

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра компьютерных технологий и систем

МЕХОВИЧ
Константин Игоревич

**АЛГОРИТМ ОПРЕДЕЛЕНИЯ ГЕОПОЗИЦИИ НА ОСНОВЕ
ВИДИМЫХ WI-FI СЕТЕЙ И СОТОВЫХ ВЫШЕК**

Дипломная работа

Научный руководитель:
доцент кафедры компьютерных
технологий и систем,
кандидат физ.-мат. наук
Козловская Инесса Станиславовна

Допущена к защите
«____» 2024 г.

Зав. кафедрой компьютерных технологий и систем
доктор педагогических наук,
кандидат физ.-мат. наук,
профессор В. В. Казаченок

Минск, 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	7
1 ГЕОЛОКАЦИЯ НА ОСНОВЕ WI-FI И СОТОВЫХ СЕТЕЙ	9
1.1 Принципы работы Wi-Fi сетей	9
1.2 Определение геопозиции с использованием Wi-Fi сетей	10
1.3 Принципы работы сотовых сетей	11
1.4 Определение геопозиции с использованием сотовых сетей	12
1.5 Влияние характеристик сотовых устройств на точность определения геопозиции	12
1.6 Влияние расположения сотовых вышек на точность определения геопозиции	13
1.7 Различия сотовых вышек и их влияние на определение геопозиции	14
1.8 Существующие технологии и алгоритмы определения геопозиции	16
2 ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ	18
2.1 Android приложение	18
2.2 Фронтенд-страницы для анализа	24
2.2.1 Страница статистики	24
2.2.2 Карта всех Wi-Fi точек	25
2.2.3 Лог всех запросов	25
2.2.4 Визуализация каждого запроса на карте	26
3 РАЗРАБОТКА АЛГОРИТМА ОПРЕДЕЛЕНИЯ ГЕОПОЗИЦИИ	28
3.1 Анализ исходных данных	28
3.2 Формулы нахождения расстояния	29
3.2.1 Нахождения расстояния между географическими координатами	29
3.2.2 Нахождения расстояния до точки доступа	30
3.3 Основные шаги алгоритма	31
3.3.1 Алгоритм записи в базу данных	31
3.3.2 Алгоритм детекции пользователя	32
3.4 Применение алгоритма	33
3.5 Особенности алгоритма	34
4 ТЕСТИРОВАНИЕ И АНАЛИЗ РЕЗУЛЬТАТОВ	36
4.1 Методика тестирования	36
4.2 Визуализация данных	36
4.3 Заключение по результатам тестирования	38

ЗАКЛЮЧЕНИЕ	39
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	40
ПРИЛОЖЕНИЕ А	41
ПРИЛОЖЕНИЕ Б	46

РЕФЕРАТ

Дипломная работа, 53 стр., 14 иллюстр., 10 источников.

Ключевые слова: ГЕОПОЗИЦИЯ, WI-FI СЕТИ, СОТОВЫЕ ВЫШКИ, ANDROID, KOTLIN, PYTHON, GPS, БАЗА ДАННЫХ.

Объект исследования: использование доступных Wi-Fi сигналов и сотовых вышек для определения геопозиции.

Предмет исследования: преобразование данных из имеющихся Wi-Fi сигналов и сотовых вышек в геопозицию с помощью известных данных о их расположении.

Цель работы: разработка эффективного алгоритма для определения геопозиции устройства, основываясь на информации от видимых Wi-Fi сетей и сотовых вышек, и создание соответствующего Android приложения.

Методы исследования: программирование на языках Kotlin и Python, наблюдения, тестирование, анализ данных, статистический анализ, расчеты, использование базы данных, GPS-трекинг.

Результат: был разработан алгоритм для определения геопозиции устройства не только на основе информации от домашних и общедоступных Wi-Fi сетей, но также с использованием данных с сотовых вышек. Эта функциональность была реализована в виде Android-приложения и серверной части на языке Python. Android-приложение сканирует Wi-Fi сети и сигналы сотовых вышек, отправляя эти данные на сервер либо в сочетании с GPS-данными, либо без них. Серверная часть, используя специализированный алгоритм и обширную базу данных, обрабатывает полученную информацию для точного определения геопозиции устройства. Это позволяет повысить точность определения местоположения, особенно в условиях городской застройки и внутри зданий, где сигнал GPS может быть нестабильным.

Области применения: метод может быть использован в областях, где важно умение точно определить геопозицию устройства, например, в мобильной связи, в системах предупреждения об экстренных ситуациях, в навигационных системах, туризме или поиске устройств.

РЭФЕРАТ

Дыпломная праца, 53 старонкі, 14 ілюстрацый, 10 крыніц.

Ключавыя слова: ГЕАПАЗІЦЫЯ, WI-FI СЕТКІ, СОТОВЫЯ ВЫШКІ, ANDROID, KOTLIN, PYTHON, GPS, БАЗА ДАДЗЕНЫХ.

Аб'ект даследвання: выкарастанне даступных Wi-Fi сігналаў і сотовых вышак для вызначэння геапазіцыі.

Прадмет даследвання: пераўтварэнне дадзеных з наяўных Wi-Fi сігналаў і сотовых вышак у геапазіцыю з дапамогай вядомых дадзеных аб іх размяшчэнні.

Мэта працы: распрацоўка эфектыўнага алгарытма для вызначэння геапазіцыі прыстасавання, засноўваючыся на інфармацыі ад бачных Wi-Fi сетак і сотовых вышак, і стварэнне адпаведнай Android прылады.

Метады даследвання: праграмаванне на мовах Kotlin і Python, на-зіранне, тэставанне, аналіз дадзеных, статыстычны аналіз, разлікі, выкарыстванне базы дадзеных, GPS-трэкінг.

Вынік: быў створан алгарытм для вызначэння геапазіцыі прыстасавання ня толькі на выснове інфармацыі ад хатніх і грамадскіх Wi-Fi сетак, але таксама з выкарыстоўваннем дадзеных з сотовых вышак. Гэта функцыянальнасць была рэалізавана ў выглядзе Android-прылады і сервернай часткі на мове Python. Android-прылада скануе Wi-Fi сеткі і сігналы сотовых вышак, адпраўляя гэтыя дадзенныя на сервер альбо ў спалученні з GPS-дадзенымі, альбо без іх. Серверная частка, выкарыстоўвае спецыяльны алгарытм і вялізную базу дадзеных, апрацоўвае атрыманую інфармацыю для дакладнага выяўлення геапазіцыі прыстасавання. Гэта дазваляе павысіць дакладнасць вызначэння месцазнаходжання, асабіста ва умовах гарадской забудовы і ўнутры будынкаў, дзе сігнал GPS можа быць нестабільным.

Вобласці прымянеñня: метад можа быць выкараставан у вобласцях, дзе важна уменне дакладна вызначаць геапазіцыю прыстасавання, напрыклад, у мабільной сувязі, у сістемах папярэджвання аб экстрадных сітуацыйах, у навігацыйных сістэмах, турызме або ў шуканні прыстасаванняў.

ABSTRACT

Diploma thesis, 53 pages, 14 illustrations, 10 sources.

Keywords: GEOGRAPHICAL LOCATION, WI-FI NETWORKS, GSM CELLS, ANDROID, KOTLIN, PYTHON, GPS, DATABASES.

Object of research: the use of available Wi-Fi signals and GSM cells to determine the geographical location.

Subject of research: the transmission of data from Wi-Fi signals and GSM cells to a geographical location using known location data.

Objective: to develop an effective algorithm for determining the geographical location of the device using the information from visible Wi-Fi networks and GSM cells and creating an appropriate application for Android.

Research methods: programming in Kotlin and Python languages, observations, testing, data analysis, statistical analysis, calculations, database usage, GPS tracking.

Result: the algorithm for determining the geographical location of the device has been developed. The algorithm works not only on the basis of home and local Wi-Fi networks but also using GSM cells. This functionality has been implemented in the Android application and in the Python backend. Android application scans the signals of Wi-Fi networks and GSM cells, sends this data to server with or without GPS data. The backend uses the algorithm and a large database to process information to accurately determine the geographical position. This helps to improve accuracy in urban environments or inside buildings where the GPS signal may be unstable.

Scope: the method can be used in scopes where it is important to accurately determine the geographical position, for example, in mobile networks, in emergency warning systems, in navigation systems, in tourism or when searching for devices.

ВВЕДЕНИЕ

В эпоху стремительного развития информационных технологий, все больше внимания уделяется вопросам улучшения качества и точности определения геолокации устройств. Такие вопросы становятся особенно актуальными в свете развития систем безопасности, мобильной коммерции, мобильных игр, служб доставки и других областей, где неотъемлемой частью является точное и быстрое определение геопозиции [1].

Однако, в ряде условий использование GPS-трекинга не предоставляет достаточной точности, или недоступно вовсе. В данной работе предлагается разработать и реализовать алгоритм, позволяющий определить геолокацию устройства на основе доступных Wi-Fi сигналов и сотовых вышек [2].

Целью данной работы является разработка подобного алгоритма, создание на его основе Android-приложения в среде Kotlin и серверной части на языке Python. Намеченная цель предполагает решение ряда задач, включая исследование существующих методов определения геопозиции, разработку алгоритма, тестирование и корректировку алгоритма.

Работа состоит из следующих этапов: обзор литературы, постановка задачи, разработка алгоритма, реализация алгоритма, тестирование, анализ результатов. В работе представлены не только теоретические расчёты, но и результаты практического тестирования разработанного алгоритма и приложения.

Данная работа может быть полезна для специалистов в области мобильных технологий и геолокации, а также для всех тех, кто интересуется вопросами определения геопозиции и стремится улучшить качество и точность такого определения. Эта работа также имеет практическую ценность, поскольку результаты исследования могут быть использованы в реальных условиях и приложениях.

Целью работы было разработать и протестировать алгоритм определения геопозиции на основе видимых Wi-Fi сетей и сотовых вышек. Для достижения данной цели были использованы средства для разработки мобильных приложений на языке Kotlin и серверной части на языке Python. Дополнительно было использовано хранилище данных на основе СУБД MySQL, а для тестирования REST интерфейса приложения была использована утилита Postman.

Входные данные алгоритма представляют собой информацию о видимых Wi-Fi сетях и сотовых вышках, собранную с использованием разработанного Android-приложения. Иногда эти данные дополняются GPS-координатами, иногда вводятся без них.

Результат работы алгоритма представляет собой географическую позицию устройства. Основная задача алгоритма состоит в том, чтобы корректно интерпретировать данные о Wi-Fi сетях и сотовых вышках и преобразовать эти данные в специфическую геолокацию.

