МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е. АЛЕКСЕЕВА»
(НГТУ)

Институт (факультет) Институт радиоэлектроники и информационных технологий (ИРИТ)
Направление подготовки (специальность) 09.03.01 Информатика и вычислительная техника
Направленность (профиль) образовательной программы Вычислительные машины, комплексы, системы и сети
Кафедра Вычислительные системы и технологии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

бакалавра
Студента Шляпникова Сергея Михайловича группы 14-В-1

на тему Программная система отслеживания качества дорожного покрытия

СТУДЕНТ

Шляпников С. М.
(подпись) (фамилия, и., о.)

РУКОВОДИТЕЛЬ

Гай В.Е.
(подпись) (фамилия, и., о.)

ЗАВЕДУЮЩИЙ КАФЕДРОЙ

Кондратьев В.В.
(подпись) (фамилия, и., о.)

КОНСУЛЬТАНТЫ:

1. По _____
_____ _____
(подпись) (фамилия, и., о.)

2. По _____
_____ _____
(подпись) (фамилия, и., о.)

3. По _____
_____ _____
(подпись) (фамилия, и., о.)

ВКР защищена _____ _
(дата)

протокол №_____

с оценкой_____

МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е. АЛЕКСЕЕВА»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Шляпникова Сергея Михайловича

(фамилия, имя, отчество)

Институт (факультет) Институт радиоэлектроники и информационных технологий (ИРИТ)

Кафедра Вычислительные системы и технологии

Группа 14-В-1

Дата защиты «___» _____

Индекс

Наименование содержимого	Выпускная квалификационная работа	
Название (тема) ВКР	Программная система отслеживания качества дорожного покрытия	
ФИО студента (полностью)	Шляпников Сергей Михайлович	
Институт, факультет	Институт радиоэлектроники и информационных технологий (ИРИТ)	
Выпускающая кафедра	Вычислительные системы и технологии	
	код	Наименование
Направление подготовки /специальность	09.03.01	Информатика и вычислительная техника
Профиль подготовки / магистерская программа/ специализация	Вычислительные машины, комплексы, системы и сети	
Форма обучения	очная	Группа: 14-В-1
ФИО руководителя	Гай Василий Евгеньевич	
Год защиты ВКР	2018	

Содержание

Введение

В настоящее время одной из самых важных и приоритетных задач при планировании маршрута поездки является качество дорожного покрытия. Ключевым этот фактор является по той причине, что неудовлетворительное содержание и обслуживание улично-дорожной сети влияет не только на ухудшение технических характеристик автотранспортного средства и преждевременный выход его из строя, но и оказывает огромное влияние на безопасность использования этих дорог. Так, согласно статистике, за 2016 год в России произошло 173 694 ДТП, причиной 71 550 аварий послужило плохое качество дорожного полота (этот показатель вырос в сравнении с прошлым годом на 13.4%).

Как видно из статистики, динамика роста числа аварий остается положительной. По этой причине задача своевременного обнаружения проблемных участков автодорог (с последующим их устранением) становится всё более актуальной. К тому же, с ростом суммарной протяжённости дорог эта задача также становится всё более сложной и требует пересмотра традиционных методов контроля качества.

На данный момент методы контроля качества регулируются Приказом Минтранса РФ от 8 июня 2012 г. N 163 "Об утверждении Порядка проведения оценки уровня содержания автомобильных дорог общего пользования федерального значения", однако, результаты оценки не публикуются в общем доступе, что делает невозможным использование этих данных в личных целях.

Целью выполняемой работы является устранение недостатков уже существующих методов, а также их автоматизация и модернизация.

1. Техническое задание

1.1. Назначение разработки и область применения

Программный продукт предназначен для проведения анализа качества дорожного полотна, а также отображения полученных и обработанных сведений на карту дорожной сети.

В настоящее время имеется несколько альтернативных путей получения этих сведений, но они имеют ряд недостатков, таких как:

1) Необходимость приобретения дополнительного оборудования для произведения измерений

2) Высокая стоимость проведения измерений, обслуживания и содержания дополнительных инструментов

3) Требуют больших временных затрат (обладают низкой скоростью)

Все эти недостатки влекут за собой недостаточный, а порой очень низкий, уровень качества мониторинга состояния дорожного полотна. Что в свою очередь приводит к увеличению темпов старения и износа дорожного покрытия. А именно этот фактор является одним из определяющих в снижении аварийности.

Предлагаемый способ лишен вышеназванных недостатков. Кроме того, данный метод обладает и рядом преимуществ в сравнении с существующими системами: например, появляется гораздо больше возможностей по анализу и взаимодействию с данными, что обеспечивается легкостью и массовостью их получения. Также нельзя не отметить и простоту, которая заключается в том, что для проведения измерений нет необходимости в обладании какими-либо специальными и узкими знаниями.

Самым большим преимуществом разрабатываемой системы является её дешевизна в сравнении с конкурентами, а точнее отсутствие необходимости в приобретении специального оборудования. Всё, что необходимо – это смартфон с поддержкой систем навигации (GPS / Российская система Глонасс / Китайская Бэйдоу) и встроенным акселерометром, удерживающее устройство для телефона, а также автомобиль.

1.2. Технические требования

Опишем требования, предъявляемые к разрабатываемой системе. Можно выделить три группы требований к разным устройствам: мобильному устройству, компьютеру-серверу, клиентскому компьютеру.

Требования к мобильному устройству:

- Наличие подключения к Интернету (Но возможна работа в автономном режиме)
- Операционная система не ниже Android 4.4 KitKat

- Устройство должно быть оборудовано модулем приема спутникового сигнала местоположения

- Устройство должно иметь акселерометр

Требования к серверному компьютеру:

- Наличие подключения к Интернету

- Любая ОС с поддержкой виртуальной машины Java

- Аппаратные требования: 1024Мб ОЗУ, процессор Pentium Dual-Core 3.3 ГГц или аналогичное решение AMD, 500Мб ПЗУ (Может превысить в зависимости от количества данных).

Требования к клиентскому компьютеру:

- Наличие подключения к Интернету

- Любая ОС, имеющая Интернет-браузер

- Требования к аппаратному обеспечению определяются ОС

- Мышь, дисплей, клавиатура

- Минимально поддерживаемые версии браузеров: IE9, Mozilla Firefox 6, Google Chrome 10, Safari 5.

1.3. Требования к функциональной структуре мобильного приложения

Функционал продукта должен обеспечиваться несколькими взаимодействующими друг с другом модулями:

- Модуль сбора сведений о вибрациях, основанный на акселерометре устройства, которым выполняется фиксация неровностей дороги;

- Модуль сбора сведений о местоположении, принцип работы которого основан на получении данных от систем спутниковой навигации;

- Модуль сопоставления данных о вибрации с местоположением, где выполняется процесс сопоставления «сырых» данных;

- Сетевой модуль, который реализует взаимодействие с сервером;

- Картографический модуль, визуализирующий собранную информацию и наносящий её на карту дорожной сети;

- Центральный модуль, основная задача которого сводится к обеспечению взаимодействия вышеописанных модулей.

1.4. Требования к реализации

Требования к технической стороне реализации системы:

- В качестве протокола взаимодействия между компонентами системы на транспортно-сетевом уровне необходимо использовать протокол TCP/IP;

- Для реализации обмена информацией между компонентами системы необходимо использовать протоколы прикладного уровня: HTTP и HTTPS;

- Для организации структурированного хранения всех сведений использовать БД MySQL;

- Для реализации картографических средств системы использовать Google Map API и Open Street Map API.

1.5. Требования к информационной безопасности

Требования к обеспечению информационной безопасности реализуемого программного продукта:

- Сбор всех сведений осуществляется без персонализации (анонимно);

- Доступ к базам данных должен быть ограничен;

- Необходимо реализовать поддержку разграничение прав доступа: пользователи и администратор.

2. Анализ технического задания

2.1. Выбор операционной системы для разрабатываемого мобильного приложения

Прежде всего перед началом разработки необходимо определить операционную систему, для которой будет разрабатываться система мониторинга качества дорожного покрытия. На текущий момент на рынке существует несколько мобильных операционных систем. Согласно статистике, на конец 2017 года самыми популярными ОС являются:

- Android (73.53%)
- iOS (19.37%)
- Windows Phone (0.77%)

Из статистики видно, что с большим преимуществом на рынке лидирует мобильная операционная система от Google. В то время, как смартфоны на базе ОС от Microsoft – уже достаточная редкость.

Таким образом, одним из главных и значительных преимуществ Android является его распространенность среди пользователей.

2.2. Сравнение инструментов разработки

Проведем сравнение программных средств, предназначенных для разработки под каждую из операционных систем.

В этой категории сравнения у всех систем паритет. Каждая из них имеет как основную среду, предоставляемую разработчиком системы, так и аналоги от сторонних производителей. Например, предоставляемая Google бесплатная среда для разработки – Android Studio, у Microsoft это – Visual Studio, а у Apple – Apple[XCode]. Однако здесь есть важный момент: для разработки приложений под операционную систему Apple, нужно иметь установленную настольную систему Mac, что делает невозможным разработку при отсутствии компьютера этой компании.

В итоге, начать разрабатывать приложения под операционные системы Google и Microsoft несколько проще, чем под iOS, так как первые не накладывают дополнительных ограничений на вход.

2.3. Сравнение магазинов приложений операционных систем

Так как одним из важных факторов для развития программного продукта является его популярность и количество пользователей, то стоит учитывать и количество активных пользователей каждой из систем. По той причине, что Windows Phone имеет низкое количество пользователей, эту ОС рассматривать не будем.

Достаточной массовости и популярности продукта можно добиться в случае публикации его в магазине приложений. Каждая из ОС предлагает свой магазин приложений: Android – Play Store, а iOS в свою очередь – App Store. Сравнение этих двух магазинов показывает, что число загрузок (а, следовательно, и активных пользователей) из магазина Google более чем в три с половиной раза превышает показатели магазина Apple (920 миллиардов против 260).

Поэтому в плане развития проекта, операционная система Android может предоставить больший потенциал, нежели конкурент.

2.4. Сравнение устройств, использующих рассматриваемые ОС

При выборе операционной системы важным фактором является и то, что она во многом определяет и сам телефон (аппаратное обеспечение устройства). Так, на операционной системе Android свои устройства выпускает сразу несколько вендоров, в то время как устройства с операционной системой Apple выпускает лишь сама компания. Устройства на операционной системе от Google, как правило, более доступные в сравнении с конкурентами, при этом они обладают минимально необходимыми характеристиками для работы разрабатываемого продукта.

2.5. Сравнение API операционных систем

В соответствии с требованиями к продукту, в мобильном приложении необходимо будет работать с аппаратными компонентами устройства, а именно: акселерометром и приемником спутникового сигнала для определения местоположения. Поэтому следует учитывать и удобство работы с этими системами.

