "PASSWORD";

    protected ProgressDialog progressDialog;

    public void showMessage(String message) { //вывод уведомления
        Snackbar.make(getCurrentFocus(), message, Snackbar.LENGTH_LONG).show();
    }

    public void showProgress(String message) { //отображение индикатора выполнения
        if (progressDialog == null)
            progressDialog = new ProgressDialog(this);

        progressDialog.setMessage(message);
        progressDialog.show();
    }

    public void hideProgress() { //скрытие индикатора
        if (progressDialog != null) {
            progressDialog.dismiss();
        }
    }
}
```

ПРИЛОЖЕНИЕ Р

Листинг Р.1. – Исходный код “Presenter”

```
package susu.com.animallocation.fragment.presenters.MapPresenter;

import android.os.Handler;

import com.google.android.gms.maps.model.LatLng;

import java.util.Timer;
import java.util.TimerTask;

import susu.com.animallocation.activities.MapActivity;

public class Presenter {
    private MapActivity activity;

    private Timer timer;

    private static final long DELAY  = 0;
    private static final long PERIOD = 5;

    public Presenter(MapActivity activity) {
        this.activity = activity;
    }

    public void startUpdateMap() { // условия для обновления карты
        timer = new Timer();
        timer.schedule(
                getInstanceTimerTask(),
                0,
                10000
        );
    }

    private void cancelUpdate() { //отмена обновления
        timer.cancel();
    }

    private TimerTask getInstanceTimerTask() {
        return new TimerTask() {
            @Override
            public void run() {
                handler.post(runnable);
            }
        };
    }

    private final Handler handler = new Handler();

    private final Runnable runnable = () -> Presenter.this.updateMap();
    private void updateMap() { //обновление карты
        LatLng animalCoordinates = activity.getAnimalCoordinates();

        if(animalCoordinates == null) {
            return;
        }
```

Продолжение листинга Р.1.

```
        if(activity.getMapMarker() != null) {
            activity.deleteMarker();

        }

        activity.setMarker(animalCoordinates);
    }
}
```

ПРИЛОЖЕНИЕ С

Листинг С.1. – Исходный код “LoginFragmentPresenter”

```
package susu.com.animallocation.fragment.presenters;

import android.content.Intent;
import android.support.transition.TransitionManager;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentTransaction;
import android.transition.Transition;

import susu.com.animallocation.R;
import susu.com.animallocation.fragment.LoginFragment;
import susu.com.animallocation.fragment.RegistrationFragment;
import susu.com.animallocation.fragment.RootFragment;
import susu.com.animallocation.models.Settings;
import susu.com.animallocation.models.apiclient.AccountInfo;
import susu.com.animallocation.models.apiclient.authentication.Authenticator;
import susu.com.animallocation.models.apiclient.authentication.LoginCallback;

public class LoginFragmentPresenter {

    private LoginFragment fragment;
    private Settings settings;

    private String email;
    private String password;

    public LoginFragmentPresenter(LoginFragment fragment) {
        this.fragment = fragment;
        settings = new Settings(fragment.getActivity());
    }

    public void checkAuthentication() { //аутенфикация пользователя
        if(settings.isLogin()) fragment.redirectToMap();
    }

    public void redirectToRegistration() { //перенаправление на страницу регистрациии
        fragment.getActivity()
                .getSupportFragmentManager()
                .beginTransaction()
                .setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN)
                .replace(R.id.fragment_container, new RegistrationFragment())
                .addToBackStack(null)
                .commit();
    }

    Authenticator authenticator;

    public void cancelAuth() { //отмена авторизации
        if(authenticator != null) {
            authenticator.cancel();
        }
        fragment.hideProgress();
    }
    public void login() { //авторизация
        fragment.showProgress();
```

Продолжение листинга С.1.

```
        email = fragment.getEmail();

       password = fragment.getPassword();
        authenticator = new Authenticator();

        authenticator.Authenticate(email, password, new LoginCallback() {
            @Override
            public void onComplete(AccountInfo accountInfo) { // успешное завершение
авторизации
                settings.saveAccountInfo(accountInfo);
                fragment.redirectToMap();
                fragment.hideProgress();
            }

            @Override
            public void onFail(String message) { //ошибка
                fragment.showMessage(message);
                fragment.hideProgress();
            }

            @Override
            public void onFailRequest() { //нет подключения

fragment.showMessage(fragment.getString(R.string.error_connection_server));
                fragment.hideProgress();
            }
        });
    }
}
```

ПРИЛОЖЕНИЕ Т

Листинг Т.1. – Исходный код “RegistrationFragmentPresenter”

```
package susu.com.animallocation.fragment.presenters;

import android.text.TextUtils;

import java.util.ArrayList;

import susu.com.animallocation.R;
import susu.com.animallocation.fragment.RegistrationFragment;
import susu.com.animallocation.models.apiclient.AccountInfo;
import susu.com.animallocation.models.apiclient.entities.Animal;
import susu.com.animallocation.models.Settings;
import susu.com.animallocation.models.Validator;
import susu.com.animallocation.models.apiclient.registration.ApiRegistration;
import susu.com.animallocation.models.apiclient.registration.RegistrationCallback;

public class RegistrationFragmentPresenter {
    private RegistrationFragment fragment;
    private Settings settings;

    private String email;
    private String password;

    public RegistrationFragmentPresenter(RegistrationFragment fragment) {
        this.fragment = fragment;
        settings = new Settings(fragment.getActivity());
    }

    ApiRegistration registration;
    public void registration() { //регистрация
        email             = fragment.getEmail();
        password          = fragment.getPassword();
        String firstname  = fragment.getFirstname();
        String surname    = fragment.getSurname();
        String birthday   = fragment.getBirthday();
        String animalName = fragment.getAnimalName();
        String animalId   = fragment.getAnimalId();

        if(!validateFields(email, password, firstname, surname, birthday, animalName,
animalId))
            return;

        Animal animal = new Animal(animalName, animalId);
        ArrayList<Animal> animals = new ArrayList<>();
        animals.add(animal);

        registration = new ApiRegistration();

        registration.Registration(email, password, firstname, surname, birthday,
animals,
                registrationCallback);
    }
    public void cancelRegistration() {
        if(registration != null) {
            registration.cancel();
```

Продолжение листинга Т.1.