В рамках улучшения работы и обеспечения удобного анализа результатов разработки алгоритмов определения геолокации был реализован фронтенд с использованием Flask и шаблонизатора Jinja. Этот подход позволил создать интерактивный веб-интерфейс, через который пользователи могут отслеживать статистику, проводить сравнение различных алгоритмов определения геолокации, а также отслеживать каждый запрос в реальном времени.

Важной частью фронтенда стала интеграция с картами OpenStreetMap, что позволило визуализировать данные о Wi-Fi сетях и сотовых вышках, используемые для определения геопозиции, прямо на интерактивных картах. Благодаря этому пользователи могут не только видеть результаты определения местоположения наглядно, но и анализировать плотность Wi-Fi сетей в различных районах, что является ключевым фактором для оптимизации алгоритмов.

Таким образом, разработка фронтенда на основе Flask и Jinja в сочетании с картами OpenStreetMap стала ценным дополнением к работе, значительно расширяя возможности анализа и тестирования разработанных методик определения геолокации без использования GPS.

В итоге было произведено сложное многофакторное исследование, которое требует не только знания языков программирования и работы с базами данных, но и глубокого понимания принципов функционирования Wi-Fi и сотовых сетей, а также алгоритмов и методов определения геопозиции. В результате работы получился алгоритм позволяющий определять геолокацию устройства без использования GPS-сигнала.

ГЛАВА 1. ГЕОЛОКАЦИЯ НА ОСНОВЕ WI-FI И СОТОВЫХ СЕТЕЙ

1.1 Принципы работы Wi-Fi сетей

Wi-Fi (Wireless Fidelity) – это беспроводная технология, которая использует радиоволны для передачи информации. Коммуникация через Wi-Fi обеспечивается путем передачи данных между различными устройствами через радиоволны на частотах 2,4 ГГц или 5 ГГц.

Wi-Fi сети могут быть открытыми (доступными для всех) или защищенными паролем. Каждая Wi-Fi сеть имеет уникальный идентификатор BSSID (Basic Service Set Identifier), который уникален для каждой точки доступа и используется устройствами для подключения.

Основой работы Wi-Fi является стандарт 802.11. Обмен данными происходит по средствам электромагнитного излучения в 2,4-5 ГГц диапазоне волн, с использованием протокола специального назначения. Протокол определяет такие параметры, как скорость передачи данных, дальность сигнала и многие другие.

Существует несколько версий стандарта 802.11, включая 802.11a, 802.11b, 802.11g, 802.11n и 802.11ac, каждый из которых имеет свои характеристики и может использоваться в разных условиях (рисунок 1.1).

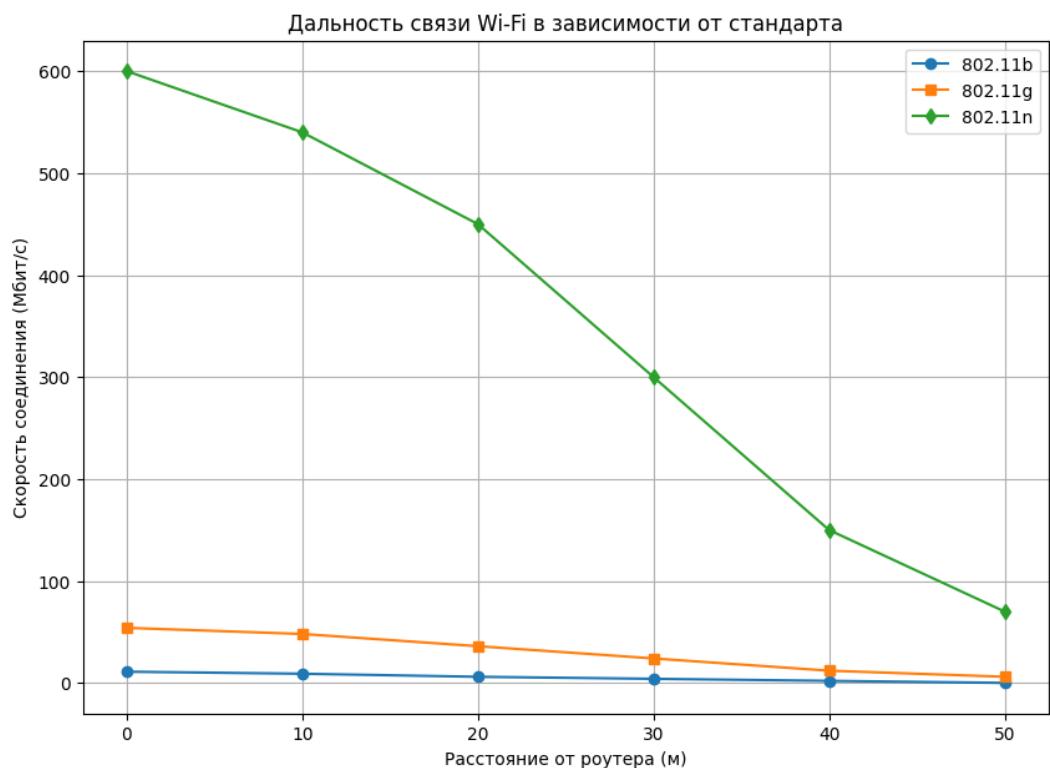


Рисунок 1.1 — Скорости и дальности связи различных стандартов Wi-Fi

Диапазон действия Wi-Fi обычно ограничен несколькими сотнями метров в открытом пространстве, хотя этот показатель может значительно варьироваться в зависимости от наличия препятствий при распространении сигнала (рисунок 1.2), мощности передающего устройства и многих других факторов.

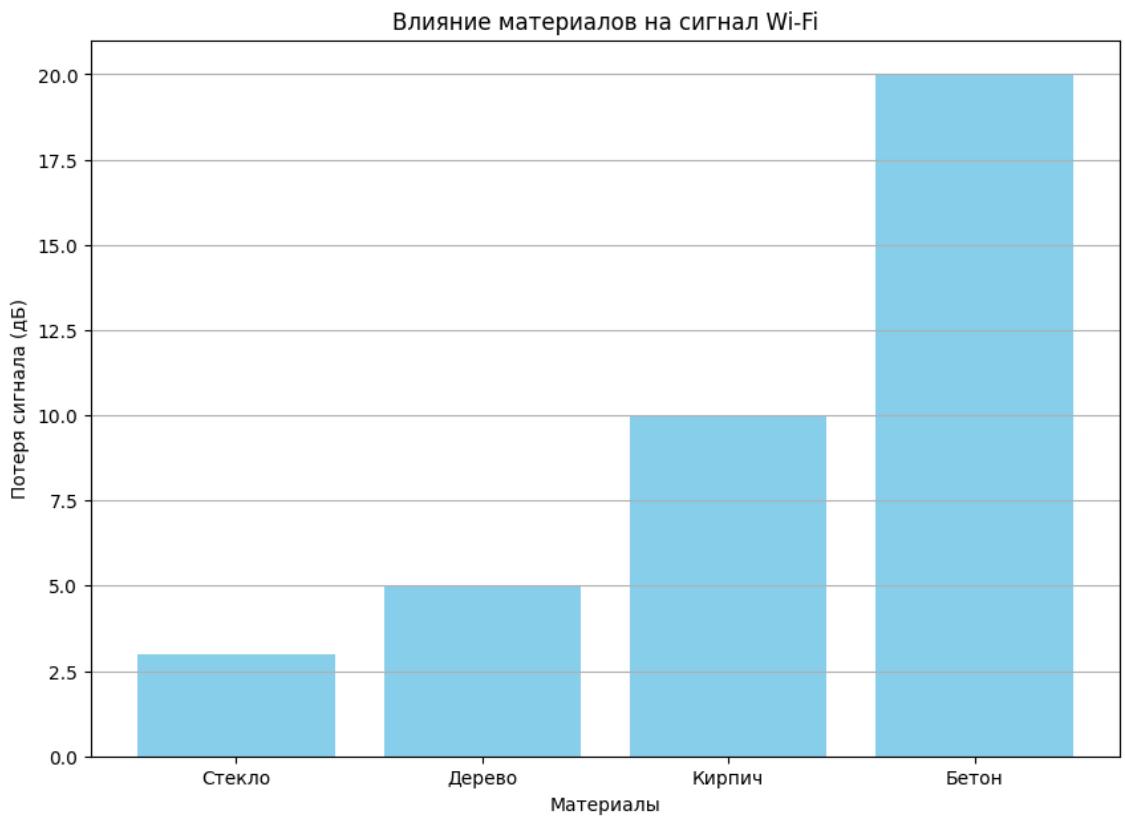


Рисунок 1.2 — Потеря сигналов Wi-Fi в зависимости от материала препятствия

Информация о видимых Wi-Fi сетях, такая как BSSID и уровень сигнала, может использоваться для определения геопозиции устройства. Существует корреляция между уровнем приема сигнала и расстоянием от точки доступа, что позволяет определить вероятное географическое положение пользовательского устройства.

1.2 Определение геопозиции с использованием Wi-Fi сетей

Определение геопозиции с использованием Wi-Fi сетей основывается на принципе, что у каждой точки доступа Wi-Fi есть уникальный идентификатор - BSSID, который всегда остается стабильным. Когда мобильное устройство находится в зоне действия Wi-Fi точки, оно может обнаружить эту сеть и ее BSSID.

С помощью специальной базы данных, которая связывает BSSID с точкой на земле, можно узнать физическое положение устройства. Это и является

основой работы определения геопозиции с использованием Wi-Fi сетей (рисунок 1.3).