В этом вопросе системы достаточно схожи: они предоставляют гибкий и удобный интерфейс для обращения к нужным подсистемам и работе с ними. Так, для этого у операционной системы Apple имеется класс UIAcceleration, который аккумулирует всю информацию, полученную с датчиков, включая временной маркер, позволяющий определить время замера указанных величин. В Android этот функционал реализован интерфейсом SensorEventListener, который включает в себя метод onSensorChanged(), который вызывается каждый раз, когда изменяются показания датчиков.

Как видно, по открытости и доступности аппаратных компонентов обе операционные системы схожи между собой.

Таким образом, на начальном этапе развития проекта, закрытость для разработчиков компании Apple является недостатком. Однако процент рынка у iOS и её популярность достаточно велики, поэтому в дальнейшем стоит обязательно уделить внимание и этой системе. На текущем, «концептуальном» этапе развития проекта предпочтение стоит отдать

системе Android, ввиду большей популярности и распространенности, а также доступности устройств на этой системе, и более низким порогом входа в разработку.

2.6. Выбор языка программирования мобильного приложения

Когда выбрана операционная, можно переходить к выбору языка разработки. На текущий момент существует несколько языков программирования, на которых ведется разработка для этой операционной системы. Среди них Java, и недавно появившийся, но быстро набирающий популярность, Kotlin. Рассмотрим их подробнее.

Kotlin – молодой язык, разработанный российской компанией JetBrains, который заручился официальной поддержкой Google в июне 2017 года. Можно выделить несколько ключевых возможностей и преимуществ этого языка, среди которых:

- Компилируется в байткод JVM
- Можно совмещать с существующими Java-фреймворками и библиотеками
- Достаточно прост в изучении
- Является null-безопасным языком
- Имеет гибкий и простой синтаксис

Рассмотрим отличия от Java

Как было сказано выше, Kotlin является null-безопасным, что исключает неопределенности, которые могут возникать в Java.

Также в Kotlin появились специальные классы данных (Data Classes), основное предназначение которых, автоматически реализующих ряд функций, таких как: getter, setter, equals, hashCode и toString.

Была добавлена возможность расширения существующего класса, при этом не прибегая к наследованию, а используя функцию-расширения.

Язык обзавелся умным приведением типов, что обеспечивает большее удобство при разработке.

Но несмотря на все предоставляемые удобства и новшества, выбор был сделан в пользу Java, на это решение повлияло несколько фактора:

- Компиляция Kotlin-проекта занимает примерно на 10-15% больше времени, чем компиляция аналогичного проекта на языке Java

- Kotlin – молодой язык, и это сказывается и на объеме доступной информации по нему – её не так много, и порой это доставляет определенные трудности и сложности в процессе разработки

У Kotlin большое будущее, скорее всего, он станет следующим этапом развития Java (с которым он полностью совместим), однако на момент разработки текущего проекта он

достаточно молод и выполнять на нем столь серьезный и ответственный проект – это довольно рискованно.

2.7. Выбор среды разработки

Как говорилось выше, Google предоставляет официальную бесплатную среду для разработки приложений для их операционной системы. Выбор был остановлен именно на ней, поскольку она обладает рядом преимуществ:

- Разрабатывается в сотрудничестве с Google, а, следовательно, сразу поддерживает все новые функции обновленных версий Android и методы взаимодействия с ними: например, присутствует расширенный режим отладки

- Удобный Log, поддерживающий раздельный вывод в зависимости от процессов, потоков и приложений

- Удобный и интуитивно понятный интерфейс

- Множество «горячих» клавиш, которые значительно увеличивают время разработки

- Высокая скорость работы интерфейса (в отличии от Eclipse)

- Крайне удобное дерево проектов

- Встроенная система контроля версий, поддерживающая Git

2.8. Выбор языка программирования для разработки сервера

Кроме мобильного приложения проект подразумевает и реализацию удаленного сервера, так как необходимо реализовать клиент-серверную модель взаимодействия.

Для выполнения этой задачи была выбрана связка из двух языков: PHP и Java.

На PHP написана реализация API сервера, с помощью которого выполняются запросы к базе данных MySQL, добавление и получение из неё данных – иными словами, реализуется интерфейс доступа к базе данных.

Серверная часть, разработанная на языке Java, отвечает за более «тяжелые» вычисления: обработку координат треков, их анализ, корректировку и преобразование.

Программы, разработанные на этом языке программирования, предоставляют лучшее соотношение между безопасностью, производительностью и широкой совместимостью. Кроме того, нельзя не отметить изначальную направленность Java на корпоративный сектор, что выливается в широкий набор инструментов для работы с сетью, а также безопасность этой работы. Кроме того, повышенная безопасность обеспечивается и регулярными обновлениями Java Runtime Environment.

2.9. Выбор среды разработки

Для разработки на языке Java мною была выбрана IntelliJ IDEA – именно на её основе развивается Android Studio от Google. Эта среда разработки обладает все теми же вышеназванными достоинствами, привычным удобным интерфейсом. Главное отличие от Android Studio – это отсутствие инструментов, которые предназначены для разработки под операционную систему Android.

Для разработки серверной стороны на языке PHP выбрал удобный и легковесный текстовый редактор Notepad++, он обладает всем необходимым функционалом:

- Подсветка синтаксиса
- Автодополнение
- Форматирование
- Высокая скорость
- Низкие системные требования

2.10. Выбор языка программирования для веб-версии приложения

Разрабатываемый проект также подразумевает и наличие веб-версии приложения, которой можно будет воспользоваться из любого современного браузера. Назначение веб-версии уже мобильной, оно заключается лишь в просмотре уже записанных данных.

Для реализации этой версии был выбран стандартный стек технологий для веб-разработки, а именно: HTML + CSS + JavaScript + PHP.

Также для отображения карты было задействовано Google Map Web API. А для взаимодействия с компонентами HTML прямо из JavaScript кода был использован бесплатный фреймворк с открытым исходным кодом jQuery.

2.11. Обзор существующих систем анализа качества дорожного покрытия

Все существующие решения можно разделить на две категории: первая включает в себя привычный всем «ручной» способ анализа качества, когда проблемные участки обнаруживаются «на глаз» и личные ощущения в процессе объезда какого-либо участка транспортной сети, либо по жалобам других участников движения – такой способ крайне неэффективный: он долгий, имеет низкую степень охвата. Во вторую категорию можно включить автоматизированные методы сборы данных о качестве дорожного покрытия.

Категория автоматизированного сбора информации отличается более оптимальным подходом и лучшими характеристиками: она быстрее, точнее, всеобъемлющее.

Подавляющее большинство методов для автоматизированного определения качества дорожного полотна основана на анализе вибраций, фиксирующимися сенсорами. Эти вибрации вызваны деформациями и дефектами дорожного покрытия. В таком случае

основную часть работы выполняют датчики, устанавливаемые в автомобиле, они фиксируют любые неровности и другие аномалии. Но с ростом популярности смартфонов, также начали разрабатываться и методы анализа посредством использования смартфона в качестве источника данных. Подавляющее большинство методов используют два сенсора телефона: акселерометр и приемник сигнала спутниковой навигации.

Таким образом, категорию автоматизированного сбора сведений можно разделить на две подкатегории: методы с использованием специализированных датчиков и методы с использованием датчиков обычного смартфона.

Методы с использованием специальных сенсоров. Этот метод включает в себя использование набора датчиков, размещаемых в автомобиле, которые и будут производить сбор сведений. Как правило, данные в режиме реального времени отправляются на удаленный сервер, где уже обрабатываются и преобразуются. Эти системы используют модули навигации для определения перемещения машины, её скорости. Для передачи данных на удаленный узел используются любые доступные стандарты, такие как: Wi-Fi, Bluetooth или USB.

В дальнейшем эти сведения могут быть использованы самыми разнообразными приложениями, например, для построения маршрута, путешествий, статистики, выявления проблемных участков и дальнейшего их ремонта.

Pothole Patrol система использует трёх-осевой акселерометр, установленный на приборную панель. Данная система идентифицирует не только выбоины, но и отличает их от других видов аномалий, для этого используются алгоритмы машинного обучения. Записанные сигналы проходят через серию фильтров, каждый из которых разработан с целью определения люков, компенсационных швов и железнодорожных переездов.

Каждая единица данных содержит в себе следующую информацию: текущее время, местоположение, скорость и показания акселерометра.

Общая схема функционирования системы представлена на рисунке 1.

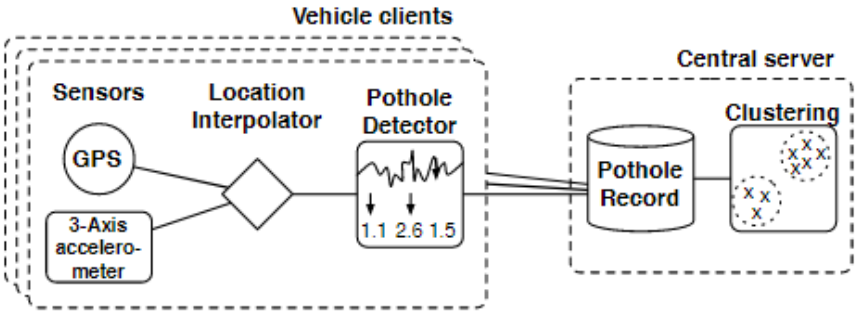

Рисунок 1. Схема функционирования системы Pothol Patrol

Вторая категория систем подразумевает использование смартфона.

В этом случае для обнаружения ям, неровностей и выбоин производится с помощью акселерометра, микрофона, GSM сигналов и приемников спутникового сигнала для позиционирования. В случае использования акселерометра и сотовой связи вместо сигналов GPS и микрофона, повышается энергоэффективность системы, однако, в этом случае снижается точность и качество полученных данных. Одним из недостатков этой системы является необходимость сопоставления осей акселерометра с осями автомобиля перед началом проведения замеров.

Однако существенным минусом всех этих систем является то, что они не объединяют данные от всех пользователей, в то время, как объединение таких сведений может значительно повысить качество отображаемой информации.

Основными преимуществами разрабатываемой системы будет устранение недостатков уже существующих подобных систем:

- Объединение данных с разных замеров

- Произвольное (но обязательно фиксированное) положение смартфона в салоне автомобиля.

3. Разработка структуры системы

Автоматизированная система оценки качества дорожного покрытия состоит из нескольких ключевых частей.