```
        }
    }

    private boolean validateFields(String email, String password, String firstname,
String surname,
                                   String birthday, String animalName, String
animalId) { //проверка правильности ввода

        if(TextUtils.isEmpty(email)) {

fragment.setErrorLogin(fragment.getString(R.string.error_field_required));
            return false;
        }

        if(TextUtils.isEmpty(password)) {

fragment.setErrorPassword(fragment.getString(R.string.error_field_required));
            return false;
        }

        if(!Validator.validateEmail(email)) {

fragment.setErrorLogin(fragment.getString(R.string.error_incorrect_email));
            return false;
        }

        if(!Validator.validatePassword(password)) {

fragment.setErrorPassword(fragment.getString(R.string.error_incorrect_password));
            return false;
        }

        if(TextUtils.isEmpty(firstname)) {

fragment.setErrorFirstname(fragment.getString(R.string.error_field_required));
            return false;
        }

        if(TextUtils.isEmpty(surname)) {

fragment.setErrorSurname(fragment.getString(R.string.error_field_required));
            return false;
        }

        if(TextUtils.isEmpty(birthday)) {

fragment.setErrorBirthday(fragment.getString(R.string.error_field_required));
            return false;
        }

        if(!Validator.validateBirthday(birthday)) {

fragment.setErrorBirthday(fragment.getString(R.string.error_incorrect_birthday));
            return false;
        }

        if(TextUtils.isEmpty(animalName)) {
```

Продолжение листинга Т.1.

```
fragment.setErrorAnimalName(fragment.getString(R.string.error_field_required));
            return false;


        }

        if(TextUtils.isEmpty(animalId)) {

fragment.setErrorAnimalId(fragment.getString(R.string.error_field_required));
            return false;
        }

        return true;
    }

    private RegistrationCallback registrationCallback = new RegistrationCallback() {

        @Override
        public void onComplete(AccountInfo accountInfo) { //успешное завершение
            settings.saveAccountInfo(accountInfo);
            fragment.redirectToMap();
        }

        @Override
        public void onFail(String message) { // завершение с ошибкой
            fragment.showMessage(message);
        }

        @Override
        public void onFailRequest() { //ошибка подключения

fragment.showMessage(fragment.getString(R.string.error_connection_server));
        }
    };
}
```

ПРИЛОЖЕНИЕ У

Листинг У.1. – Исходный код “LoginFragment”

```
package susu.com.animallocation.fragment;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.view.inputmethod.EditorInfo;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import susu.com.animallocation.R;
import susu.com.animallocation.activities.RootActivity;
import susu.com.animallocation.fragment.presenters.LoginFragmentPresenter;

public class LoginFragment extends RootFragment {
    private AutoCompleteTextView emailView;
    private EditText passwordView;

    LoginFragmentPresenter presenter;

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) { //начальная установка
параметров
        super.onCreate(savedInstanceState);
        presenter = new LoginFragmentPresenter(this);
        presenter.checkAuthentication();
    }

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater,
                             @Nullable ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.login_fragment, container, false);
    }

    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        initialViews(view);
        getView().requestFocus();
    }

    @Override
    public void onPause() {
        super.onPause();
        presenter.cancelAuth();
    }
```

Продолжение листинга У.1.

```
    @Override
    public void onStop() {
        super.onStop();
        presenter.cancelAuth();
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        presenter.cancelAuth();
    }

    public String getEmail() { //получение логина
        return emailView.getText().toString();
    }

    public String getPassword() { //получение пароля
        return passwordView.getText().toString();
    }

    private void initialViews(View view) { //реакция на нажатие на экран
        emailView = view.findViewById(R.id.email);
        emailView.setOnEditorActionListener(editViewListener);

        passwordView = view.findViewById(R.id.password);
        passwordView.setOnEditorActionListener(editViewListener);

        Button buttonSignIn = view.findViewById(R.id.signInButton);
        buttonSignIn.setOnClickListener(buttonSignInListener);

        TextView registrationButton = view.findViewById(R.id.registrationButton);
        registrationButton.setOnClickListener(registrationButtonClickListener);
    }

    TextView.OnEditorActionListener editViewListener = (textView, id, keyEvent) -> {
        if (id == EditorInfo.IME_ACTION_DONE || id == EditorInfo.IME_NULL) {
            return true;
        }
        return false;
    };

    OnClickListener buttonSignInListener = v -> presenter.login();

    OnClickListener registrationButtonClickListener = v ->
presenter.redirectToRegistration();

    public void showProgress() {
        ((RootActivity) getActivity()).showProgress();
    }

    public void hideProgress() {
        ((RootActivity) getActivity()).hideProgress();
    }
}
```

ПРИЛОЖЕНИЕ Ф

Листинг Ф.1. – Исходный код “RegistrationFragment”

```
package susu.com.animallocation.fragment;

import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import susu.com.animallocation.R;
import susu.com.animallocation.fragment.presenters.RegistrationFragmentPresenter;

public class RegistrationFragment extends RootFragment {

    private AutoCompleteTextView emailView;
    private EditText passwordView;
    private AutoCompleteTextView userFirstnameView;
    private AutoCompleteTextView userSurnameView;
    private TextView userBirthdayView;
    private AutoCompleteTextView animalNameView;
    private AutoCompleteTextView animalIdView;

    private RegistrationFragmentPresenter presenter;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
                             @Nullable ViewGroup container,
                             @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.registration_fragment, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        presenter = new RegistrationFragmentPresenter(this);
        getView().requestFocus();
        initialViews(view);
    }

    @Override
    public void onPause() {
        super.onPause();
        presenter.cancelRegistration();
    }

    @Override
    public void onStop() {
        super.onStop();
```

Продолжение листинга Ф.1.