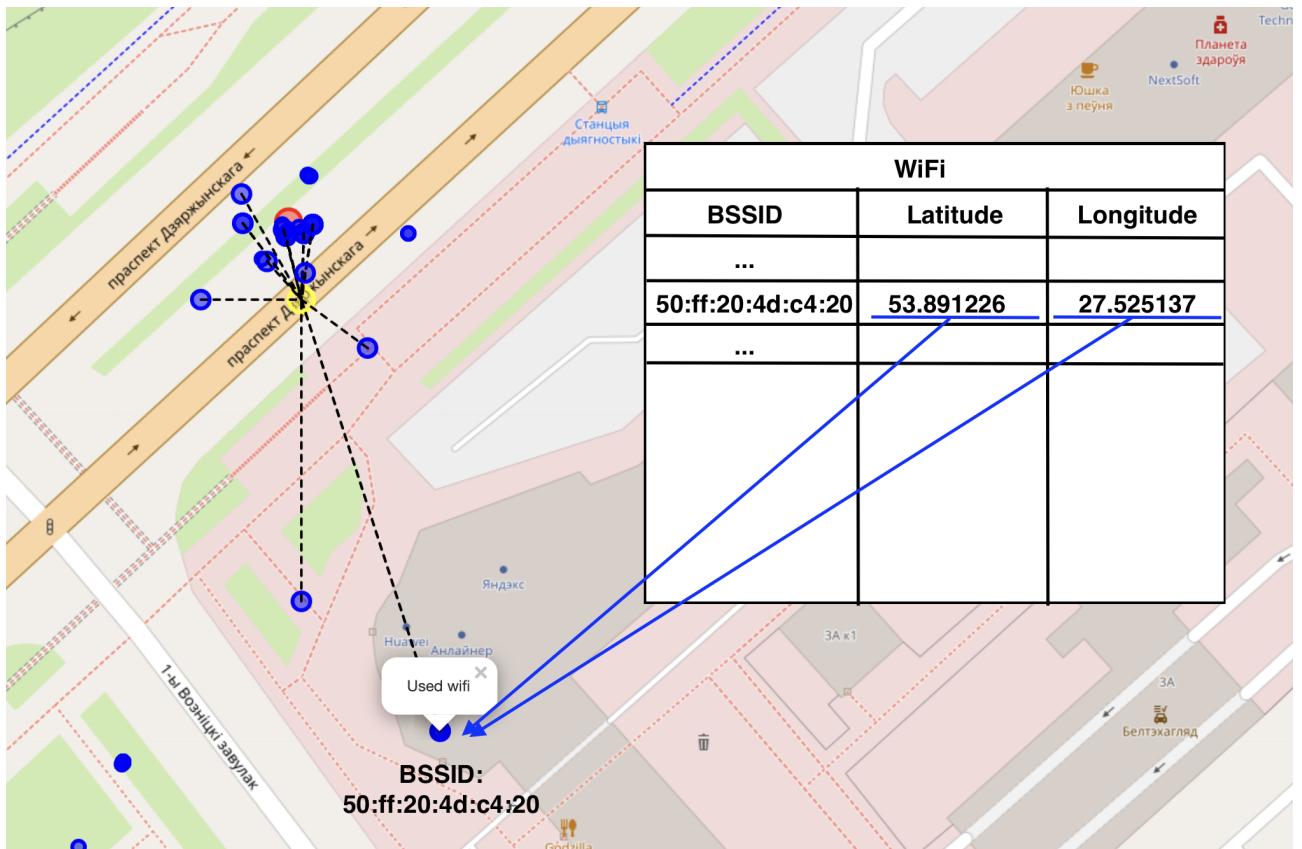


Рисунок 1.3 — Примерная схема работы получения геопозиции Wi-Fi сети

Этот метод имеет свои ограничения и не всегда обеспечивает высокую точность, особенно в условиях городской застройки, где сигнал Wi-Fi может значительно искажаться. Стоит также отметить, что не все точки доступа Wi-Fi зарегистрированы в общедоступных базах данных, что создает дополнительные сложности. Тем не менее, использование Wi-Fi в сочетании с другими методами определения местоположения, такими как GPS и сотовые вышки, может существенно повысить точность геопозиционирования. Это особенно актуально в помещениях, где сигнал GPS может быть слабым или отсутствовать вовсе.

В этой работе мы исследуем алгоритмы и методы, направленные на улучшение точности определения местоположения с помощью данных о Wi-Fi сетях.

1.3 Принципы работы сотовых сетей

Одним из видов беспроводной связи является сотовая связь. Она служит для поддержки связи между мобильными телефонами и модемами. Главными элементами беспроводной связи являются: базовые станции, управляющие центры, оборудование, которое генерирует радиосигналы.

В своем внутреннем устройстве сотовые связи используют радиоволны, которые передаются между устройствами и сотовыми вышками. Другое название сотовых вышек - соты. Отсюда и название «сотовая связь». Сота покрывает связью некоторый шестиугольник в пространстве, а шестиугольник похож на пчелиные соты. Это позволяет покрывать шестиугольниками большие территории. Каждая сота имеет уникальный идентификатор, который доступен мобильному устройству.

Определение геопозиции по сотовым сетям может быть недостаточно точным, так как сигнал может отражаться в городе от зданий, размеры сот могут быть слишком больше. Совместное использование с Wi-Fi и GPS повысит точность позиционирования.

1.4 Определение геопозиции с использованием сотовых сетей

Геопозицию определять через данные о сотовых сетях можно на основе принципа триангуляции. Получить местоположение устройство можно благодаря постоянному взаимодействию мобильных телефонов с одной или несколькими сотовыми вышками, используя задержку сигнала от вышки к устройству. Если имеются данные о трех или более вышках, то можно воспользоваться методом триангуляции. Метод вычислит примерную геопозицию в многоугольнике, ограниченном сотовыми вышками в вершинах.

Также возможно использовать метод трилатерации, который отличается связью с базовыми станциями. Этот метод не будем в дальнейшем использовать, так как количество базовых станций недостаточно для определения геопозиции.

Эти методы в частных случаях являются эффективными, но не всегда достаточно точную, особенно учитывая положение в плотных городских застройках.

Целью работы будет использование триангуляции в комбинации с данными Wi-Fi сетях для более точного определения местоположения.

1.5 Влияние характеристик сотовых устройств на точность определения геопозиции

При разработке алгоритма использующего данные сотовых сетей нужно учитывать не только расположения сотовых вышек, но и характеристики устройства, например тип сим-карты, производителя, операционную систему, модель. Например, исследования показали, что на смартфоне Xiaomi Pocophone аппаратно возможно получить информацию только от одной сотовой вышки, к которой устройство подключено в данный момент. Это огра-

ничение снижает эффективность методов триангуляции, требующих информации минимум от трех вышек для точного определения местоположения.

Такие аппаратные ограничения значительно влияют на алгоритмы геопозиционирования, разработанные в данном исследовании. Многие современные методы определения местоположения зависят от данных нескольких вышек, и их эффективность падает при использовании устройств, подобных Xiaomi Pocophone.

Эта ситуация подчеркивает важность тестирования новых методов на различных типах устройств для обеспечения их совместимости и эффективности при разных условиях использования. Характеристики мобильного устройства, такие как возможность доступа к данным от множества вышек, сильно влияют на точность геолокации.

Для повышения точности определения местоположения и расширения возможности использования разработанных алгоритмов необходимо искать способы обхода аппаратных ограничений конкретных моделей смартфонов и разрабатывать универсальные решения, адаптируемые к разнообразию существующих устройств.

1.6 Влияние расположения сотовых вышек на точность определения геопозиции

Определение геопозиции через сотовые сети традиционно базируется на принципе триангуляции, при котором местоположение мобильного устройства вычисляется через анализ сигналов, переданных между устройством и несколькими базовыми станциями. Однако, в процессе исследования было выявлено, что сотовая вышка, к которой на данный момент подключено устройство, зачастую может располагаться на значительном удалении от реальной позиции пользователя, особенно это заметно при сравнении с данными, получаемыми от исходных точек доступа Wi-Fi.

Эта диспропорция особенно очевидна в условиях, когда мобильное устройство использует данные с одной вышки, что приводит к значительным погрешностям из-за ограничений в точности определения расстояния по одному сигналу (рисунок 1.4). Кроме того, целостность метода триангуляции оптимально реализуется в зонах с высокой плотностью базовых станций, как в городских агломерациях. В то время как в сельских и удаленных районах, где сотовые вышки разбросаны на больших расстояниях друг от друга, точность значительно снижается.

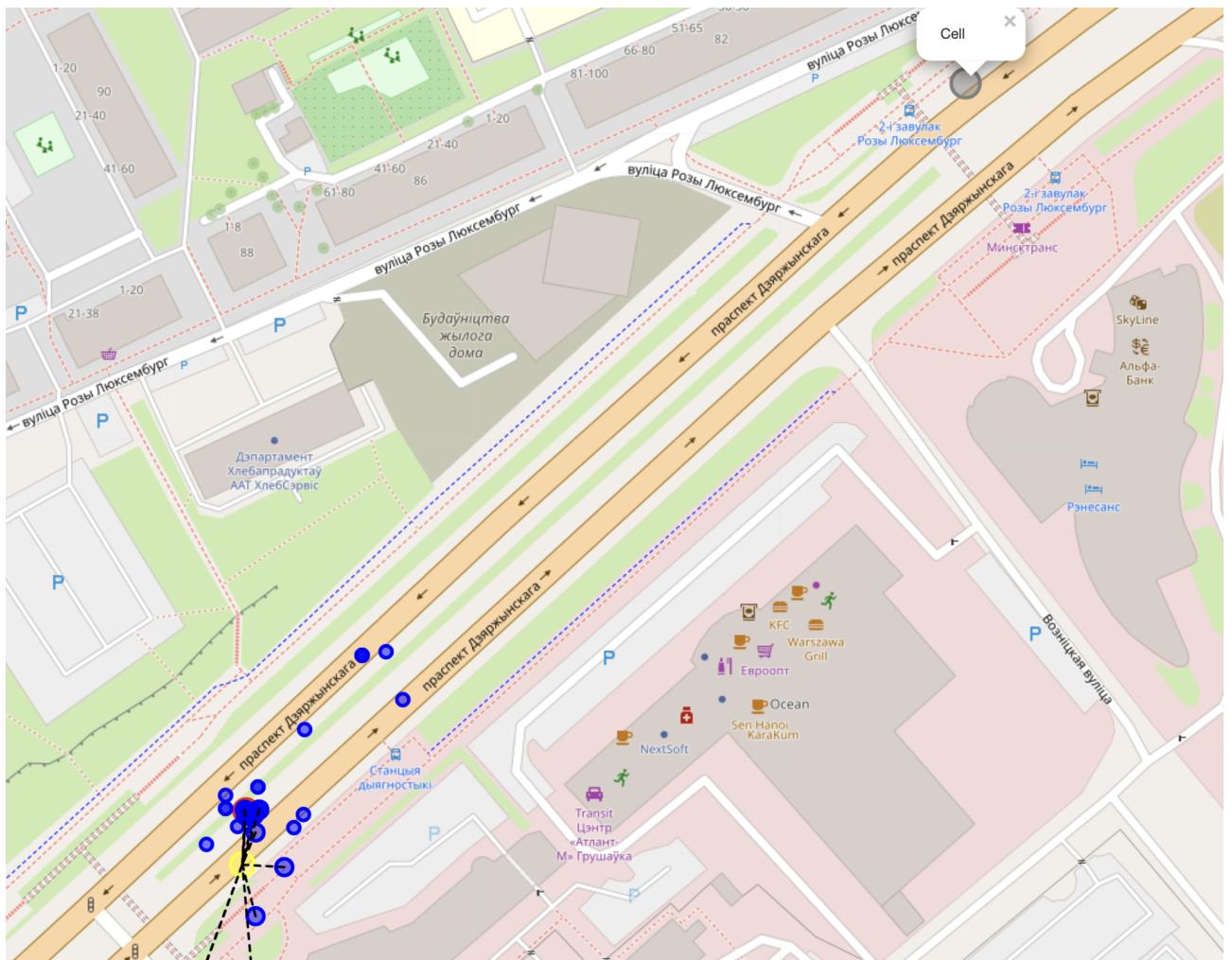


Рисунок 1.4 — Удаленность сотовой вышки в городе в сравнении с Wi-Fi

Чтобы улучшить точность определения местоположения, в нашем исследовании мы рассматриваем применение комбинированного подхода, интегрируя данные из сотовых сетей и Wi-Fi. Такой подход позволяет нивелировать недостатки каждой из систем по отдельности. Например, информация от Wi-Fi точек доступа обычно доступна в частных и коммерческих зданиях и может обеспечить более высокую точность в определенных сценариях, несмотря на ограниченный радиус действия.