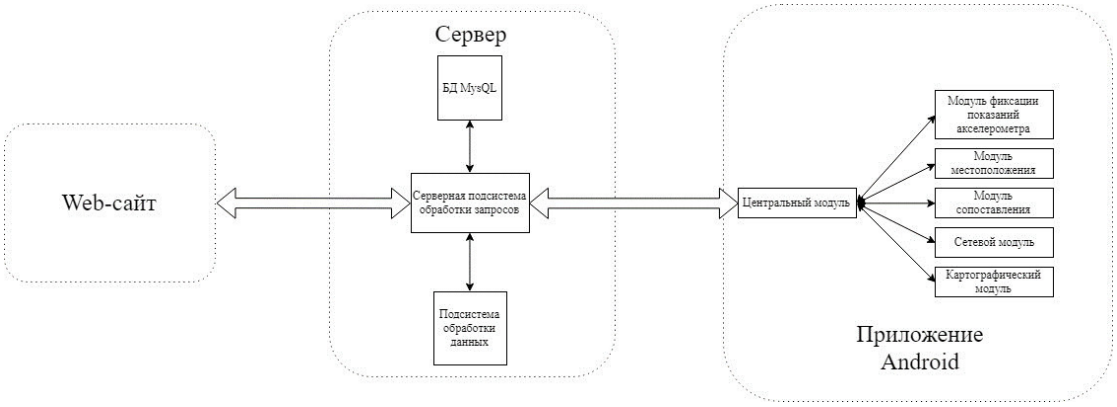

Рисунок 2. Структурная схема разрабатываемой системы

Можно выделить три ключевых составляющих системы:

- Приложение для мобильного устройства (клиентская часть)

- Серверная часть

- Web-сайт

В результате использования клиент-серверной архитектуры были решены сразу несколько важных целей:

- Достижение снижения нагрузка на мобильные устройства (за счет этого возрастает автономность работы устройств). Стоит отметить то, что несмотря на ежегодно возрастающую производительность устройств, обрабатывать на них большие массивы данные, когда этого можно избежать – не лучшее решение.

- Достижение централизованной обработка и хранения данных. Это позволяет отправлять с устройства лишь «сырые» данные, а дальнейшую деятельность по преобразованию информации переложить на сервер.

- Клиент-серверная архитектура гарантирует то, что у всех клиентов будет одинаковая и актуальная информация.

Рассмотрим составляющие в отдельности.

3.1. Структура мобильного приложения

Мобильное приложение реализовано с использование архитектуры в виде модульной структуры, что обеспечивает гораздо большую гибкость и масштабируемость.

Например, при невозможности функционирования какого-либо модуля, вся система не потеряет работоспособность. Так, в случае отсутствия Интернет-соединения программа продолжит исправно функционировать, за исключением исполнения сетевых функций.

Также при желании расширения функциональности программного продукта, будет отсутствовать необходимость в переписывании всего кода приложения. Кроме того, такая структура способствует облегчению совместной работы над проектом в команде: достаточно разделить задачи по модулям, в результате чего, разработчики будут не обязаны знать о внутренних тонкостях реализации того или иного модуля, им необходимо будет подключить модуль и использовать уже готовые внешние интерфейсы для работы с модулем. Таким образом, достигается больший уровень абстракции.

Рассмотрим структуру и назначение всех модулей системы.

Одним из самых важных модулей является модуль обнаружения и анализа вибраций, фиксируемых акселерометром. Его основное предназначение заключается в получении сведений с аппаратного компонента устройства – акселерометра, а также обработке этих сведений.

В это системе можно выделить несколько этапов действий над полученными данными: в первую очередь производится предварительная обработка данных, затем происходит усреднение и объединение, после чего выполняется классификация (принимается решение о причастности к той или иной категории).

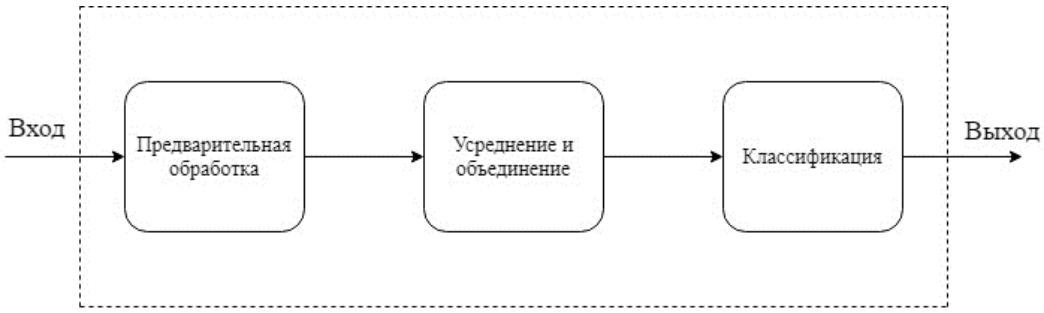

Рисунок 3. Структурная схема функционирования модуля работы с акселерометром

В результате преобразований на выходе из системы имеется усредненный показатель уровня вибраций за 0,5 секунды, с которым производятся все дальнейшие действия.

Далее следует рассмотреть другой не менее важный компонент системы - модуль местоположения.

Основное предназначение этого модуля – предоставление системе информации о текущем местоположении устройства. В результате обращения к аппаратным компонентам устройства, а именно – приёмнику спутникового сигнала навигации, данный модуль получает информацию о текущих координатах, скорости и времени получения сигнала, после чего передает эту информацию центральному модулю.

Следующим по важности является модуль сопоставления данных.

Задача, возложенная на этот модуль, состоит в объединении данных, которые были получены от модуля определения местоположения и модуля фиксации вибраций.

Основная сложность функционирования этого модуля состоит в том, что поток приходящий на него информации неоднороден. То есть может возникнуть такая ситуация, при которой информация о вибрациях за промежуток времени может приходить чаще, чем информация о местоположении. Это может произойти по то причине, что спутниковая навигация допускает погрешность в пределах 5-15 метров, поэтому устройство, переместившись на расстояние в пределах этих значений в реальности, может считаться расположенным на прошлом месте. Таким образом, данные о вибрациях будут обновляться чаще, чем данные о местоположении.

Для того, чтобы решить эту проблему, было решено производить интерполяцию значений местоположения.

Графически эту проблему можно изобразить на временной шкале следующим образом.

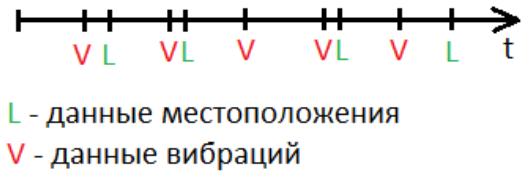

Рисунок 4. Получение данных на шкале времени

Вышеописанная задача выполняется модулем сопоставления данных. Кроме того, еще одна задача, которая выполняется этим модулем, напрямую связана с его названием – модуль объединяет значения вибрации, времени, координаты широты и долготы в одну единую структуру данных, которая выдается на выходе.

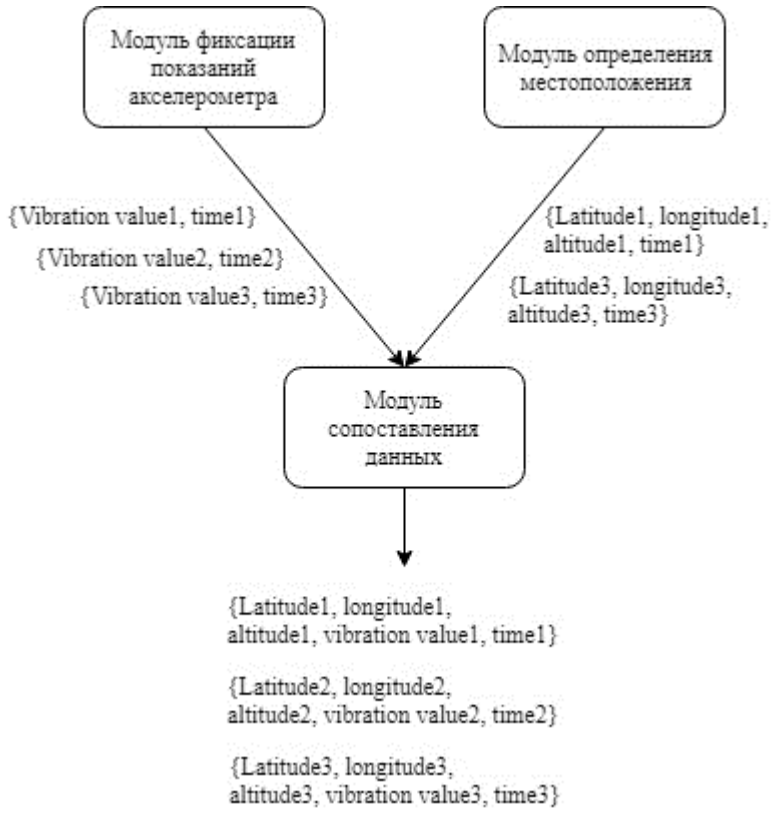

Рисунок 5. Принцип сопоставления данных

Использование обобщенной структуры данных в дальнейшем упрощает работу с информацией и облегчает её хранение.

Сетевой модуль является неотъемлемой частью клиент-серверного приложения.

В задачи этого модуля входит обращение к удаленным ресурсам, получение информации от них, а также загрузка и передача данных.

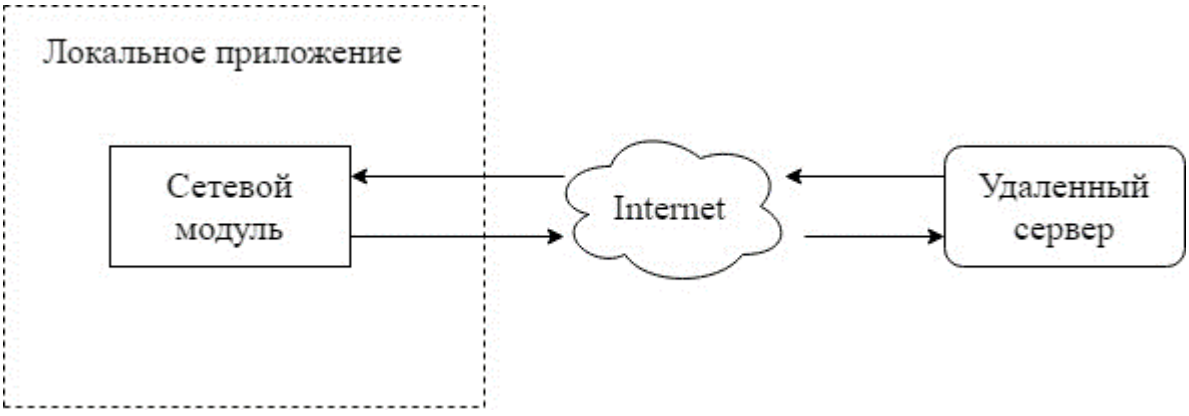

Рисунок 6. Сетевой модуль

Другой не менее важный модуль – это картографический модуль.

Основная, главная его задача – это предоставление объекта «Карта» для отображения её в мобильном приложении.