```
        presenter.cancelRegistration();
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        presenter.cancelRegistration();
    }

    private void initialViews(View view) { //реакция на нажатие
        emailView          = view.findViewById(R.id.email);
        passwordView       = view.findViewById(R.id.password);
        userFirstnameView  = view.findViewById(R.id.userFirstname);
        userSurnameView    = view.findViewById(R.id.userSurname);
        userBirthdayView   = view.findViewById(R.id.userBirthday);
        animalNameView     = view.findViewById(R.id.animalName);
        animalIdView       = view.findViewById(R.id.idKey);

        Button registrationButton = view.findViewById(R.id.signInButton);
        registrationButton.setOnClickListener(buttonRegistrationListener);
    }
//ввод данных
    public String getEmail()      { return emailView.getText().toString(); }
    public String getPassword()   { return passwordView.getText().toString(); }
    public String getFirstname()  { return userFirstnameView.getText().toString(); }
    public String getSurname()    { return userSurnameView.getText().toString(); }
    public String getBirthday()   { return userBirthdayView.getText().toString(); }
    public String getAnimalName() { return animalNameView.getText().toString(); }
    public String getAnimalId()   { return animalIdView.getText().toString(); }

//обработка ошибок
    public void setErrorFirstname(String error)  {
        userFirstnameView.setError(error);
        userFirstnameView.requestFocus();
    }

    public void setErrorSurname(String error)    {
        userSurnameView.setError(error);
        userSurnameView.requestFocus();
    }

    public void setErrorBirthday(String error)   {
        userBirthdayView.setError(error);
        userBirthdayView.requestFocus();
    }

    public void setErrorAnimalName(String error) {
        animalNameView.setError(error);
        animalNameView.requestFocus();
    }
    public void setErrorAnimalId(String error)   {
        animalIdView.setError(error);
        animalIdView.requestFocus();
    }
    public void setErrorLogin(String error) {
```

Продолжение листинга Ф.1.

```
        emailView.setError(error);
        emailView.requestFocus();
    }



    public void setErrorPassword(String error) {
        passwordView.setError(error);
        passwordView.requestFocus();
    }
    private OnClickListener buttonRegistrationListener = v -> {
        presenter.registration();
    };
}
```

ПРИЛОЖЕНИЕ X
ИСХОДНЫЙ КОД ROOTFRAGMENT

Листинг X.1. – Исходный код “RootFragment”

```
package susu.com.animallocation.fragment;

import android.content.Intent;
import android.support.design.widget.Snackbar;
import android.support.v4.app.Fragment;

import susu.com.animallocation.R;
import susu.com.animallocation.activities.MapActivity;

public class RootFragment extends Fragment {
    public void redirectToFragment(Fragment fragment) {
        getActivity()
                .getSupportFragmentManager()
                .beginTransaction()
                .replace(R.id.fragment_container, fragment)
                .commit();
    }

    public void redirectToMap() { //переход на карту
        Intent intent = new Intent(getActivity(), MapActivity.class);
        startActivity(intent);
        getActivity().finish();
    }

    public void showMessage(String message) {
         Snackbar.make(getActivity().getCurrentFocus(), message,
Snackbar.LENGTH_LONG).show();
    }
}
```

ПРИЛОЖЕНИЕ Ц
ИСХОДНЫЕ КОДЫ APICLIENT

Листинг Ц.1. – Исходный код “UpdateAnimalLocationTask ”

```
package susu.com.animallocation.models.apiclient.animal.location;

import android.os.AsyncTask;
import android.support.annotation.NonNull;
import com.google.android.gms.maps.model.LatLng;
import com.google.gson.Gson;
import java.io.IOException;
import okhttp3.Response;
import susu.com.animallocation.models.apiclient.ApiClient;

class UpdateAnimalLocationTask extends AsyncTask<Void, Void, LatLng> {

    private static final String ACTION_ROUTE = "/api/animallocation/get";

    private String email;
    private String idKey;
    private String accessToken;

    private Gson gson;

    private UpdateAnimalLocationCallback callback;

    public UpdateAnimalLocationTask(@NonNull String email,
                                    @NonNull String idKey,
                                    @NonNull String accessToken,
                                    @NonNull UpdateAnimalLocationCallback callback) {
        this.email    = email;
        this.idKey    = idKey;
        this.accessToken = accessToken;

        this.gson     = new Gson();

        this.callback = callback;
    }

    @Override
    protected LatLng doInBackground(Void... voids) { //отправка запроса
        try {
            ApiClient client = new ApiClient();
            String jsonData = gson
                    .toJson(new UpdateAnimalLocationModel(email, idKey));

            String url = ApiClient.HOST + ACTION_ROUTE;

            Response response = client.postRequest(url, accessToken, jsonData);
            if(response.isSuccessful() && response.code() == ApiClient.SUCCESS_CODE)
{
                String jsonResponse = response.body().string();
                LatLng animalLocation = gson.fromJson(jsonResponse, LatLng.class);

                return animalLocation;
            }
```

Продолжение листинга Ц.1.