Ограничения существующих методов подчеркивают значимость разработки нового алгоритма, гармонично комбинирующего данные от Wi-Fi и сотовых сетей для повышения точности и надежности определения геопозиции в различных условиях, включая сложные городские и сельские ландшафты.

1.7 Различия сотовых вышек и их влияние на определение геопозиции

Сотовые сети состоят из множества базовых станций, или сотовых вышек, которые обеспечивают охват различными стандартами связи, включая

GSM (Global System for Mobile Communications) и LTE (Long-Term Evolution). Каждый из этих стандартов имеет свои особенности, которые влияют на точность и методы определения геопозиции мобильных устройств.

GSM является более старым стандартом, который поддерживает основные функции телефонной связи и передачу данных низкой скорости. Системы GSM используют время-разделительную мультиплексацию (TDMA) для разделения сигналов. В GSM доступны такие данные, как идентификаторы сотовых вышек (Cell ID), информация о секторах и LAC (Location Area Code), что позволяет приблизительно определять местоположение устройства.

LTE, с другой стороны, представляет собой стандарт современной высокоскоростной передачи данных, который использует технологии, значительно увеличивающие пропускную способность и скорость. LTE предоставляет больше параметров, таких как идентификатор ячейки (eNB-ID) и индикатор качества сигнала, например RSSI (Received Signal Strength Indicator) и RSRP (Reference Signal Received Power). Эти параметры могут использоваться для более точного определения геопозиции устройства за счёт анализа уровня сигнала и его задержки.

Один из ключевых параметров, используемый в алгоритмах определения геопозиции, — это *уровень сигнала* или *level*. Наиболее распространенные значения — это RSSI для GSM и RSRP для LTE. Уровень сигнала отражает качество связи между устройством и сотовой вышкой: чем выше значение, тем лучше качество связи и, соответственно, тем точнее можно определить расстояние до вышки.

Для улучшения качества и точности определения геопозиции была использована внешняя база данных OpenCellID (рисунок 1.5), которая является самой крупной общедоступной базой данных геолокаций сотовых вышек по всему миру. Использование OpenCellID позволяет получить географические координаты сотовых вышек на основе их уникальных идентификаторов, что значительно улучшает точность определения местоположения пользователей. Благодаря использованию этой базы, разработанный алгоритм может эффективно компенсировать отсутствие данных от сотовых вышек и увеличить охват и точность геопозиционирования.

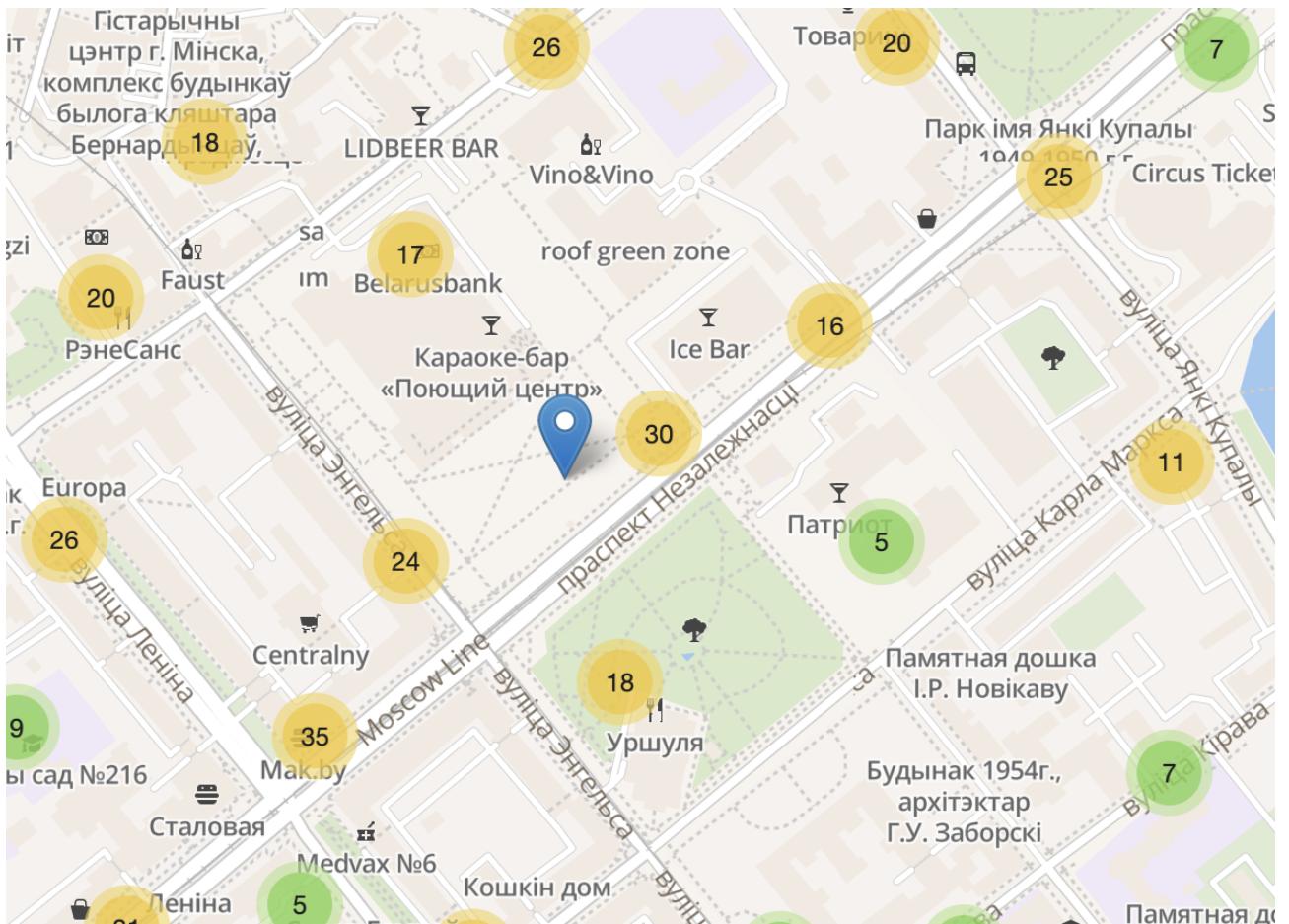


Рисунок 1.5 — Количество сотовых вышек в центре Минска согласно OpenCellID

Различия между стандартами связи GSM и LTE, а также специфика доступных данных от сотовых вышек, имеют значительное влияние на методы определения геопозиции. Включение данных о уровне сигнала и использование внешних баз, таких как OpenCellID, позволяют значительно повысить точность геолокации, предоставляя дополнительные данные для коррекции и уточнения расчетных местоположений устройств.

1.8 Существующие технологии и алгоритмы определения геопозиции

Огромное количество приложений и сервисов, таких как навигация, реклама, социальные сети и безопасность, требуют точного определения местоположения устройства. Существуют различные технологии и алгоритмы, которые используются для удовлетворения этой потребности.

GPS: система глобального позиционирования (GPS) - это самый распространенный метод определения местоположения [3].

Однако, в зданиях, районах с высокими зданиями или в других условиях, которые препятствуют прямому видению GPS спутников, этот метод может

быть ненадежным [1].

Wi-Fi: определение местоположения с использованием Wi-Fi обычно включает использование баз данных Wi-Fi точек доступа для определения местоположения. База данных обычно содержит информацию о BSSID и местоположении каждой точки доступа.

Cell-ID: этот метод использует идентификаторы сотовых вышек для определения местоположения устройства.

Гибридные системы: гибридные системы пытаются объединить различные источники местоположения для улучшения точности определения местоположения.

В данной работе мы рассматриваем алгоритм, который использует как данные Wi-Fi, так и сотовые сети для определения геопозиции, и строим архитектуру системы, которая может использовать различные методы и источники данных для достижения большей точности и надежности.

ГЛАВА 2. ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ

2.1 Android приложение

Целью создания Android-приложения на языке Kotlin было обеспечение удобства использования и простоты взаимодействия с продуктом. Приложение использует OpenStreetMap для визуализации геолокации пользователей и точек Wi-Fi. Графический интерфейс разработан с применением интуитивно понятных и простых для понимания принципов.

Центральным элементом приложения является карта OpenStreetMap, занимающая большую часть экрана. Это позволяет пользователю наглядно представить свое местоположение и ситуацию вокруг себя.

В нижней части экрана расположена кнопка «Send Wi-Fi list» или «Detect», функциональность которой меняется в зависимости от положения переключателя ниже. Переключатель позволяет выбирать между двумя режимами работы:

Scanning - при нажатии на кнопку «Send Wi-Fi list» приложение отправляет на сервер геолокацию пользователя по GPS и список доступных Wi-Fi сетей. Местоположение пользователя, определенное по GPS отображается на карте красным маркером (рисунок 2.1).