Кроме того, в задача, которые выполняются картографическим модулем входит и отображения картографических данных на объекте «Карта». Сюда входит как отображение текущего местоположения, так и записанного маршрута.

Последний из «обслуживающих модулей» - модуль доступа к базе данных.

Этот модуль мобильного приложения реализует интерфейс доступа к локальной базе данных, которая располагается на устройстве. Эта база данных используется для хранения информации о треках, которые были записаны на данном устройстве.

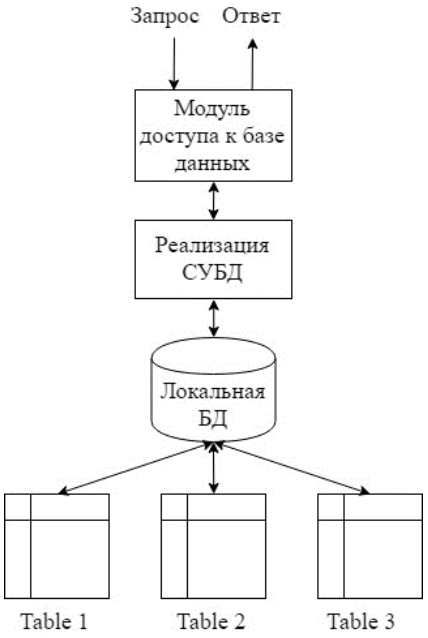

Рисунок 7. Структура модуля работы с БД

Заключительным модулем, но не по значимости, является центральный модуль приложения. Он выступает своего рода «ядром» программы, которое координирует всё взаимодействие между другими независимыми модуля.

В результате организации подобной структуры мобильного приложения удается достичь не только свободы и гибкости, но, что не менее важно, и централизованной структуры – что невероятно удобно в большом проекте.

Все эти архитектурные решения помогают избежать одной из самых важных и сложных проблем: с ростом функциональности, объема кода, как правило, значительно возрастает и сложность проекта, его запутанность, а порой для добавления той или иной функциональности приходится полностью переделывать какие-то готовые участки и уже функционирующие решения.

Модульный же подход позволяет избежать всех вышеописанных проблем и сложностей, что позволяет считать его удачным и правильным решением.

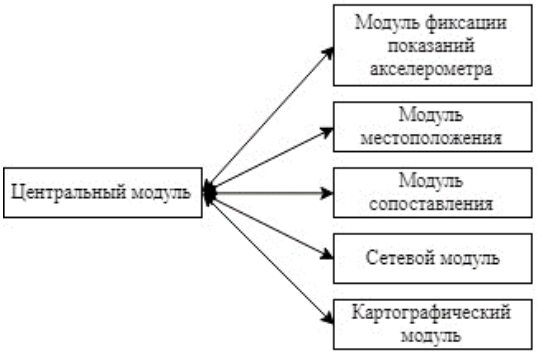

Рисунок 8. Модульная структура Android приложения

3.2. Структура серверной части продукта

Не менее важным и сложным вопросом стала реализация серверной стороны продукта. Стояла задача реализовать как простое и легковесное API для доступа к данным, находящихся в серверной базе данных, так и более «мощную» и тяжеловесную программу для обработки всех поступающих данных.

Создание API для доступа к данным позволило получить несколько весомых преимуществ:

- Обращение к данным из клиентского приложения стало проще и прозрачнее

- В дальнейшем имеется возможность сделать API доступным для всех, что позволит сторонним разработчикам использовать все ранее полученные данные в своих целях

Ряд функций, которые реализуются в API:

- Получение информации о треке по его ID номеру

- Получение списка улиц, по которым был записан трек

- Получение списка узлов, описывающих улицу

- Метод добавления улицы в список

- Метод получения следующего свободного ID для присвоения его треку

- Другие методы (общее количество предоставляемых API методов – 15)

Как видно, использование специализированного API для доступа и редактирования данных на сервере значительно упрощает взаимодействие с удаленным сервером и расширяет возможное его использование.

Но кроме решения задачи упрощения доступа к информации, необходимо было также решить задачу и обработки поступающих на сервер данных.

Для решения этой задачи была разработана серверная программа. В её основные функции входит преобразование «сырых» исходных данные, которые были получены от

клиента. В результате этих преобразований на выходе мы имеем полностью скорректированную и готовую для отображения информацию об обработанном маршруте.

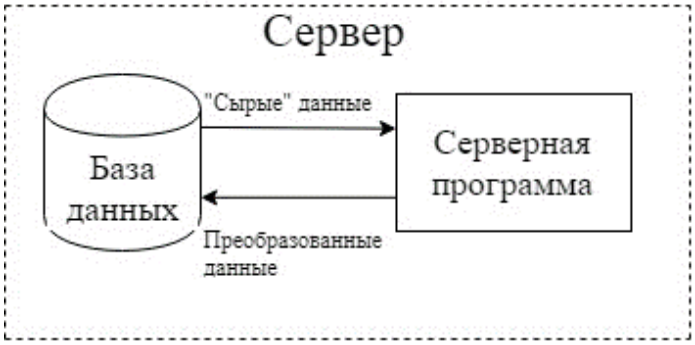

Рисунок 9. Серверная сторона программного продукта

3.3. Признаки классификации

Для того, чтобы принять решение о принадлежности того или иного участка дороги к определенной категории качества дорожного полотна, необходимы заранее установленные категории качества и методы оценки принадлежности к ним.

Для этого была введена следующая градационная шкала:

Таблица 1. Градационная шкала

Состояние	Значение ускорения a, м/с^2
Отличное	$a <= 1$
Хорошее	$1 < a <= 2$
Удовлетворительное	$2 < a <= 3$
Плохое	$3 < a <= 4$
Ужасное	$a > 4$

Эти оценки показателей были установлены путем проведения нескольких тестовых заездов по дорогам, имеющим разное качество. После чего были скорректированы на основе ощущения пассажирами качества дорожного полотна, по которому двигался автомобиль.

Характеристики каждого из возможного состояния:

- Отличное

Рисунок 10. Дорога отличного качества

Идеально ровная дорога, не имеющая видимых и акцентных дефектов и неровностей на своей поверхности.

- Хорошее

Рисунок 11. Дорога хорошего качества

В целом ровные дороги, имеющий на поверхности продольные или поперечные трещины, величина которых не превышает 5-10 мм.

- Удовлетворительное

Рисунок 12. Дорога удовлетворительного качества

Дорога, имеющая хорошо заметные и ощущаемые дефекты покрытия, возможно наличие выбоин и неровностей, глубина которых достигает 10 – 25 мм.

- Плохое

Рисунок 13. Дорога плохого качества

Дорога, имеющая неровности глубиной более 25 мм, возможно частичное отсутствие твердого покрытия. Также эта характеристика описывает и дороги, выполненные из щебня:

Рисунок 14. Дорога из щебня

- Ужасное

Рисунок 15. Дорога ужасного качества

Дорога, по которой опасно двигаться на скорости, превышающей 10-15 км/ч, отличается большим количеством и частотой неровностей, ям, выбоин, глубина которых превышает 25 мм.

3.4. Разработка алгоритма усреднения и объединения

Применение алгоритма для усреднения и объединения стало необходимостью вследствие особенностей работы акселерометра, которая заключается в том, что частота получения данных достаточно велика – она может достигать частоту одного измерения в 5 - 10 миллисекунд. Также стоит отметить то, что повышением частоты измерения достигается и возрастание точности измерения.

Таким образом, учитывая вышеописанную особенность, было принять решение комбинировать показания за установленный промежуток времени. Этот промежуток времени был выбран равным 0,5 секунды.

Изначально для объединения было принято решение использовать нахождение среднего значения среди всех полученных показателей за период измерения (то есть за время равное 0,5 секунды):

$$a = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}(x_1 + \cdots + x_n)$$

,где a – усредненный показатель за период времени

n – число измерений за период времени

x_i – i-тое измерение уровня вибраций

Однако использование такого способа не всегда верно отражало реальное состояния дороги. Рассмотрим пример, в котором такой способ оказывался непоказательным.

Пусть за промежуток времени 0,5 секунд удалось получить следующие данные:

Таблица 2. Данные, полученные с акселерометра

x, м/с^2	0,2	0,13	4	4,1	0,1	3,9	4,17	4,21	0,11	0,09

Тогда согласно вышенаписанной формуле получим следующий результат:

$$a = \frac{1}{n}\sum_{i=1}^{10} x_i = \frac{1}{10}(0{,}2 + 0{,}13 + 4 + 4{,}1 + 0{,}1 + 3{,}9 + 4{,}17 + 4{,}21 + 0{,}11 + 0{,}09) = 2{,}101$$

В соответствие с классификацией такой участок дороги следует отнести к «Удовлетворительному»:

$$2 < 2{,}101 \leq 3$$

Однако в реальности состояние этого участка дороги будет желать лучшего, по нему будет дискомфортно и опасно двигаться, даже несмотря на наличие «Отличных» показаний акселерометра.

Из-за этого недостатка потребовалось избрать более подходящий алгоритм, который отражал бы реальную состояние качества дорожного полотна.

В результате этого было принято решение ввести дополнительные весовые коэффициенты, которые бы корректировали приоритетность выбора.

Для каждой из категории состояния качества дорог был выбран соответствующий коэффициент, который бы увеличивал или снижал значимость.

Опытным путем были выбраны следующие коэффициенты:

Таблица 3. Весовые коэффициенты

Состояние	Весовой коэффициент
Отличное	0,25
Хорошее	0,33
Удовлетворительное	0,5
Плохое	1
Ужасное	2

После ввода коэффициентов, для определения усредненного показателя уровня вибрации за период времени (0,5 секунды), можно использовать следующий алгоритм действий:

1) Определить количество измерений, относящихся к той или иной группе в соответствие с приведенной классификацией;

2) Произвести расчет по следующей формуле:

$$R_i = W_i * n_i$$

, где R_i – итоговый показатель

W_i – весовой коэффициент, соответствующий i-той категории состояния дорожного полотна

n_i – количество измерений, соответствующих i-той категории за отведенный период времени

3) Провести сравнение всех итоговых показателей и принять решение в пользу того, значение которого окажется наибольшим

В результате вводов весовых коэффициентов более высокий приоритет получили отрицательные показатели, которые описывают дорожное покрытие, которое имеет худшего состояния.

Теперь проведем расчеты по новому алгоритму, используя данные, которые приведены в таблице 2, после чего сравним результаты расчетов и решение, которое будет принято в итоге.

Выполним алгоритм по шагам:

1) Определяем количество измерений в той или иной категории.