```
            else if(response.code() == ApiClient.NOT_AUTHENTICATION_CODE) {
//                TODO: needed update access token
                return null;
            } else return null;
        } catch (IOException e) {
            return null;
        }
    }

    @Override
    protected void onPostExecute(LatLng latLng) {
        if(isCancelled()) return;

        if(latLng != null)
            callback.OnComplete(latLng);
        else
            callback.OnFail();
    }
}
```

Листинг Ц.2. – Исходный код “LoginTask”

```
package susu.com.animallocation.models.apiclient.authentication;

import android.os.AsyncTask;

import com.google.android.gms.common.api.Api;
import com.google.gson.Gson;

import okhttp3.Response;
import susu.com.animallocation.models.apiclient.AccountInfo;
import susu.com.animallocation.models.apiclient.ApiClient;
import susu.com.animallocation.models.apiclient.AuthResult;

class LoginTask extends AsyncTask <Void, Void, AuthResult> {

    private LoginModel model;
    private ApiClient client;
    private LoginCallback callback;

    private static final String ACTION_LOGIN = "/api/account/login";

    LoginTask(String email, String password, LoginCallback callback) {
        model = new LoginModel(email, password);
        client = new ApiClient();
        this.callback = callback;
    }
    @Override
    protected AuthResult doInBackground(Void... voids) { //отправка запроса
        Gson gson = new Gson();
        cancel(false);
        String jsonData = gson.toJson(model);
        try {
            String url = ApiClient.HOST + ACTION_LOGIN;
            Response response = client.postRequest(url, jsonData);
            if(response.code() == ApiClient.SUCCESS_CODE) {
```

Продолжение листинга Ц.2.

```
                String jsonResponse = response.body().string();

 AccountInfo accountInfo = gson.fromJson(jsonResponse, AccountInfo.class);

                return new AuthResult(accountInfo);
            } else {
                return new AuthResult(response.code(), response.message());
            }
         } catch (Exception e) {
            return null;
        }
    }
    @Override
    protected void onPostExecute(AuthResult result) {
        if(isCancelled()) return;

        if(result == null) {
            callback.onFailRequest();
          return;
}


        if(result.getCode() == ApiClient.SUCCESS_CODE) {
            callback.onComplete(result.getAccountInfo());
        } else {
            callback.onFail(result.getMessage());
        }
    }

    private class LoginModel {
        private String email;
        private String password;

        public LoginModel(String email, String password) {
            this.email = email;
            this.password = password;
        }
    }
}
```

Листинг Ц.3. – Исходный код “RegistrationTask”

```
package susu.com.animallocation.models.apiclient.registration;

import android.os.AsyncTask;
import android.text.TextUtils;

import com.google.android.gms.common.api.Api;
import com.google.gson.Gson;
import java.io.IOException;
import java.util.List;

import okhttp3.Response;
import susu.com.animallocation.models.apiclient.AccountInfo;
import susu.com.animallocation.models.apiclient.AuthResult;
```

Продолжение листинга Ц.3.

```
import susu.com.animallocation.models.apiclient.entities.Animal;
import susu.com.animallocation.models.apiclient.ApiClient;

class RegistrationTask extends AsyncTask<Void, Void, AuthResult> {

    private final String ACTION_REGISTRATION = "/api/account/registration";

    private RegistrationCallback callback;
    private RegistrationModel model;
    private Gson gson;
    private ApiClient apiClient;

    RegistrationTask(String email, String password,
                     String name, String surname, String birthday,
                     List<Animal> animals,
                     RegistrationCallback callback) {

        model = new RegistrationModel(name, surname, birthday, email, password,
animals);
        this.callback = callback;
        gson = new Gson();
        apiClient = new ApiClient();
    }

    @Override
    protected AuthResult doInBackground(Void... voids) {
        String url = ApiClient.HOST + ACTION_REGISTRATION;
        String jsonData = gson.toJson(model);
        try {
            Response response = apiClient.postRequest(url, jsonData);
            if(response.isSuccessful() && response.code() == ApiClient.SUCCESS_CODE)
{
                String jsonResponse = response.body().string();
                AccountInfo accountInfo = gson.fromJson(jsonResponse,
AccountInfo.class);

                return new AuthResult(accountInfo);
            } else {
                return new AuthResult(response.code(), response.message());
            }
        } catch (IOException e) {
            return null;
        }
    }

    @Override
    protected void onPostExecute(AuthResult result) {
        if(isCancelled()) return;

        if(result == null) {
            callback.onFail("Connection failed");
            return;
        }
```

Продолжение листинга Ц.3.

```
        if(result.getCode() == ApiClient.SUCCESS_CODE) {
            callback.onComplete(result.getAccountInfo());
        }else {
            callback.onFail(result.getMessage());
        }
    }

    private class RegistrationModel {
        private String name;
        private String surname;
        private String birthday;
        private String email;
        private String password;
        private List<Animal> animals;

        public RegistrationModel(String name, String surname, String birthday,
                                 String email, String password,List<Animal> animals )
{
            this.name = name;
            this.surname = surname;
            this.birthday = birthday;
            this.email = email;
            this.password = password;
            this.animals = animals;
        }
    }
}
```

ПРИЛОЖЕНИЕ Ш

Листинг Ш.1. – Исходный код “AuthResult”