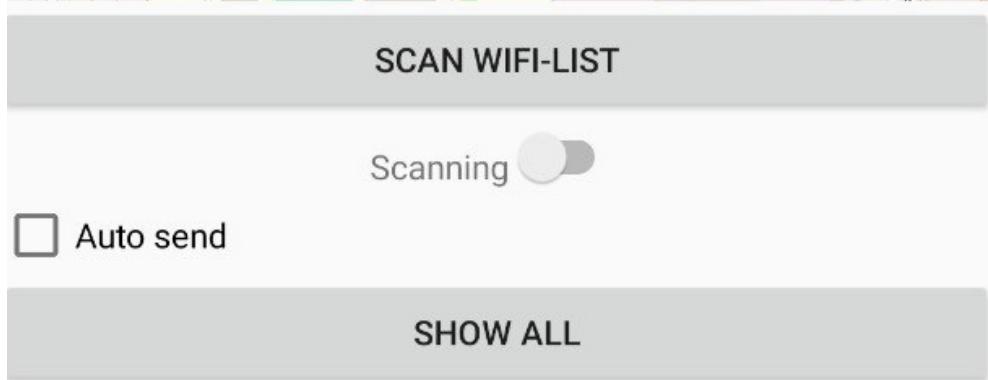
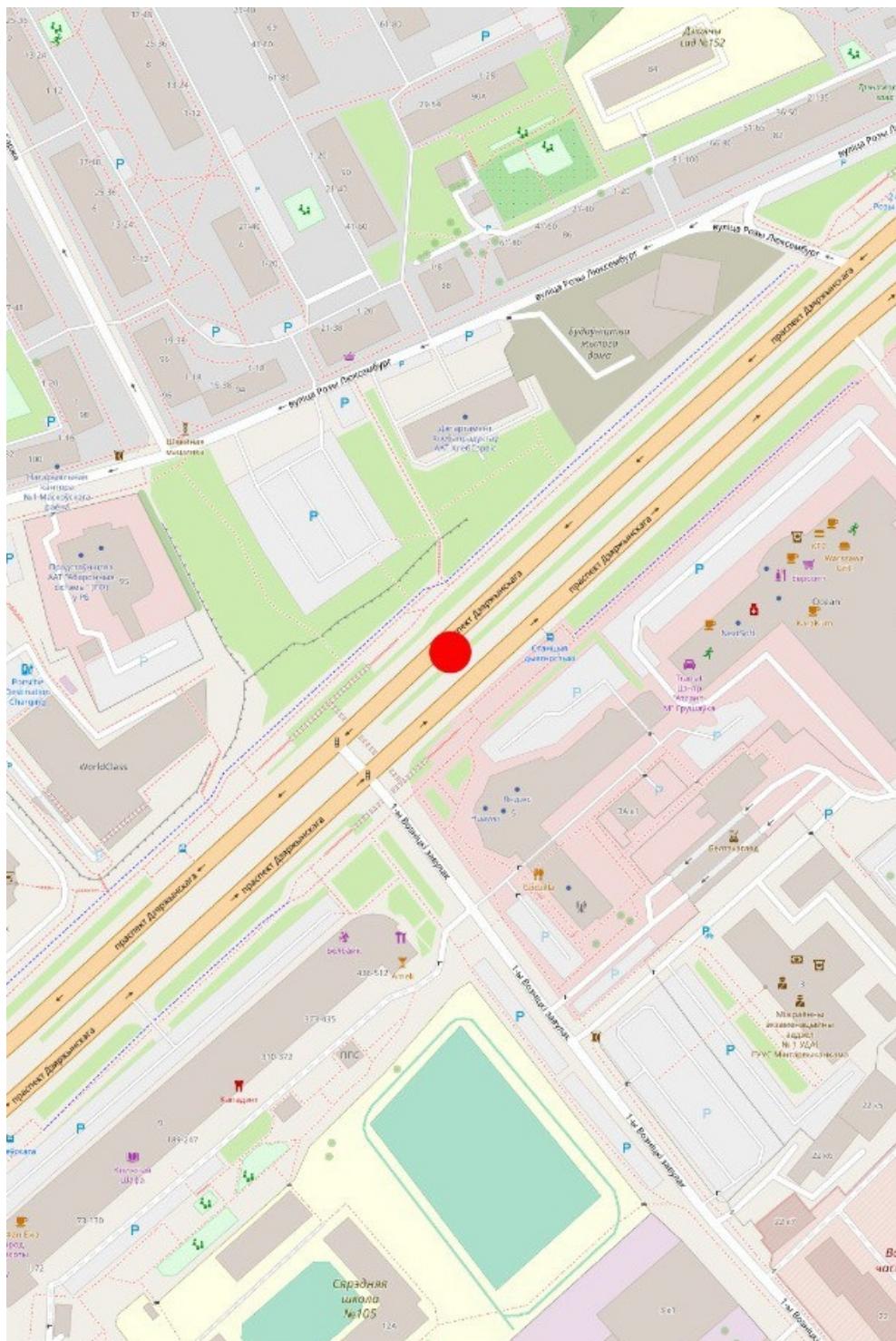
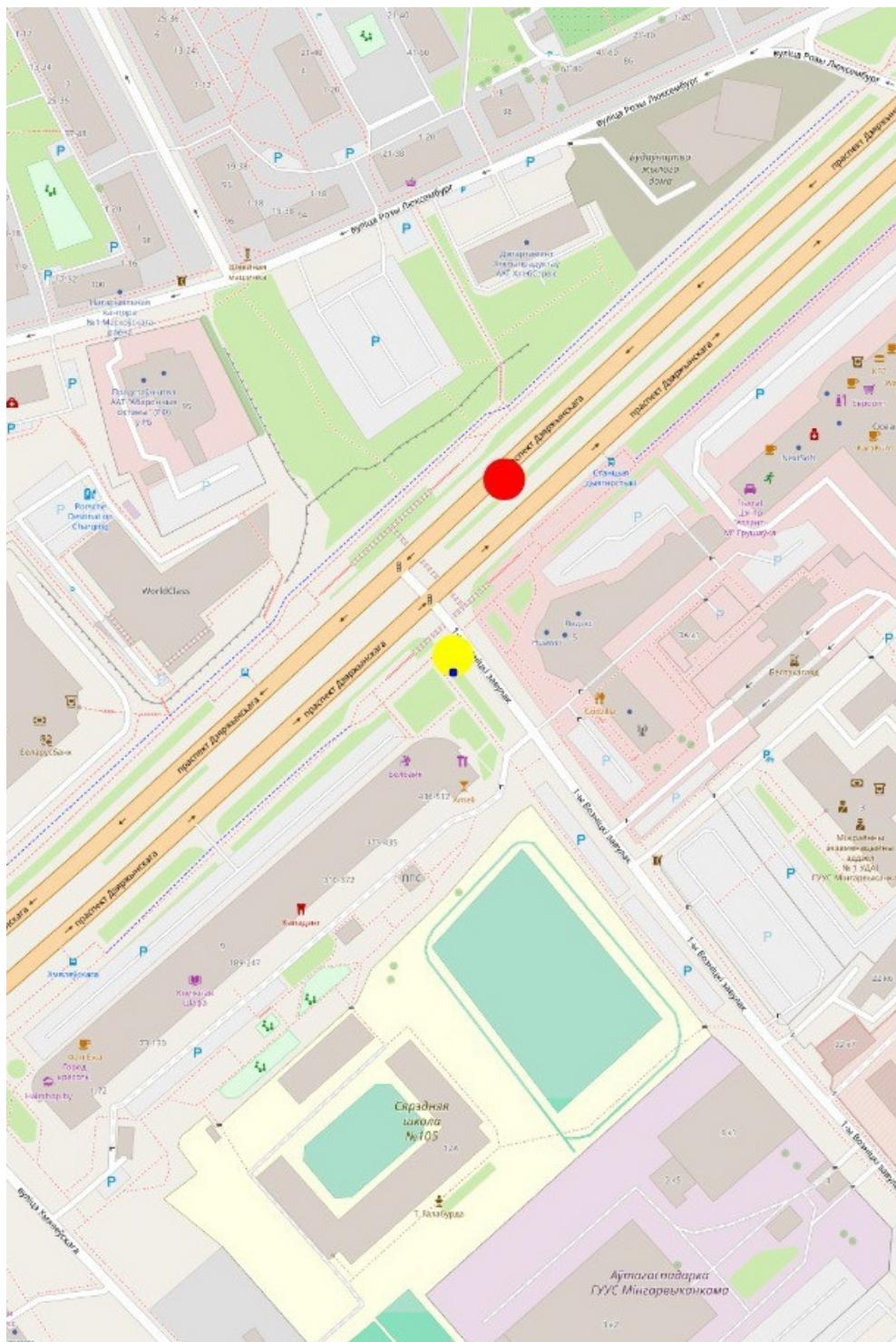


Рисунок 2.1 — Интерфейс пользователя в режиме «Scanning»

Detecting - в этом режиме приложение по нажатию на кнопку отправляет на сервер только список доступных Wi-Fi сетей, и сервер возвращает предполагаемые координаты пользователя и координаты Wi-Fi сетей, на основании которых был дать ответ. Данные Wi-Fi сети отображаются на карте синими маркерами, предполагаемое местонахождение пользователя отображается желтым цветом (рисунок 2.2).



DETECT

Detecting

Auto select Location based on 32 wifi. Error is 74.16 meters

[SHOW ALL](#)

Рисунок 2.2 – Интерфейс пользователя в режиме «Detecting»

Снизу расположена опция автоматической отправки данных - «Auto send». При ее активации, приложение автоматически нажимает кнопку «Send Wi-Fi list» или «Detect» каждые две секунды.

Также, в нижней части экрана есть кнопка «Show All», которая при нажатии обращается к серверу за запрашиванием из базы данных информации о всех Wi-Fi сетях. Они визуализируются на карте зелеными маркерами (рисунок 2.3).



Рисунок 2.3 — Интерфейс при нажатии кнопки «Show All»

Интерфейс приложения предельно функционален и интуитивно понятен, что позволяет пользователям легко освоиться и эффективно использовать его для геолокации.

2.2 Фронтенд-страницы для анализа

Для работы приложения был также разработан малейший фронтенд для анализа статистики работы алгоритма. Фронтенд использует шаблоны Jinja в своей реализации. Были разработаны следующие инструменты:

- Страница статистики: содержит общую статистику ошибок по алгоритмам в разрезе основных перцентиляй,
- Карта всех Wi-Fi точек: содержит карту OpenStreetMap, на которую нанесены все известные приложению Wi-Fi точки,
- Лог всех запросов: содержит весь лог запросов, записанный приложением. Имеются данные о времени запроса, использованных данных про Wi-Fi и GSM, а также результаты работы,
- Визуализация каждого запроса на карте: позволяет просмотреть на карте каждый запрос: Wi-Fi точки, которые не повлияли и повлияли результат, GSM вышки, GPS расположение и ошибку алгоритма.

Такой фронтенд повышает удобство использования системы в целом. Он обеспечивает инструментами для анализа данных и дополняет интерактивным доступом к ним. В дальнейшем это поможет в сравнительном анализе между алгоритмами.

2.2.1 Страница статистики

Статистика расположена на странице по адресу /stats. Страница содержит анализ работы системы полностью. Разработчик алгоритма или пользователи могут проанализировать распределение ошибок в случае разных алгоритмов по перцентилям. Это значительно помогает сравнить алгоритмы в общем случае.

Algo	25%	50%	75%	90%	95%	99%
v0	14.48	16.06	20.06	23.43	65.16	97.73
v0_True_Tr	13.93	18.33	18.33	22.70	24.12	24.97
v0_True_Fa	23.30	32.49	36.93	36.93	36.93	36.93
v0_False_T	11.58	11.58	15.48	18.38	18.72	20.24
v0_False_F	23.41	26.06	27.20	27.20	27.54	29.10

Рисунок 2.4 — Страница статистики

2.2.2 Карта всех Wi-Fi точек

Карта с отображением всех Wi-Fi точек расположена на странице по адресу [/карты](#). Там можно наблюдать все точки, собранные Android-приложением. Эта страница также очень полезна, так как позволяет визуально проанализировать плотность собранных Wi-Fi и установить районы, где нужно больше данных для более точного определения геопозиции.