В данном случае измерения имеются только в трех категориях: «Отличное», «Плохое», «Ужасное»:

«Отличное» - 5 измерений

«Плохое» - 1 измерения

«Ужасное» - 4 измерения

2) Произвожу расчет по формуле:

$R_1 = 0{,}25 * 5 = 1{,}25$

$R_4 = 1 * 1 = 1$

$R_5 = 2 * 4 = 8$

3) Произвести сравнение:

$R_4 < R_1 < R_5$

Таким образом, данный участок дороги будет отнесен к категории, имеющей «Ужасное» качество дорожного полотна.

Как видно, в результате использования нового алгоритма были применены более критичные методы оценок участка, в результате чего вместо «Удовлетворительной» оценки была получена самая негативная из возможных – «Ужасное» качество дорожного полотна.

3.5. Алгоритм сопоставления данных

Другим важным алгоритмом, которому следует уделить не меньшее внимание, является алгоритм, который отвечает за сопоставление данных: данных местоположения (включающего в себя широту, долготу, а также временной штамп) и данных, которые были получены с акселерометра (они включают в себя непосредственно показания акселерометра и штамп времени).

Основная задача этого алгоритма – совместить два разных вида данных в одну единую структуру.

Сложность, с которой пришлось столкнуться заключается в том, что поток данных не является однородным. Все показания с датчиков приходят в произвольные моменты времени, что значительно усложняет работу по их сопоставлению в одну единую структуру.

Для решения этой задачи был разработан алгоритм с применением интерполяции.

Выбор был сделан в пользу интерполяции местоположения по нескольким важным причинам:

1) Местоположение с использованием датчиков спутниковой навигации в мобильном устройстве само по себе уже имеет некоторую погрешность (от 5 метров и более, в зависимости от погодных условий, количества доступных спутников, находящихся рядом отражателей сигнала в виде высоких зданий). Поэтому пренебрежение величиной, уже имеющей в себе заложенную некоторую погрешность является более предпочтительным в сравнении с точными показаниями акселерометра.

2) Если расстояние между двумя точками местоположения не столь велико, то предсказать примерные координаты, расположенные между двумя этими точками и находящимися в определенном временном интервале, не составляет никакого труда. В то же время интерполяция значений показания акселерометра является практически нереализуемой задачей. Пусть имеется две порции данных, полученные в моменты времени t_1 и t_2 соответственно. В момент времени t_1 показатели акселерометра были 0,5 м/с^2, а в момент времени t_2 показания равнялись 4,5 м/с^2. На основании этих данных невозможно предсказать значения, которые были на интервале времени $t_1 – t_2$, можно предположить, что значение было 2,5 м/с^2, но оно могло быть равным как 7,3 м/с^2, так и любым другим.

Таким образом, по вышеназванным причинам выбор был сделан в пользу интерполяции местоположения.

Данный алгоритм реализуется в модуле «Совмещения и сопоставления» данных. Рассмотрим его подробно:

1) Проверяются флаги получения данных двух различных типов: местоположения и вибрации. Если данные были получены, то переходим на следующий шаг, иначе необходимо ожидать получения.

2) Проверяется количество данных о местоположении, для выполнения интерполяции и дальнейшего совмещения в единую структуру, данные навигации должны поступить минимум два раза. Если в распоряжении имеется минимум две порции данных, то переходим на следующий шаг. В противном случае необходимо вернуться на шаг алгоритма под номером 1.

3) Определяется количество данных вибраций, полученных за промежуток времени между получением последнего известного местоположения в момент времени t_2 и предшествующему последнему в момент времени t_1. Количество полученных показателей акселерометра напрямую влияет на количество точек, которые будут интерполированы.

4) Выполняется вычисление изменения времени между двумя точками местоположения:

$$dt = t_2 - t_1$$

5) Выполняется расчет расстояния между двумя точками местоположения, для этого используется три доступные координаты: широта, долгота, высота.

6) После чего находится скорость приращения каждой из координат за единицу времени: dLongitude, dLatitude, dAltitude.

7) Затем находится первая из порций данных акселерометра, находящихся в промежутке временного интервала t_1 - t_2. Вычисляется разность между временем, когда были получены данные по вибрации, и временем t_1.

8) Полученная на шаге 7 разность умножается на скорость приращения каждой из координат, которая была получена на шаге 6. В результате выполнения этих действий были вычислены координаты местоположения (широта, долгота, высота) для полученных данных с акселерометра с учетом времени их получения.

9) Выполняется объединение данных местоположения с данными акселерометра, соответствующим им, на выходе имеем единую структуру данных, совмещающую в себе данные местоположения и ускорения.

10) Увеличить счетчик обработанных показаний акселерометра, перейти на шаг 7. Когда все показания будут обработаны, перейти на следующий шаг.

11) Сбросить флаги готовности новых данных, ожидать их получения.

В результате выполнения этого алгоритма на выходе мы имеем набор данных, который включает в себя объединенные показатели модуля навигации и акселерометра.

3.6. Алгоритм построения пройденного маршрута.

Изначально эта задача кажется достаточно тривиальной и простой, однако, и она имеет несколько сложностей с которыми пришлось столкнуться и решить их.

Исходные данные включают в себя список структур данных, которые необходимо отобразить на карте, используя координаты местоположения и показания акселерометра для окраски отрезка в соответствующий цвет.

Изначально было применено самое простое решение, находящееся на поверхности. Сущность этого решения заключалась в построении отрезков от первой координаты ко второй, от второй к третьей, и так далее.

В результате было получено следующее изображения маршрута, нанесенное на карту дорожной сети.

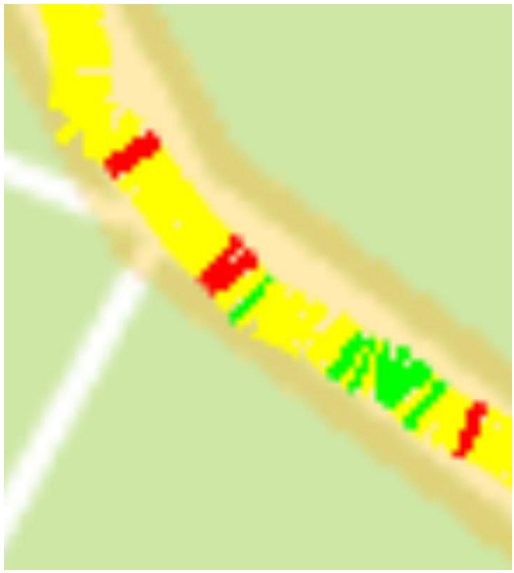

Рисунок 16. Нанесенные на карту данные

Как видно, каждый небольшой отрезок пути, длина которого не превышает примерно 0.5 - 1 метра, является отдельным и самостоятельным объектом на карте.

В результате этого спустя всего несколько минут записи маршрута, производительность снижалась до недопустимо низкого уровня: наблюдались торможения интерфейса, снижение отзывчивости и плавности. Это происходило по той причине, что даже спустя короткий промежуток времени количество таких самостоятельных объектов насчитывалось уже несколько десятков тысяч.

Для снижения количества объектов для рисования на карте, было принято использовать одно не совсем очевидное свойство: дорога, как правило, однородна на достаточно протяженно участке. То есть, если дорога ровная, то очень велика вероятность того, что она будет ровная и через 10 метров, и через 100, и, возможно, даже через 1000.

Поэтому был использован следующий оптимизированный алгоритм:

1) Строится отрезок между первыми двумя точками, окрашивается в соответствующий цвет.

2) Категория, к которой был отнесен участок дороги, записывается.

3) Берутся следующие две точки, сверяется категория их отрезка с тем, что находится в памяти. Если совпадают, то данные точки добавляются в один объект к предыдущему отрезку, образуется ломаная линия. Если не совпадают, то создается новый объект отрезка, окрашивается в необходимый цвет. Снова выполняется шаг 3.

В результате выполнения данного алгоритма происходит значительная экономия вычислительных мощностей и памяти, а количество объектов снижается в тысячи раз.

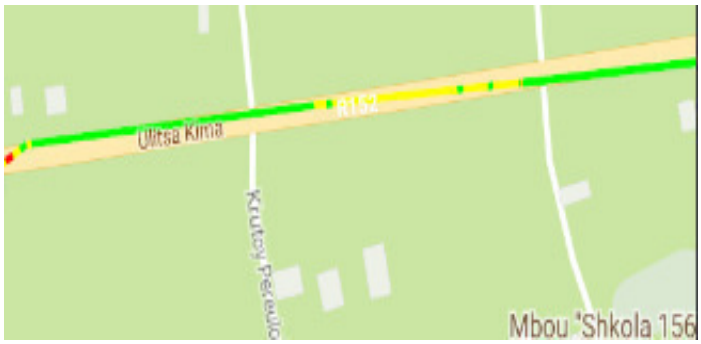

Рисунок 17. Оптимизированное нанесение данных на карту

3.7. Алгоритм объединения нескольких треков на одном участке

Так как разрабатываемая система является ориентированной на клиент-серверное взаимодействие, она подразумевает наличие сразу нескольких активных клиентов. В таком случае есть большая доля вероятности того, что два клиента проедут по таким маршрутам, которые будут частично иметь общие участки на маршруте. Более того, такая ситуация возможна и с одним единственным клиентом. В том случае, если клиент проедет по одному и тому же маршруту несколько раз.

Ситуации, которые подобны вышеописанным, необходимо обрабатывать особым образом и уделить им особое внимание.

Существует два пути решения этой задачи. Первый заключается в полном игнорировании измерений всех других треков, которые проходят по одному и тому же маршруту.

Плюсы этого подхода:

- Простой, легкий в реализации. Берутся данные одного произвольного трека и отображаются на нужном участке.

Но у такого подхода есть и существенные минусы:

- Как правило, чем больше выборка, тем более высокого уровня точности удается достичь. Таким образом, выбирая и учитывая показания только одного трека из нескольких доступных, мы пренебрегаем ценной информацией, содержащейся в других записанных треках.

Второй возможный подход состоит в том, чтобы учитывать и сопоставлять все треки и данные, которые имеются в распоряжении.

Преимущества и недостатки этого метода полностью противоположны первому.

Недостатки:

- При использовании этого метода в разы возрастает сложность реализации, появляется необходимость в использовании трудных алгоритмов, возрастают требования к вычислительным мощностям.

Преимущества:

- В результате применения второго метода, учитываются абсолютно все имеющиеся данные. Итоговая информация содержит в себе объединение каждого из треков. Мы используем больше данных, а значит вместе с возрастанием данных увеличивается и точность информации на выходе.

После оценки сильных и слабых сторон каждого из возможных методов, было принято решение приступить к реализации именно второго метода, поскольку в приоритете находится точность отображаемых данных, а не простота решения.

Первая проблема, с которой приходится столкнуться – это то, что даже в том случае, если несколько треков абсолютно совпадают по пройденному маршруту, можно сказать со стопроцентной уверенностью то, что точки, описывающие эти маршруты, будут иметь разные координаты широты, долготы и высоты. Эта проблема проиллюстрирована на рисунке ниже.