```
package susu.com.animallocation.models.apiclient;

public class AuthResult {
    private AccountInfo accountInfo;
    private String message;
    private int code;

    public AuthResult(int code, String message) {
        this.code = code;
        this.message = message;
    }

    public AuthResult(AccountInfo accountInfo) { //сравнение пользователей
        this.accountInfo = accountInfo;
        this.code = ApiClient.SUCCESS_CODE;
        this.message = null;
    }

    public AccountInfo getAccountInfo() {
        return accountInfo;
    }

    public String getMessage() {
        return message;
    }

    public int getCode() {
        return code;
    }
}
```

ПРИЛОЖЕНИЕ Щ

Листинг Щ.2. – Исходный код “LoginTask”

```
package susu.com.animallocation.models.apiclient;

import java.io.IOException;
import java.util.concurrent.TimeUnit;

import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;
import okhttp3.OkHttpClient.Builder;

public class ApiClient {

//параметры подключения
    public static final String HOST = "http://80.78.248.202";

    public static final int SUCCESS_CODE            = 200;
    public static final int BAD_REQUEST             = 400;
    public static final int NOT_AUTHENTICATION_CODE = 401;

    private static final MediaType JSON = MediaType.parse("application/json;
charset=utf-8");
    private static final String AUTHORIZATION = "Authorization";
    private static final int TIMEOUT = 30;

    private OkHttpClient client;
    public ApiClient() {
        client = new Builder()
                .writeTimeout(TIMEOUT, TimeUnit.SECONDS)
                .readTimeout(TIMEOUT, TimeUnit.SECONDS)
                .build();
    }

    public Response postRequest(String url, String data) throws IOException {
//формирование запроса

        RequestBody body = RequestBody
                .create(JSON, data);

        Request request = new Request.Builder()
                .post(body)
                .url(url)
                .build();
       Response response = client.newCall(request).execute();
       return response;
    }
    public Response postRequest(String url, String token, String jsonData) throws
IOException {
        RequestBody body = RequestBody.create(JSON, jsonData);

        String authToken = String.format("Bearer %s", token);

        Request request = new Request.Builder()
```

Продолжение листинга Щ.1.

```
                .post(body)
                .header(AUTHORIZATION, authToken)

                .url(url)
                .build();

        return client.newCall(request).execute();
    }

}
```

ПРИЛОЖЕНИЕ Э

Листинг Э.1. – Исходный код “Setting”

```
package susu.com.animallocation.models;

import android.content.Context;
import android.content.SharedPreferences;

import susu.com.animallocation.models.apiclient.AccountInfo;
import susu.com.animallocation.models.apiclient.entities.Animal;

public class Settings {

    private static final String ACCOUNT_SETTINGS  = "ACCOUNT_SETTINGS";
    private static final String NAME = "NAME";
    private static final String SURNAME = "SURNAME";
    private static final String ACCESS_TOKEN = "ACCESS_TOKEN";

    private static final String AMOUNT_ANIMALS = "AMOUNT_ANIMALS";

    private static final String IS_LOGIN = "IS_LOGIN";

    private SharedPreferences connectionSettings;
    private SharedPreferences accountSettings;

    public Settings(Context context) {
        accountSettings = context.getSharedPreferences(ACCOUNT_SETTINGS,
Context.MODE_PRIVATE);
    }

    public void saveAccountInfo(AccountInfo accountInfo) {
        if(accountInfo == null) throw new NullPointerException("account info is
empty");

        SharedPreferences.Editor editor = accountSettings.edit()
                .putString(NAME, accountInfo.getName())
                .putString(SURNAME, accountInfo.getSurname())
                .putString(ACCESS_TOKEN, accountInfo.getAccessToken())
                .putInt(AMOUNT_ANIMALS, accountInfo.getAnimals().size())
                .putBoolean(IS_LOGIN, true);

        for (Animal animal :
                accountInfo.getAnimals()) {
            editor.putString(animal.getIdKey(), animal.getName());
        }

        editor.apply();
    }

    public String getAccessToken() {
        return accountSettings.getString(ACCESS_TOKEN, null);
    }

    public boolean isLogin() {
        return accountSettings.getBoolean(IS_LOGIN, false);
    }
}
```

ПРИЛОЖЕНИЕ Ю

Листинг Ю.1. – Исходный код “LoginTask”

```
package susu.com.animallocation.models;

import android.text.TextUtils;

public class Validator {
    public static boolean validateEmail(String email) {
        return true;
    }

    public static boolean validatePassword(String password) {
        return true;
    }

    private final String pattern = "[1-9]{2}-[1-9]{2}-[1-9]{2}";
    public static boolean validateBirthday(String birthday) {
        return true;
    }

    public static boolean isEmptyField(String field) {
        return TextUtils.isEmpty(field);
    }
```

ПРИЛОЖЕНИЕ Я

Листинг Я.1. – Исходный код “ UpdateAnimalLocationService”

```
package susu.com.animallocation.services.UpdateAnimalLocation;

import android.app.Service;
import android.content.Intent;
import android.os.Binder;
import android.os.IBinder;
import android.support.annotation.Nullable;

import com.google.android.gms.maps.model.LatLng;

import java.util.Timer;
import java.util.TimerTask;

import
susu.com.animallocation.models.apiclient.animal.location.UpdateAnimalLocationCallback
;
import susu.com.animallocation.models.apiclient.animal.location.UpdaterAnimaLocation;

public class UpdateAnimalLocationService extends Service {
    private static final String TAG = "UpdateAnimalLocationService";

    private final IBinder binder = new ServiceBinder();

//    coordinates of animal
    private LatLng animalLocation;

//    status result getting coordinates of animal
    private boolean isSuccessful;

    private String email;
    private String idKey;
    private String accessToken;

    @Override
    public void onCreate() {
        super.onCreate();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        return super.onStartCommand(intent, flags, startId);
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        email = intent.getStringExtra("EMAIL");
        idKey = intent.getStringExtra("ID_KEY");
        accessToken = intent.getStringExtra("TOKEN");
```

Продолжение листинга Я.1.