Рисунок 2.5 — Карта всех Wi-Fi точек

2.2.3 Лог всех запросов

Лог всех запросов расположен по адресу [/](#). Он содержит все запросы отправленные пользователями через приложение. Детализированная история всех запросов позволяет лучше осознать как работает система на практике и почему получился тот или иной результат.

Global Log Entries

Index	Input GPS	Input WiFis	Input Cells	Output	Error Code	Queries	Time	Algo	
26	{'lat': 53.9099177, 'lon': 27.76580606}	[{"BSSID": "5c:6a:80:58:7c:b0", "SSID": "Rublevskiy_18", "capabilities": "[WPA2-PSK-CCMP][RSN-PSK-CCMP][ESS][WPS]", "frequency": 2452, "lat": 0.0, "level": -49, "lon": 0.0}, {"BSSID": "70:77:81:76:ec:95", "SSID": "HP-Print-95-LaserJet Pro MFP", "capabilities": "[ESS]", "frequency": 2437, "lat": 0.0, "level": -68, "lon": 0.0}, {"BSSID": "30:0c:23:a3:be:a2", "SSID": "Rublevskiy_18", "capabilities": "[WPA2-PSK-CCMP][RSN-PSK-CCMP][ESS][WPS]", "frequency": 2452, "lat": 0.0, "level": -81, "lon": 0.0}, {"BSSID": "cc:10:fe:c1:e5:ea", "SSID": "Kreksen", "capabilities": "[WPA-PSK-TKIP+CCMP][WPA2-PSK-TKIP+CCMP][RSN-PSK-TKIP+CCMP][ESS][WPS]", "frequency": 2412, "lat": 0.0, "level": -85, "lon": 0.0}, {"BSSID": "5c:6a:80:58:7c:b0", "SSID": "Sk-bag", "capabilities": "[WPA2-PSK-FT/PSK-CCMP][RSN-PSK-FT/PSK-CCMP][ESS][WPS]", "frequency": 2427, "lat": 0.0, "level": -86, "lon": 0.0}, {"BSSID": "24:7e:51:89:17:38", "SSID": "ZTE_2.4G_S6p7b", "capabilities": "[WPA-PSK-TKIP+CCMP][WPA2-PSK-TKIP+CCMP][RSN-PSK-TKIP+CCMP][ESS][WPS]", "frequency": 2427, "lat": 0.0, "level": -89, "lon": 0.0}], [{"lat": 53.9099177, "lon": 27.765864, "count": 1, "wifis": [{"lat": 53.9099177, "lon": 27.765864, "level": -49.0, "frequency": 2452.0}], "cells": []}], 200, [{"wifis": [{"bssid": "5c:6a:80:58:7c:b0", "lat": 53.909792, "lon": 27.765864, "level": -92, "frequency": 2457.0}], "cells": []}], 1710090981, v0, View Details							
27	{'lat': 53.90986347, 'lon': 27.76585272}	[{"BSSID": "5c:6a:80:58:7c:b0", "SSID": "Rublevskiy_18", "capabilities": "[WPA2-PSK-CCMP][RSN-PSK-CCMP][ESS][WPS]", "frequency": 2452, "lat": 0.0, "level": -47, "lon": 0.0}, {"BSSID": "30:0c:23:a3:be:a2", "SSID": "Rublevskiy_18", "capabilities": "[WPA2-PSK-CCMP][RSN-PSK-CCMP][ESS][WPS]", "frequency": 2462, "lat": 0.0, "level": -67, "lon": 0.0}, {"BSSID": "cc:10:fe:c1:e5:ea", "SSID": "Kreksen", "capabilities": "[WPA-PSK-TKIP+CCMP][WPA2-PSK-TKIP+CCMP][RSN-PSK-TKIP+CCMP][ESS][WPS]", "frequency": 2412, "lat": 0.0, "level": -79, "lon": 0.0}, {"BSSID": "5c:6a:80:58:7c:b0", "SSID": "Sk-bag", "capabilities": "[WPA-PSK-TKIP+CCMP][RSN-PSK-TKIP+CCMP][ESS][WPS]", "frequency": 2427, "lat": 0.0, "level": -86, "lon": 0.0}, {"BSSID": "24:7e:51:89:17:38", "SSID": "ZTE_2.4G_S6p7b", "capabilities": "[WPA-PSK-TKIP+CCMP][RSN-PSK-TKIP+CCMP][ESS][WPS]", "frequency": 2427, "lat": 0.0, "level": -89, "lon": 0.0}], [{"lat": 53.910324, "lon": 27.766889, "count": 2, "wifis": [{"lat": 53.910324, "lon": 27.766889, "level": -67.0, "frequency": 2462.0}, {"lat": 53.910428, "lon": 27.767008, "level": -77.0, "frequency": 2457.0}], "cells": []}], 200, [{"wifis": [{"bssid": "5c:6a:80:58:7c:b0", "lat": 53.909792, "lon": 27.765864, "level": -92, "frequency": 2457.0}], "cells": []}], 1710091394, v0, View Details							
		[{"BSSID": "00:4e:35:2b:6d:31", "SSID": "Guests", "capabilities": "[WPA2-PSK-CCMP][RSN-PSK-CCMP][ESS]", "frequency": 5240, "lat": 0.0, "level": -37, "lon": 0.0}, {"BSSID": "00:4e:35:2b:6d:21", "SSID": "Guests", "capabilities": "[WPA2-PSK-CCMP][RSN-PSK-CCMP][ESS]", "frequency": 2462, "lat": 0.0, "level": -38, "lon": 0.0}, {"BSSID": "00:4e:35:2b:6d:23", "SSID": "MobCert", "capabilities": "[WPA2-PSK-CCMP][RSN-PSK-CCMP][ESS]", "frequency": 2462, "lat": 0.0, "level": -38, "lon": 0.0}, {"BSSID": "00:4e:35:2b:6d:22", "SSID": "PDAS", "capabilities": "[WPA2-EAP-CCMP][RSN-EAP-CCMP][ESS]", "frequency": 2462, "lat": 0.0, "level": -38, "lon": 0.0}, {"BSSID": "00:4e:35:2b:6d:21", "SSID": "PDAS", "capabilities": "[WPA2-EAP-CCMP][RSN-EAP-CCMP][ESS]", "frequency": 2462, "lat": 0.0, "level": -38, "lon": 0.0}, {"BSSID": "00:4e:35:2b:6d:30", "SSID": "Yandex", "capabilities": "[WPA2-EAP-CCMP][RSN-EAP-CCMP][ESS]", "frequency": 5240, "lat": 0.0, "level": -38, "lon": 0.0}, {"BSSID": "00:4e:35:2b:6d:32", "SSID": "PDAS", "capabilities": "[WPA2-EAP-CCMP][RSN-EAP-CCMP][ESS]", "frequency": 5240, "lat": 0.0, "level": -38, "lon": 0.0}, {"BSSID": "00:4e:35:2b:6d:33", "SSID": "MobCert", "capabilities": "[WPA2-PSK-CCMP][RSN-PSK-CCMP][ESS]", "frequency": 5240, "lat": 0.0, "level": -38, "lon": 0.0}, {"BSSID": "00:4e:35:2b:7c:21", "SSID": "Guests", "capabilities": "[WPA2-PSK-CCMP][RSN-PSK-CCMP][ESS]", "frequency": 2412, "lat": 0.0, "level": -63, "lon": 0.0}, {"BSSID": "00:4e:35:2b:7c:c2", "SSID": "PDAS", "capabilities": "[WPA2-EAP-CCMP][RSN-EAP-CCMP][ESS]", "frequency": 2412, "lat": 0.0, "level": -63, "lon": 0.0}], [{"lat": 53.891223, "lon": 27.525286, "count": 1, "wifis": [{"lat": 53.891223, "lon": 27.525286, "level": -81, "frequency": 5240.0}], "cells": []}], 200, [{"wifis": [{"bssid": "00:4e:35:2b:6d:31", "lat": 53.891223, "lon": 27.525286, "level": -92, "frequency": 2457.0}], "cells": []}], 1710091394, v0, View Details							

Рисунок 2.6 — Лог всех запросов

2.2.4 Визуализация каждого запроса на карте

Визуализация каждого запроса на карте расположена по адресу /item/<id>, где id - номер запроса в логе. Страница показывает всех Wi-Fi точек, которые не повлияли и повлияли результат, GSM вышки, GPS расположение и ошибку алгоритма. Это позволяет внимательно проанализировать работу алгоритма, увидеть в нем проблемы, чтобы в дальнейшем исправить их