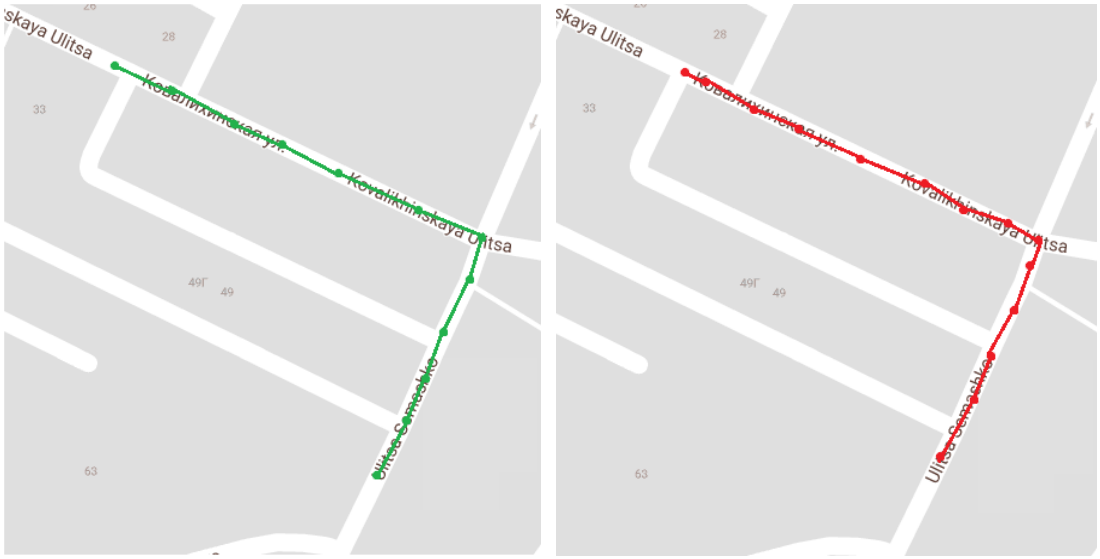

Рисунок 18.. Проблема несовпадения точек координат на одному маршруте

Как видно из рисунка, оба трека описывают одинаковые маршруты, но точки, полученные системой навигации, находятся в разных местах.

Для разрешения этой проблемы изначально было принято решение использовать метод растеризации для сопоставления треков.

Его суть состоит в некотором «огрублении» координат треков. Этот метод похож на отрисовку прямых линий, имеющих точные координаты, на пиксельном экране.

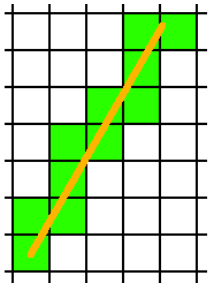

Рисунок 19. Растеризация прямой линии на сетке пикселей

Это позволит абстрагироваться от точных координат точек и перейти на более высокий уровень – работу с секторами сетки.

Для этого на карту накладывается координатная сетка с определенным заданным шагом (наиболее подходящий шаг – шаг, равный допустимой погрешности определения координат местоположения, для GPS это около 5-15 метров).

После наложения координатной сетки необходимо применить алгоритм, который решает задачу аппроксимации отрезка на дискретном графическом устройстве. Одни из самых известных алгоритмов такого типа является алгоритм Брезенхэма –

оптимизированный алгоритм, который использует целочисленную арифметику и только операции сложения и вычитания.

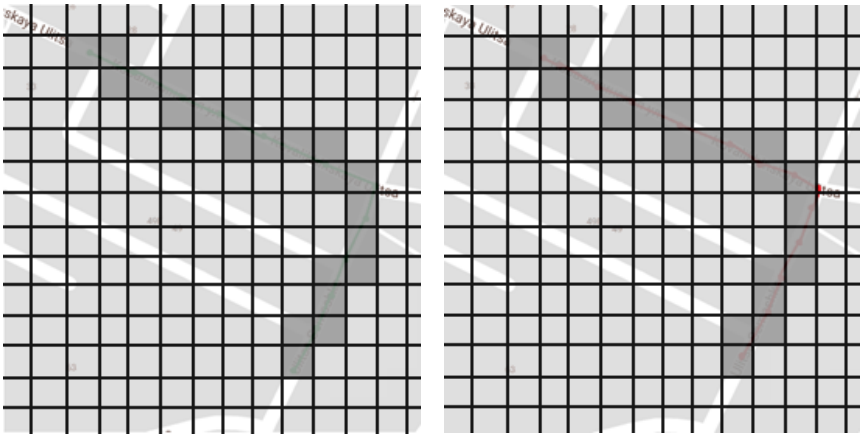

Рисунок 20. Применение растеризации к маршрутам на карте

На рисунке представлен алгоритм работы для двух треков, которые изображены на рисунках выше.

Как видно, в таком случае произвести сравнение треков и участков их пересечения становится значительно проще.

Однако в результате тестирования оказалось, что этот метод достаточно требователен к производительности, поэтому было принято решение искать более подходящее альтернативное решение.

После детального рассмотрения структуры карты, методов её построения и взаимодействия с ней, стало понятно, что можно взять за основу структуру дорог, которые представляют собой группу точек-узлов, соединенных между собой отрезками. Пример представления дороги показан на рисунке 21.

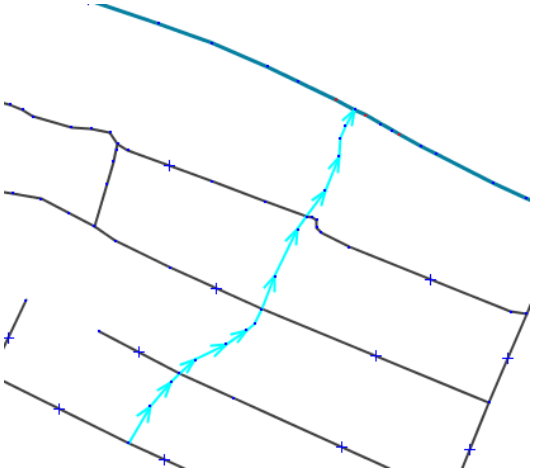

Рисунок 21. Представление дороги в картографической системе

Исходя из особенностей представления дороги, можно за основу взять точки, которые формируют её геометрию.

Таким образом, зная все ключевые узлы-точки дороги, мы можем точно определить, принадлежит ли данный трек этой дороге или её какому-то участку.

Однако на этом этапе появляется еще одна трудность, которая опять же связана с точностью получения координат местоположения.

В процессе движения транспортного средства и записи трека, получаемые координаты местоположения совсем необязательно будут располагаться именно на отрезке, соединяющем две узловые точки дороги на карте, либо попадать в них.

Для решения этой проблемы использовалось общедоступное API сервиса Open Source Routing Machine, в основе которого лежит взаимодействие с OSM – некоммерческим веб-картографическим проектом по созданию свободной и бесплатной географической карты мира.

Одна из возможностей сервиса OSRM – это привязка полученных координат к карте дорожной сети наиболее правдоподобным способом.

Результат запроса к API этого сервиса представлен на рисунке 22.

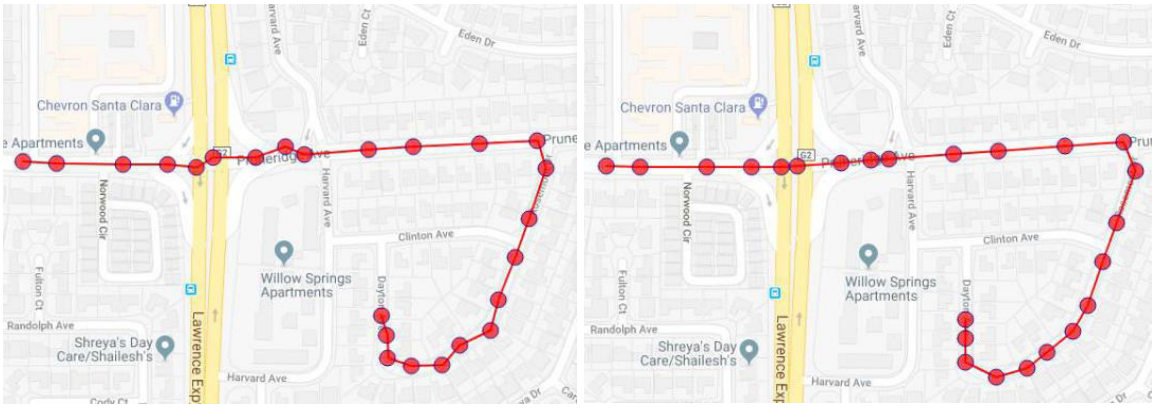

Рисунок 22. Результат обработки точек трека и их привязки к дороге

На рисунке 22 в левой части представлен набор координат, который был получен с клиентского устройства, как видно, точки не всегда принадлежат дороге, некоторые расположены в стороне от нее. На правой части изображении в результате преобразований все точки следуют ровно по дороге.

Кроме того, сервер также возвращает и список с коленями (legs) дороги, которые характеризуют отрезок пути, по которому соединяются два соседних точки координат. Данный список можно использовать для того, чтобы определить, между какими узлами-точками дороги расположена корректируемая координата. Когда такие узловые точки прямой, можно определить, к какой прямой (дороге) принадлежат эти точки.

В картографической системе каждая дорога имеет свой уникальный идентификатор. Таким образом, можно составить список дорог, по которым был проделан путь каждого из клиентов. После чего все данные сохраняются в базе данных.

Таким образом, алгоритм выглядит следующим образом:

1) Выполняется привязка точек к дороге

2) Для каждой из привязанных к дороге точек определяю уникальный идентификатор дороги, к которой она привязана

3) Все полученные данные о точках и уникальных идентификаторов дороги сохраняются в единую базу данных

В БД также хранится список всех дорог (их уникальные идентификаторы), по которым хотя бы раз записывался трек каким-либо из клиентов. Кроме этого списка в БД храню составляющие каждой из дорог – узловые точки дороги.

Когда нужно отобразить треки, наложенные на карту дорожной сети, происходит обращение к БД со списком дорог, на которые нужно наложить информацию, если в базе данных есть нужная дорога, то запрашиваю её узловые точки, если такой дороги в базе нет, значит, что там никто и не проезжал.

Далее происходит разбиение расстояние между узловыми точками на равные участки протяжённостью примерно по 20 метров, в результате этого разбиения возникают «вспомогательные точки», которые описывают каждый из участков.

После чего запрашиваются все точки треков, у которых указан уникальной идентификатор нужной дороги. После чего необходимо проанализировать все точки полученные точки, если расстояние между этими точками и «вспомогательными точками» меньше 3 метров, то происходит привязывание к «вспомогательным точкам».

Графически это можно представить следующим образом.