```
        updateLocation();

        return binder;
    }

    public void updateLocation() {
        Timer timer = new Timer();
        timer.schedule(getInstanceTimerTask(),
                0, 10000);
    }

    private TimerTask getInstanceTimerTask() {
        return new TimerTask() {
            @Override
            public void run() {
                UpdaterAnimaLocation.UpdateAnimalLocation(email, idKey, accessToken,
                        new UpdateAnimalLocationCallback() {
                            @Override
                            public void OnComplete(LatLng animalLocation) {
                                UpdateAnimalLocationService.this.animalLocation =
animalLocation;
                                UpdateAnimalLocationService.this.isSuccessful = true;
                            }


                            @Override
                            public void OnFail() {
                                UpdateAnimalLocationService.this.isSuccessful =
false;
                            }
                        });
            }
        };
    }

    public LatLng getLatLng() {
        return animalLocation;
    }

    public boolean isSuccessful() {
        return isSuccessful;
    }

    public class ServiceBinder extends Binder {
        public LatLng getAnimalCoordinates() {
            return animalLocation;
        }

        public boolean isSuccessful() {
            return isSuccessful;
        }
    }
}
```

ПРИЛОЖЕНИЕ А

Листинг А.1. – Исходный код “Login_fragment.xml”

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".fragment.LoginFragment">

    <ScrollView
        android:id="@+id/login_form"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent">

        <LinearLayout
            android:id="@+id/email_login_form"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <AutoCompleteTextView
                    android:id="@+id/email"
                    android:hint="@string/prompt_email"
                    android:inputType="textEmailAddress"
                    style="@style/EditTextStyle"/>

            </android.support.design.widget.TextInputLayout>

            <android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/password"
                    android:hint="@string/prompt_password"
                    android:inputType="textPassword"
                    style="@style/EditTextStyle"/>

            </android.support.design.widget.TextInputLayout>

            <Button
                android:id="@+id/signInButton"
                android:layout_marginTop="16dp"
```

Продолжение листинга А.1.

```
                android:text="@string/action_sign_in"
                style="@style/ButtonStyle"
                android:background="@drawable/btn_anim"/>

            <TextView
                android:id="@+id/registrationButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/registration"
                android:textSize="20dp"
                android:textColor="@color/colorPrimary"
                android:fontFamily="sans-serif-light"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="@dimen/activity_vertical_margin"/>

        </LinearLayout>
    </ScrollView>
</android.support.constraint.ConstraintLayout>
```

ПРИЛОЖЕНИЕ В

Листинг В.1. – Исходный код “Map_Activity.xml”

```
<?xml version="1.0" encoding="utf-8"?>
<fragment
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".activities.MapActivity" />
```

ПРИЛОЖЕНИЕ С

Листинг С.1. – Исходный код “Registration_fragment.xml”

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".fragment.RegistrationFragment"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <AutoCompleteTextView
                    android:id="@+id/email"
                    android:hint="@string/prompt_email"
                    android:inputType="textEmailAddress"
                    style="@style/EditTextStyle"/>

            </android.support.design.widget.TextInputLayout>

            <android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/password"
                    android:hint="@string/prompt_password"
                    android:inputType="textPassword"
                    style="@style/EditTextStyle"/>

            </android.support.design.widget.TextInputLayout>

            <android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <AutoCompleteTextView
                    android:id="@+id/userFirstname"
                    android:hint="@string/user_name"
                    style="@style/EditTextStyle"/>
```

Продолжение листинга С.1.

```
</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <AutoCompleteTextView
        android:id="@+id/userSurname"
        android:hint="@string/user_surname"
        style="@style/EditTextStyle"/>

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <EditText
        android:id="@+id/userBirthday"
        android:inputType="date"
        android:ems="10"
        android:hint="@string/user_birthday"
        style="@style/EditTextStyle"/>

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <AutoCompleteTextView
        android:id="@+id/animalName"
        android:hint="@string/animal_name"
        style="@style/EditTextStyle"/>

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">



    <AutoCompleteTextView
        android:id="@+id/idKey"
        android:hint="@string/animal_id"
        android:inputType="number"
        style="@style/EditTextStyle"/>

</android.support.design.widget.TextInputLayout>

<Button
    android:id="@+id/signInButton"
    android:layout_marginTop="16dp"
```

```
                android:text="@string/registration"
                style="@style/ButtonStyle"/>
        </LinearLayout>
    </ScrollView>
</LinearLayout>
```

ПРИЛОЖЕНИЕ С

Листинг С.1. – Исходный код “Root_Activity.xml”

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ProgressBar
        android:id="@+id/progress"
        android:layout_width="match_parent"
        android:layout_height="4dp"
        style="?android:attr/progressBarStyleHorizontal"
        android:indeterminateBehavior="repeat"
        android:indeterminate="true"
        android:scaleY="4"
        android:visibility="invisible"/>

    <LinearLayout
        android:orientation="vertical"
        android:id="@+id/fragment_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</LinearLayout>
```

ПРИЛОЖЕНИЕ Е

Листинг Е.1. – Общие AT-команды

```
AT
OK
AT+GMI //информация о модуле
Quectel_Ltd
Quectel_MC60E
Revision: MTK 0828

OK
AT+GSN //IMEI модуля
861359032495515

OK
AT+GCAP //возможности модуля
+GCAP: +CGSM,+FCLASS

OK

AT+CCLK? //дата и время устройства
+CCLK: "04/01/01,00:02:37+00"

OK
AT+CBC//уровень заряда
+CBC: 0,72,3870

OK
_
```