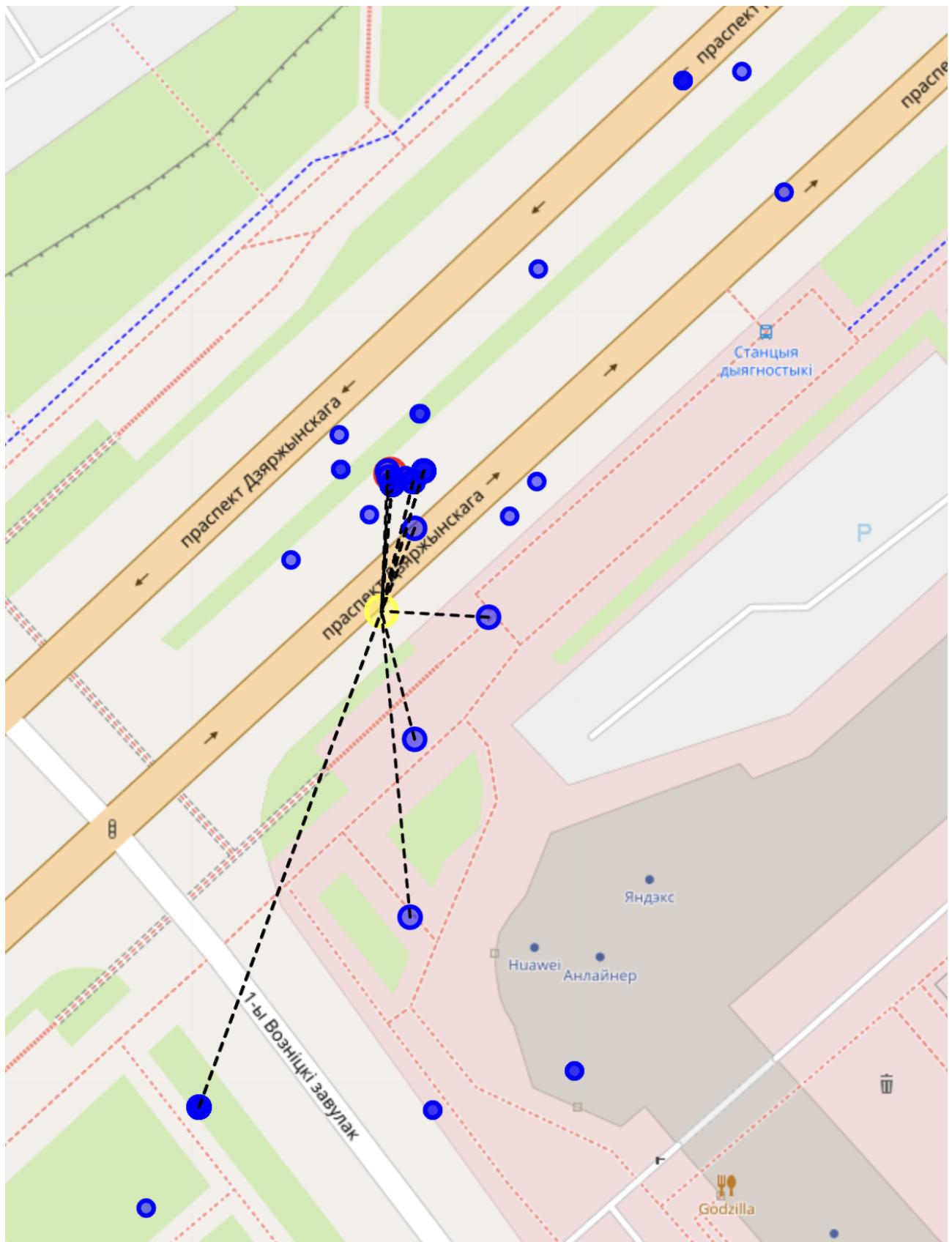


Рисунок 2.7 — Визуализация каждого запроса на карте

ГЛАВА 3. РАЗРАБОТКА АЛГОРИТМА ОПРЕДЕЛЕНИЯ ГЕОПОЗИЦИИ

3.1 Анализ исходных данных

В качестве исходного материала для разработки алгоритма определения геопозиции по Wi-Fi применялся большой массив данных, который включает в себя информацию о десятках тысяч Wi-Fi точек, собранных в городе Минске. Эти данные предоставляют важную основу для нашего исследования и потенциала для создания точного алгоритма определения геолокации.

Один из ключевых аспектов исходных данных — это качество связи с Wi-Fi точками, которое также было записано в процессе сбора данных. Эта информация может быть полезна для уточнения геолокации устройства и повышения точности алгоритма.

Однако были обнаружены некоторые сложности. Так, использование GPS при сборе данных иногда приводило к неточностям в определении геолокации Wi-Fi точек. Эти неточности при определении местоположения могут оказать влияние на разработку и точность финального алгоритма.

Отдельно следует отметить, что качество всех собранных записей было высоким и не содержало неполных или неточных данных. Таким образом, каждая запись представляет собой качественный исходный материал для обработки алгоритмом.

В целом, анализ исходных данных показывает, что собранный материал представляет собой мощную основу для разработки и тестирования алгоритма определения геопозиции на основе данных Wi-Fi точек.

При разработке нашего алгоритма определения геопозиции значительное внимание было уделено анализу данных, собранных с сотовых вышек. В частности, использование смартфона Xiaomi PocoPhone в процессе сбора данных внесло определенные особенности в структуру исходных материалов. На основе характеристик этой модели мобильного телефона мы столкнулись с ограничением: устройство предоставляет данные идентификаторов только от одной сотовой вышки, к которой оно подключено в данный момент времени. Возможность получения информации от дополнительных вышек, как правило, существенно улучшает точность методов триангуляции и трилатерации при определении местоположения, однако в данном случае мы были ограничены одним источником.

Это ограничение оказало прямое влияние на разработку и тестирование нашего алгоритма. Способность устройства обнаруживать исключительно текущую подключенную сотовую вышку представляет собой значительное ограничение для точности определения местоположения, особенно в сценариях, где важно использование данных от множества вышек для повышения

точности геолокации. Это потребовало применения дополнительных методов обработки данных и адаптации алгоритма под ограниченные входные данные, что стало вызовом на пути к разработке эффективного решения.

Учитывая эти условия, были проведены дополнительные исследования и адаптационные меры для минимизации влияния данного ограничения на конечную точность алгоритма. В частности, исследованы альтернативные подходы к определению местоположения, которые могли бы компенсировать отсутствие информации от нескольких сотовых вышек, включая усовершенствованные методы оценки расстояния до единственной доступной вышки и улучшенную интеграцию данных Wi-Fi.

Таким образом, характеристики использованного в исследовании смартфона Xiaomi Pocophone и связанные с ним ограничения добавили дополнительный слой сложности в процесс разработки алгоритма, но также способствовали разработке более гибких и адаптивных методов определения геопозиции.

3.2 Формулы нахождения расстояния

При разработке алгоритма определения геопозиции на основе Wi-Fi необходимо учесть формулы, необходимые для определения расстояния между точками по географическим координатам и расстояния до роутера в зависимости от уровня сигнала (level) и частоты (frequency).

3.2.1 Нахождения расстояния между географическими координатами

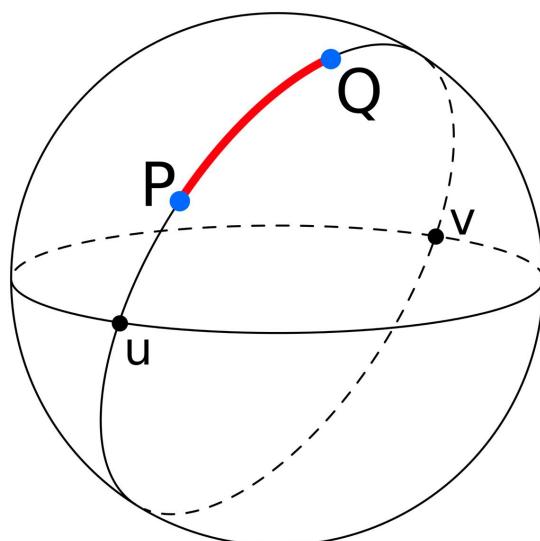


Рисунок 3.1 — Формула гаверсинуса

Для определения расстояния между точками по географическим координатам можно использовать гаверсинусную формулу (3.1):

$$d = R \cdot \arccos (\sin(\phi_1) \cdot \sin(\phi_2) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \cos(\Delta\lambda)), \quad (3.1)$$

где:

- d - расстояние между точками в метрах,
- R - радиус Земли (примерно 6371 километр),
- (ϕ_1, ϕ_2) - широта первой и второй точки соответственно (в радианах),
- $(\Delta\lambda)$ - разница долгот между точками (в радианах).

Благодаря вычислению расстояний между точками мы в последующем сможем фильтровать точки, чтобы сильно отдаленная точка доступа не влияла на результат алгоритма. Без наличия такой фильтрации выбросы могут существенно влиять на ответ алгоритма детекции [4].

3.2.2 Нахождения расстояния до точки доступа

Для определения расстояния до роутера в зависимости от уровня сигнала (level) и частоты (frequency), можно использовать модель FSPL (Free Space Path Loss) [5].

Как выражается FSPL можно наблюдать в формуле (3.2):

$$FSPL = 20 \cdot \log_{10}(d) + 20 \cdot \log_{10}(f) + K, \quad (3.2)$$

где:

- $FSPL$ - потеря сигнала в децибелах,
- d - расстояние до роутера в метрах,
- f - частота сигнала в герцах,
- K - это константа, зависящая от единиц измерения и других констант, связанных с электромагнитными волнами, обычно принимаемая как $20 \log_{10} \left(\frac{4\pi}{c} \right)$, где c - скорость света.

FSPL (Free Space Path Loss) представляет собой потерю сигнала в свободном пространстве, которая происходит из-за распространения электромагнитных волн. Эта потеря зависит от расстояния между передатчиком и приемником сигнала, частоты сигнала, преграждающих объектов и других факторов [5].

Однако, в реальных условиях существуют различные помехи и препятствия на пути сигнала, такие как стены, здания, ландшафт и т.д., которые могут значительно изменять потерю сигнала. Точное измерение значений FSPL в таких условиях может быть сложно и затруднено.

Если предположить, что нет потерь на свободном пути, тогда уровень сигнала *level* передатчика и приемника будет идентичным. В реальных условиях это встречается редко, но для теоретического расчета мы можем использовать уровень сигнала, как отправную точку для расчета *FSPL*. Если *level* задан в децибелах и представляет собой уровень сигнала у приемника, то мы можем выразить *FSPL* как указано в формулах (3.3 - 3.4):

$$\text{FSPL (dB)} = \text{Transmit Power (dBm)} - \text{Receive Level (dBm)} \quad (3.3)$$

или просто

$$\text{FSPL (dB)} = \text{Level}, \quad (3.4)$$

если у нас есть только значение уровня сигнала на приемнике.

Из формул (3.3 - 3.4) мы можем выразить формулу нахождения расстояния (3.5 - 3.6):

$$d = 10^{\frac{-\text{Level (dBm)} - 20 \log_{10}(f) - 20 \log_{10}(\frac{4\pi}{c})}{20}}. \quad (3.5)$$

Скорость света *c* в вакууме равна 3×10^8 метров в секунду. Подставляем значение *c* в формулу:

$$d = 10^{\frac{27.55 - \text{level} - 20 \log_{10}(f)}{20}}. \quad (3.6)$$

Таким образом, данная формула позволяет оценить расстояние до роутера (в м) на основе измеренного уровня сигнала (в дБм) и его частоты (в МГц), предполагая, что другие потери в системе или окружении незначительны или уже учтены в значении *level*.

3.3 Основные шаги алгоритма

3.3.1 Алгоритм записи в базу данных

Работу предложенного алгоритма можно разбить на следующие основные шаги:

- Получение данных от пользователя: при каждом нажатии кнопки «Send Wi-Fi list» или активации «Auto send», android приложение отправляет данные о местонахождении пользователя и списке доступных Wi-Fi сетей на сервер.

- Обработка полученных данных: полученные данные проходят через встроенный фильтр по точности GPS. Это значит, что данные, у которых точность GPS больше 70 метров, игнорируются. Это делает формируемую базу данных более точной и надежной.
- Запись данных в базу данных: если полученные данные о Wi-Fi сети отсутствуют в базе данных, то они записываются как новый вайфай с индивидуальным идентификатором и параметрами местоположения, а также уровнем сигнала.
- Обновление существующих данных: если данные о Wi-Fi сети уже присутствуют в базе данных, то они обновляются только в том случае, если уровень сигнала связи с данной сетью стал выше.

Таким образом, данный алгоритм направлен на сбор, обработку и сохранение качественных и точных данных о Wi-Fi сетях для последующего использования в задачах определения геолокации пользователя. Точная фильтрация данных и обновление информации только при повышении уровня сигнала позволяют сократить вероятность накопления неточной информации в базе данных.