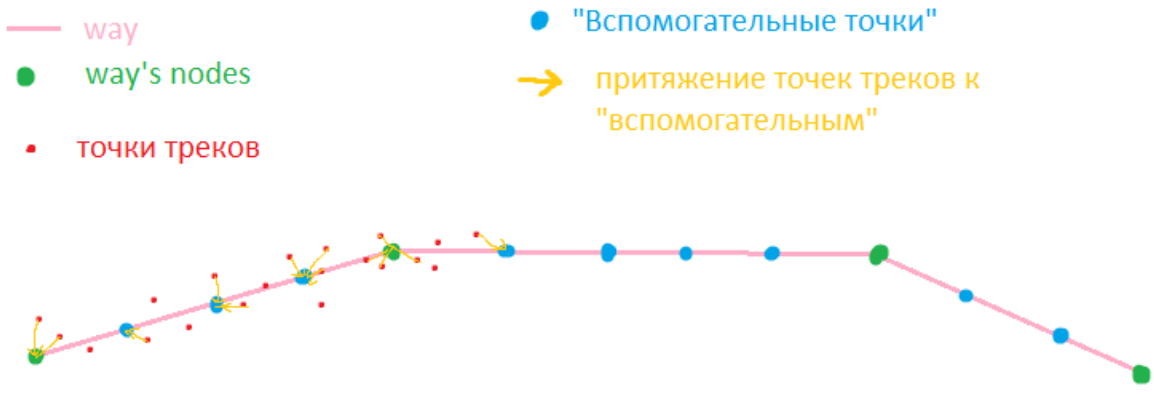

Рисунок 23. Объединение данных, полученных из нескольких источников

4. Разработка программных средств

4.1. Общие сведения о программной реализации

Система мониторинга качества дорожного покрытия работает следующим образом: клиентские устройства записывают маршрут своего передвижения, который затем передается на сервер, после чего сервер обрабатывает поступившие данные и готовит их для отображения.

Схема работы представлена на рисунке 24.

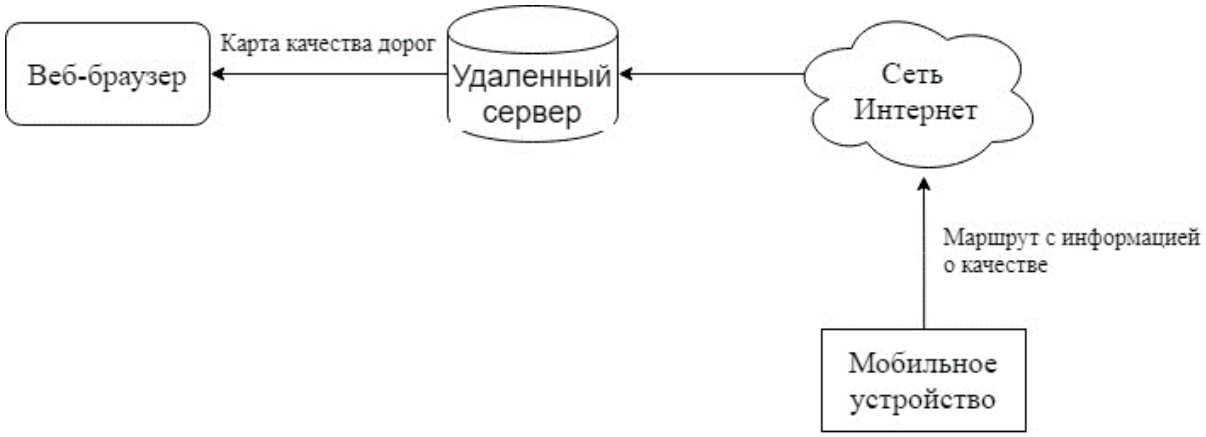

Рисунок 24. Схема работы

Удаленным сервер, на котором производятся вычисления, может выступать компьютер на любой операционной системе, поддерживающей виртуальную машину Java.

Программа, производящая вычисления, представляет из себя простое консольное приложение, не имеющее графического интерфейса.

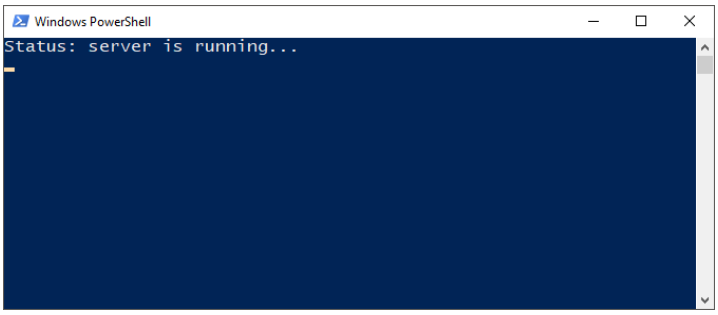

Рисунок 25. Исполняющаяся серверная программа

Для записи маршрута и анализа качества дорожного покрытия используется Android приложение, имеющее три главных вкладки.

На первой вкладке находится карта и кнопка управления.

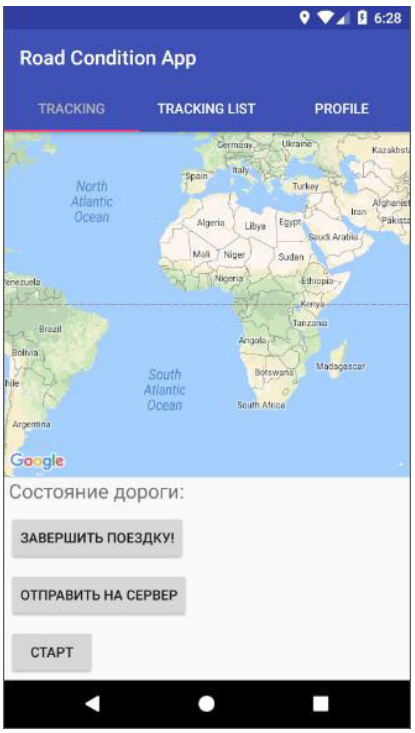

Рисунок 26. Главный экран мобильного приложения

На втором экране располагается список треков, которые были записаны на данном устройстве, с краткой информацией по каждому из треков: время и дата записи, протяженность маршрута и длительность записи.

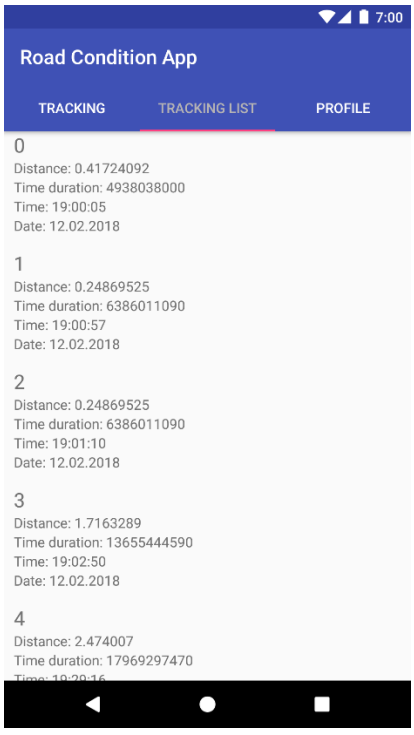

Рисунок 27. Второй экран со списком записанных треков

4.2. Особенности реализации мобильного приложения

Рассмотрим особенности реализации мобильного приложения. В процессе разработки пришлось столкнуться с рядом трудностей и решить их. Часть из них уже была рассмотрена в разделах с обзором используемых алгоритмов, другую же часть рассмотрим сейчас.

Первое, с чем пришлось столкнуться – это то, что все данные, получаемые с акселерометра, находятся в системе координат устройства, которая изображена на рисунке 28.

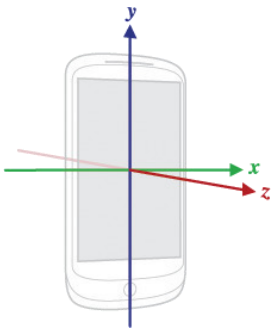

Рисунок 28. Система координат устройства

Исходя из этого, было два возможных пути решения этой проблемы: либо обязать пользователя располагать устройство определенным образом, либо производить расчет из одной системы координат в другую (из системы координат устройства в мировую, где ось Z направлена вертикально вверх).

Для большего удобства использования был выбран второй вариант.

Чтобы выполнить задачу необходимо располагать данными о положении устройства в пространстве в данный момент времени, для этого необходимо знать три величины: крен, тангаж и азимут (Roll, Pitch, Azimuth).

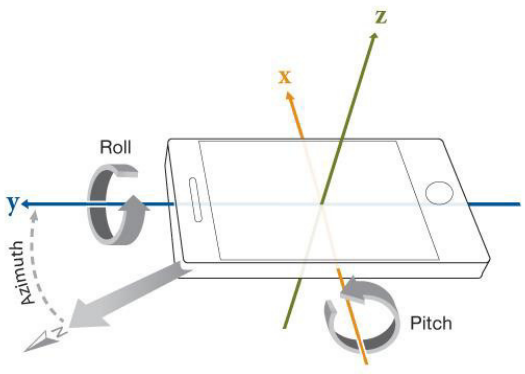

Рисунок 29. Характеристики расположения устройства в пространстве

Первые две величины можно рассчитать из показаний акселерометра, для определения азимута необходимо использовать магнитный компас устройства.

Для расчета этих величин была использована стандартная функция Android API - getRotationMatrix().

Результатом исполнения этой функции является матрица наклона, а также матрица вращения, преобразующая вектор из системы координат устройства в систему координат мира, которая определяется как прямой ортонормированный базис (ортогональный базис — базис, составленный из попарно ортогональных векторов).

Таким образом, был выполнен переход из системы координат мобильного устройства в мировую систему координат.

Другой важной особенностью мобильного приложения является то, что оно выполняется в нескольких потоках. Всего их три: основной, в котором происходит отрисовка графики и взаимодействие с интерфейсом пользователя, поток, в котором происходит трекинг и отслеживание маршрута, а также поток, в котором выполняются сетевые запросы.

Необходимостью выделения функции отслеживания маршрута послужила особенность жизненного цикла приложений в операционной системе Android.

Все дело в том, что операционная система Android, несмотря на то, что пытается сохранять состояние процесса приложения как можно больше времени, в конечном итоге будет вынуждена удалить старые, по её оценке, процессы, чтобы освободить память для новых. Для оценки важности процессов используется иерархия важности, процессы с наиболее низким уровнем важности удаляются в первую очередь, затем следующие по важности и так далее. Всего можно выделить пять типов процессов в порядке важности:

1) Процессы переднего плана – процесс, с которым пользователь непосредственно взаимодействует в настоящее время. Как правило, в каждый период времени работает только один процесс переднего плана.