ПРИЛОЖЕНИЕ F

Листинг F.1. – AT-команды для GSM

```
AT+QSIMDET?//режим детектирование СИМ карты
+QSIMDET: 0,0,0

OK
AT+QSIMDET = 1, 1
OK
AT+QSIMDET?
+QSIMDET: 1,1,0

OK

AT+COPS? //оператор
+COPS: 0,0,"TELE2"

OK
AT+GSN
861359032495515

OK
AT+CPAS //состояние модуля
+CPAS: 0 //готов к работе

OK

AT+CREG? //сеть
+CREG: 0,1

OK

AT+CSQ//сигнал
+CSQ: 16,0

OK

ATD +79049766348; //звонок
OK
ATH
OK
```

ПРИЛОЖЕНИЕ G

Листинг G.1. – AT-команды для GNSS

```
AT+QGNSSC=1 // вкл GNSS

OK

AT+QGNSSRD? //информация о местоположении

+QGNSSRD: $GNRMC,004141.669,V,,,,,0.00,0.00,010104,,,N*5E

$GNVTG,0.00,T,,M,0.00,N,0.00,K,N*2C

$GNGGA,004141.669,,,,,0,0,,,M,,M,,*5F

$GPGSA,A,1,,,,,,,,,,,,,,*1E

$GLGSA,A,1,,,,,,,,,,,,,,*02

$GPGSV,1,1,00*79

$GLGSV,1,1,00*65

$GNGLL,,,,,004141.669,V,N*6D


OK

AT+QGNSSTS? //время синхронизации

+QGNSSTS: 1


OK

AT+QGNSSEPO = 1// вкл AGPS

OK

AT+QGEPOAID

OK
```

ПРИЛОЖЕНИЕ Н

Листинг Н.1. – AT-команды для GPRS

```
AT+CGATT?
+CGATT: 1

OK
AT+CGPADDR
+CGPADDR: 1, ""

+CGPADDR: 2, ""

+CGPADDR: 3, ""

OK
AT+QIFGCNT?
+QIFGCNT: 0,0

OK
AT+QIFGCNT=1
OK
AT+QGSMLOC =1 //местоположение, дата, время
+QGSMLOC: 0,61.373543,55.121454,2019/06/10,11:07:56

OK
AT+QICSGP=1,"internet.tele2.ru"
OK
```

ПРИЛОЖЕНИЕ I

Листинг I.1. – AT-команды для передачи данных

```
AT+QIOPEN="TCP","80.78.248.202","80" //подключение
OK

CONNECT OK

AT+QISEND //запрос

> POST /api/animallocation/put HTTP/1.1

HOST: 80.78.248.202

Accept: application/json

Content-Type: application/json

Content-Length: 104

{"email":"aslash11@mail.ru","idKey":"221133","Coordinates":{"latitude":
55.121543,"longitude": 61.37159}}


AT+QICLOSE //отключение
CLOSE OK
_
```

ПРИЛОЖЕНИЕ J

Листинг J.1. – последовательность AT-команд для определения местоположения

```
AT+QSIMDET=1, 1
AT+QSIMDET=1, 1
OK
AT+QGNSSC=1
AT+QGNSSC=1
OK
AT+QIFGCNT=2
AT+QIFGCNT=2
OK
AT+QICSGP=1,"internet.tele2.ru"
AT+QICSGP=1,"internet.tele2.ru"
OK
AT+CREG?;+CGREG?
AT+CREG?;+CGREG?
+CREG: 0,1

+CGREG: 0,1

OK
AT+QGNSSEPO=1
AT+QGNSSEPO=1
OK
AT+QGEPOAID
AT+QGEPOAID
OK



AT+QGNSSRD?
AT+QGNSSRD?
+QGNSSRD:
$GNRMC,144641.000,A,5507.2813,N,06122.3096,E,0.77,221.28,170619,,,A*75
$GNVTG,221.28,T,,M,0.77,N,1.43,K,A*2E
$GNGGA,144641.000,5507.2813,N,06122.3096,E,1,4,3.16,240.1,M,-11.6,M,,*68
$GPGSA,A,3,15,13,27,,,,,,,,,3.31,3.16,0.98*05
$GLGSA,A,3,68,,,,,,,,,,,3.31,3.16,0.98*14
$GPGSV,3,1,09,20,79,219,,21,66,128,,10,50,237,,27,47,291,17*74
$GPGSV,3,2,09,15,43,071,30,13,21,041,20,08,19,319,,16,14,253,*7C
$GPGSV,3,3,09,30,01,356,*42
$GLGSV,3,1,10,84,86,211,,69,67,199,,68,55,049,17,85,31,317,*66
$GLGSV,3,2,10,83,27,143,,76,14,339,,77,10,037,,70,10,216,*66
$GLGSV,3,3,10,67,04,039,,75,03,300,*69
$GNGLL,5507.2813,N,06122.3096,E,144641.000,A,A*41

OK
```

ПРИЛОЖЕНИЕ К

Листинг К.1. – последовательность АТ-команд для передачи данных

```
AT+QSIMDET=1, 1
AT+QSIMDET=1, 1
OK
AT+COPS?
AT+COPS?
+COPS: 0,0,"TELE2"

OK
AT+QIFGCNT=1
AT+QIFGCNT=1
OK
AT+QICSGP=1,"internet.tele2.ru"
AT+QICSGP=1,"internet.tele2.ru"
OK
AT+QIOPEN="TCP","80.78.248.202",80
AT+QIOPEN="TCP","80.78.248.202",80
OK

CONNECT OK

AT+QISEND
> POST /api/animallocation/put HTTP/1.1
HOST: 80.78.248.202
Accept: application/json
Content-Type: application/json
Content-Length: 104
{"email":"aslash11@mail.ru","idKey":"221133","Coordinates":{"latitude":55.121
543,"longitude":61.37159}}

SEND OK
```