2) Видимые процессы – те процессы, которые не содержат в себе компонентов переднего плана, однако по-прежнему могут влиять на отображение на экране.

3) Служебный процесс – процесс, который выполняют службу (сервис).

4) Фоновый процесс – процесс, содержащий действия, которые не видны пользователю (зачастую это процессы переднего плана, поставленные на паузу).

5) Пустой процесс - процесс, не содержащий никаких компонентов активного приложения. Единственная причина сохранять процесс такого типа — это кэширование, которое улучшает время следующего запуска компонента в этом процессе.

По умолчанию все компоненты одного приложения работают в одном процессе – главном потоке. Таким образом, если не выделять функцию записи маршрута в отдельный

поток, то есть достаточно большая вероятность того, что, когда пользователь свернет приложение и будет ожидать от него работы в фоновом режиме, система Android прекратит его выполнение, т.к. приоритет процесса изменяется с 1 уровня сразу на 4.

Службами являются компоненты приложения, которые могут выполнять длительные операции в фоновом режиме и не содержат пользовательского интерфейса.

Выделение функции записи маршрута в отдельную службу (и в отдельный поток) позволяет увеличить вероятность исправной и корректной работы приложения в свернутом виде на протяжении большого периода времени. Т.к. приоритет такой службы будет на уровне 3. Кроме того, для еще большего поднятия приоритета данной службы, был применен вывод уведомления в центр уведомления. Этот прием позволяет поднять значимость процесса еще выше, т.к. пользователь осведомлен о процессе выполнения приложения в панели уведомлений, и просто так «убить» этот процесс система не может.

Службы существуют двух типов: запущенные и привязанные. Их ключевая разница заключается в том, что первая после запуска продолжает работать так долго, на сколько это возможно, но не предоставляет интерфейса для взаимодействия со службой из основного потока. Вторая же служба отличается тем, что предоставляет удобный интерфейс взаимодействия с данными службой, однако её жизненный цикл ограничен сроком жизни вызвавшего её компонента, т.е. в случае остановки основного процесса, и эта служба прекратит своё исполнение.

Как видно, обе эти службы содержат в себе критичные для данного приложения качества – долговременность и интерфейс взаимодействия с службой. На основании этого было принято решение реализовать функцию трекинга в виде гибридной службы: запущенной и привязанной одновременно.

Это решение усложнило реализацию, однако обеспечило приобретение двух ключевых качеств: сервис обладает максимально возможным приоритетом и «живучестью», работает максимально долго, но при этом имеет и интерфейс взаимодействия с основным потоком.

Работа и хранение треков на мобильном устройстве

Для удобства работы с треками был разработан ряд структур данных.

Первое, для чего возникла необходимость создания самостоятельной структуры – это для работы с данными, получаемыми с аппаратных компонентов устройства: акселерометра и системы навигации. Для этого были разработаны две структуры данных.

Первая структура разрабатывалась для хранения данных о вибрациях, получаемых с акселерометра:

```
public class AccelerationData {
private long time;
private double vibration;

AccelerationData(double vibration) {}

public AccelerationData(double vibration, long time) {}
        public long getTime() {}

        public double getVibration() {}
}
```

Этот класс содержит два поля: первое служит для хранения полученного с акселерометра значения ускорения (вибрации), а второе – для записи времени получения этого измерения. Тип поля для хранения выбран исходя из того, что получаемое время измеряется в миллисекундах и переменная типа int не смогла бы вместить в себя необходимые значения. Кроме того, было создано два конструктора и два метода для обращения к private полям класса.

Другой структурой является класс для хранения сведений о местоположения:

```
public class LocationData {
private double lat;
private double lon;
private double alt;
private long time;

public LocationData(double lat, double lon, double alt, long time) {}

public LocationData(double lat, double lon, double alt) {}

public LatLng getLatLng() {}

private double degreesToRadians(double degrees) {
   return degrees * Math.PI / 180;
}

public double distanceInKm (LocationData destination) {
   double earthRadiusKm = 6371;

   double dLat = degreesToRadians(destination.lat - this.lat);
   double dLon = degreesToRadians(destination.lon - this.lon);

   double lat1 = degreesToRadians(this.lat);
   double lat2 = degreesToRadians(destination.lat);

   double a = Math.sin(dLat/2) * Math.sin(dLat/2) +
         Math.sin(dLon/2) * Math.sin(dLon/2) * Math.cos(lat1) * Math.cos(lat2);
   double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
   return earthRadiusKm * c;
```

```
    }

    public long getDuration(LocationData destination) {}

    public double getLat() {}

    public double getLon() {}

    public double getAlt() {}

    public long getTime() {}
}
```

Класс для хранения координат местоположения несколько сложнее, он содержит в себе 4 поля: три для записи координат местоположения (широты, долготы и высоты), и еще одно поле для записи времени получения этих сведений. Набор методов кроме двух конструкторов и нескольких методов для получения доступа к private переменных, расширен еще одним методом для нахождения расстояния между двумя координатами. Сам алгоритм основан на нахождении длины окружности, однако, с небольшим послаблением – форма Земли принимается как сфера.

Кроме метода нахождения расстояния между двумя точками по координатам, имеется и вспомогательный private метод для перевода градусов в радианы.

5. Тестирование

Для тестирования было выбрано два участка двух разных дорог.

Первый тестируемый участок является частью автомобильной дороги М-7 «Волга». Изображение участка представлено на рисунке 30.

Рисунок 30. Участок автодороги «Волга»

Как видно, дорога ровная, ямы, трещины и другие дефекты отсутствуют. В результате тестового заезда были получены результаты, представленные на рисунке 31.

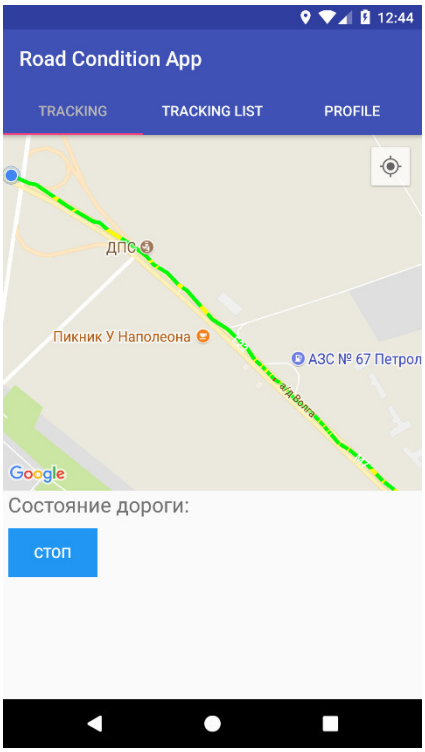

Рисунок 31. Результат тестирования №1

На рисунке 31 видно, что преобладают цвета зеленого цвета с редкими желтыми вкраплениями, что полностью соответствует ожидаемым результатам и состоянию дороги.

Для второго тестового заезда была выбрана дорога худшего качества – улица Зеленая. Она представлена на рисунке 32. На ней присутствуют ямы, люки, неровности.

Рисунок 32. Улица Зеленая

В результате тестирования были получены результаты, изображенные на рисунке 33.

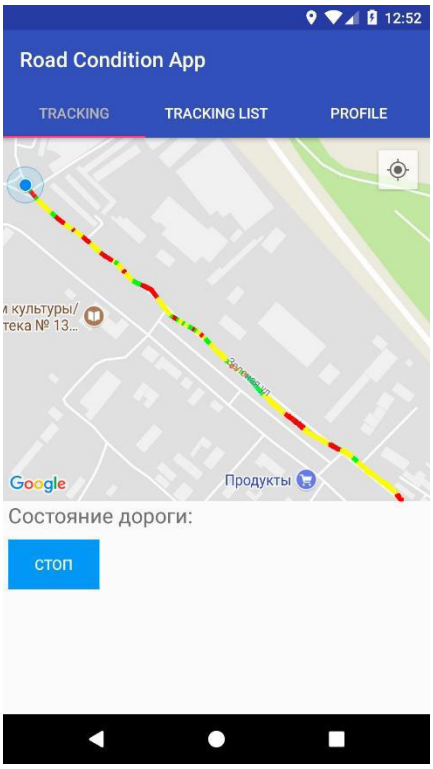

Рисунок 33. Результаты тестирования №2

На рисунке 33 видно, что на пройденном маршруте преобладают желтые и красные цвета. Такие результаты характерны для дороги качества ниже среднего. Движение по такой дороге достаточно некомфортно, кроме того, возрастает риск дорожно-транспортных происшествий.

Как видно из результатов тестирования, были получены данные, достаточно точно описывающие качество дорожного покрытия на каждом из тестируемых участков дорог.

Заключение

В результате выполнения выпускной квалификационной работы была спроектирована и реализована система мониторинга качества дорожного полотна, которая включает в себя три ключевых компонента: мобильное приложение для операционной системы Android, серверная программа с реализацией API, а также web-приложение для просмотра карты.

Созданный программный продукт отвечает всем поставленным требованиям и решает все необходимые задачи.

Кроме того, разработанная система имеет ряд преимуществ в сравнении с аналогами, из чего следует то, что она имеет большой потенциал для её дальнейшего развития и модернизации.

Список литературы

1. Bhoraskar, R., Vankadhara, N., Raman, B., Kulkarni P.; Wolverine: Traffic and Road Condition estimation using Smartphone Sensors. In: Fourth International Conference on Communication Systems and Networks (COMSNETS). IEEE (January 2012).

2. Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., Balakrishnan, H.: The pothole patrol: using a mobile sensor network for road surface monitoring. In: MobiSys 2008: Proceeding of the 6th International Conference on Mobile Systems, Applications, and Services. ACM, New York (2008)

3. Gonzlez, A., O'brien, E.J., Li, Y.Y., Cashell, K.: The use of vehicle acceleration measurements to estimate road roughness. Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility (2008)

4. Mohan, P., Padmanabhan, V.N., Ramjee, R.: Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In: Proc. ACM SenSys 2008, ACM, New York (2008)

5. Thiagarajan, A., Ravindranath, L.S., LaCurts, K., Toledo, S., Eriksson, J., Madden, S, H.: VTrack: Accurate, Energy-Aware Traffic Delay Estimation Using Mobile Phones. In: ACM SenSys 2009, Berkeley, CA (November 2009)

6. Android Developers [Электронный ресурс]. Режим доступа: https://developer.android.com/index.html (дата обращения 20.3.2018).

7. Google Map API [Электронный ресурс]. Режим доступа: https://developers.google.com/maps/ (дата обращения 20.3.2018).

8. OpenStreetMap [Электронный ресурс]. Режим доступа: https://www.openstreetmap.org/ (дата обращения 20.3.2018).

9. Project OSRM [Электронный ресурс]. Режим доступа: http://project-osrm.org/ (дата обращения 24.03.2018)