ПРИЛОЖЕНИЕ L

Листинг L.1. – расшифровка ответа от GPS

```
+QGNSSRD:
$GNRMC,144641.000,A,5507.2813,N,06122.3096,E,0.77,221.28,170619,,,A*75
//минимальный рекомендованный пакет данных
$GNVTG,221.28,T,,M,0.77,N,1.43,K,A*2E //направление движения
$GNGGA,144641.000,5507.2813,N,06122.3096,E,1,4,3.16,240.1,M,-11.6,M,,*68
//данные местоположения
$GPGSA,A,3,15,13,27,,,,,,,,,3.31,3.16,0.98*05
$GLGSA,A,3,68,,,,,,,,,,,,3.31,3.16,0.98*14
$GPGSV,3,1,09,20,79,219,,21,66,128,,10,50,237,,27,47,291,17*74 //информация о
видимых спутниках GPS
$GPGSV,3,2,09,15,43,071,30,13,21,041,20,08,19,319,,16,14,253,*7C
$GPGSV,3,3,09,30,01,356,*42
$GLGSV,3,1,10,84,86,211,,69,67,199,,68,55,049,17,85,31,317,*66 //информация о
видимых спутниках ГЛОНАСС
$GLGSV,3,2,10,83,27,143,,76,14,339,,77,10,037,,70,10,216,*66
$GLGSV,3,3,10,67,04,039,,75,03,300,*69
$GNGLL,5507.2813,N,06122.3096,E,144641.000,A,A*41 //координаты
```

ПРИЛОЖЕНИЕ М

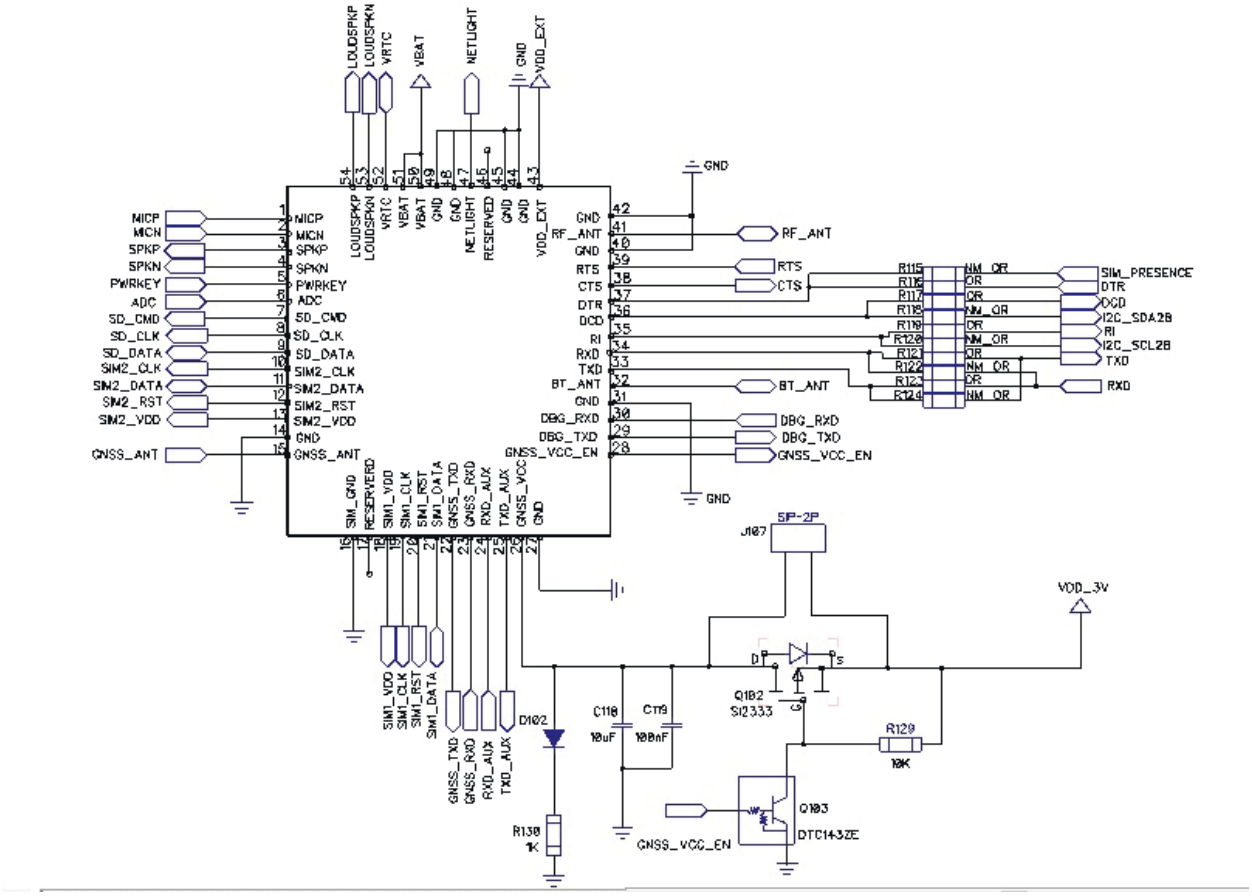

Рисунок М.1. – Электрическая схема обвязки модуля