3.3.2 Алгоритм детекции пользователя

Алгоритм детекции пользователя представляется в следующем виде и предполагает набор последовательных действий:

- Выборка данных: из базы данных извлекается информация о всех доступных Wi-Fi сетях, которая после будет использоваться для вычисления расстояний и определения геолокации пользователя, а также данные о вышках сотовой связи, которые могут помочь в определении геопозиции.
- Определение расстояний до роутеров: на основе заранее известных координат точек доступа Wi-Fi используется формула FSPL для вычисления расстояний от пользователя до каждого роутера. Если общее количество доступных роутеров менее пяти, расстояния учитываются для всех роутеров. В случае, если доступных роутеров более пяти, в вычисления включаются только те, которые находятся на расстоянии менее 100 метров от пользователя.
- Расчет попарных расстояний: затем для каждой Wi-Fi сети вычисляется суммарное расстояние от данной точки до всех остальных сетей как указано в формулах (3.7 - 3.8).

$$D_v = \sum_{u \in W} d(v, u) \quad (3.7)$$

$$Q = \{D_v: \forall v \in W\} \quad (3.8)$$

- Сортировка и фильтрация данных: Все расстояния от пользователя до разных Wi-Fi сетей сортируются. Далее производится отбор тех сетей, которые находятся в диапазоне первых 68% списка. Это позволяет исключить ошибки и улучшить точность алгоритма.

Множество отфильтрованных точек можно записать формулой (3.9):

$$\{v: D_v \leq p \text{ percentile of } Q\}, \quad (3.9)$$

где p - эмпирически подобранная константа, равная 68.

- Вычисление геолокации пользователя: владелец устройства, в соответствии с данными о Wi-Fi сетях, будет где-то рядом с этим набором точек. Отсюда, ориентируя на этих точках и их весам, вычисляется искомая геолокация пользователя как среднее арифметическое от этих точек. Если в итоге получается меньше 3 Wi-Fi сетей, то такой же вес приобретает обнаруженные сотовые вышки. Такой метод позволяет не влиять сотовым вышкам при большом количестве информации от Wi-Fi сетей (например в условиях «известной» городской среды), но также позволяет вне городской среды опираться на данные о видимых сотовых вышках.

Таким образом, данный алгоритм позволяет на основе данных о Wi-Fi сетях и их местоположении, сотовых вышках, полученных от пользователя и сохраненных в базе данных, определить приближенную геолокацию пользователя, применив принципы статистического анализа для уменьшения вероятности ошибки.

3.4 Применение алгоритма

Внедрение разработанного алгоритма имеет широкий спектр возможностей применения. Главным образом, его цель сводится к обеспечению надёжного и точного определения геопозиции пользователя на основе данных о Wi-Fi сетях.

- Усовершенствованный поиск местоположения: точное определение местоположения пользователя — основная функция разработанного алгоритма. Использование данных о Wi-Fi точках в совокупности с применением алгоритмов статистического анализа позволяет улучшить точность и сделать более прогнозируемым определение местоположения.
- Поддержка в местах с плохим сигналом GPS: в среде помещений, где сигнал GPS может быть слабым или отсутствовать, наше приложение

может использовать сети Wi-Fi, чтобы помочь определить геопозицию пользователя.

- Расширенные возможности для разработчиков и коммерческое использование: алгоритм определения геолокации может быть встроен в кастомные Android приложения, требующие высокой точности определения местоположения, например, в приложениях для навигации, туристических гидах, службах доставки, играх с дополненной реальностью и т.д.
- Междисциплинарное применение: массив данных о Wi-Fi сетях и алгоритмы их анализа могут быть использованы в различных сферах, начиная от географии и заканчивая социальными исследованиями.

В целом, применение алгоритма является достаточно гибким и может быть адаптировано под реализацию индивидуальных задач и потребностей.

3.5 Особенности алгоритма

Среди особенностей разработанного алгоритма определения геопозиции могут быть выделены следующие:

- Использование данных Wi-Fi сетей: в качестве основной информации для определения местоположения используются данные о Wi-Fi сетях, что позволяет обеспечить высокую точность даже в помещениях и городских зонах со слабым сигналом GPS.
- Фильтрация точности GPS: если данные пришедшие с устройства превышают порог в 70 метров точности, то мы не будем записывать данные о Wi-Fi вышках, так как можем соотнести им неправильное расположение. Это уменьшает вероятность ошибок алгоритма в будущем.
- Статистический подход: благодаря качественным разработанным инструментам анализа для алгоритма на большом количестве данных, получилось в общем случае улучшить работу алгоритма, а не принимать решения основываясь на отдельных погрешностях.
- Автоматическая обработка и обновление данных: если уровень связи с сетью улучшился в сравнении с предыдущей записью этого Wi-Fi, то произойдет перезапись координат этой точки. Благодаря этому поддерживается актуальность информации в базе.
- Гибкость: достаточно легко портировать программное обеспечение на другие платформы.

В итоге разработанный алгоритм является эффективным решением для исходной задачи, так как показывает высокую точность, гибкость и надежность.

ГЛАВА 4. ТЕСТИРОВАНИЕ И АНАЛИЗ РЕЗУЛЬТАТОВ

Чтобы оценить эффективность и точность работы системы были применены подходы, описанные ниже.

4.1 Методика тестирования

Будем использовать статистику по перцентилям ошибок, так как такой подход будет показывать подробный результат. Перцентиль ошибки - это статистический показатель, который показывает, какой процент наблюдений приходится на конкретную величину ошибки. Это позволяет оценить как распределются ошибки во всех измерениях. Анализ ошибок по перцентилям покажет более полный отчет о распределении ошибок по сравнению со средним значением или стандартным отклонением. Этот подход также укажет на устойчивость алгоритма в критических ситуациях и в сложных случаях.

Применяя вычисления, можно детализировать в каких условиях и как часто возникают ошибки при разных перцентилях [3]:

$$P_k = e \left(\frac{k}{100} \times n \right), \quad (4.1)$$

где:

- P_k - k -й перцентиль,
- $e(x)$ - ошибка на x -ом месте в упорядоченном списке ошибок,
- n - общее число ошибок (измерений).

Благодаря использование перцентильного анализа ошибок геопозиционирования (4.1) можно оценивать верхние границы ошибок с помощью высоких перцентиляй, таких как 95-й и 99-й. Также можно анализировать поведение алгоритма в большинстве случаев с использованием более низких перцентиляй, например, 50-го. Это обеспечивает возможность детального изучения сильных и слабых сторон разработанного алгоритма, предоставляет указания для его улучшений.

4.2 Визуализация данных

Страница с отображением конкретного запроса содержит на карте финальную точку геопозиции и все Wi-Fi точки доступа, используемые в расчетах (рисунок 4.1). Эти точки соединены пунктирными линиями с резуль-

тирующим местоположением, что наглядно демонстрирует ключевые точки, участвующие в процессе определения местоположения.

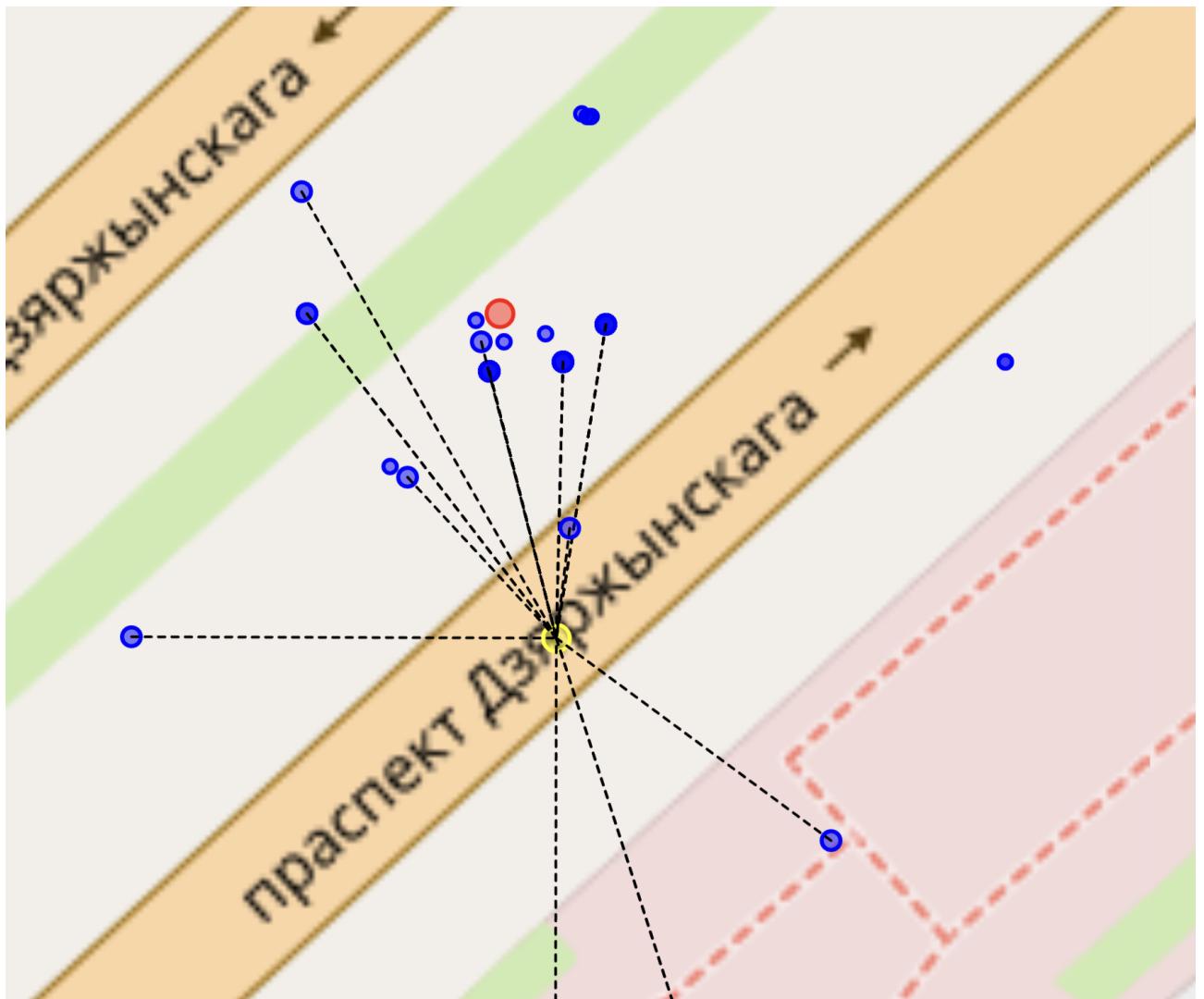


Рисунок 4.1 — Интерфейс, отображающий информацию о запросе на карте

Визуализация позволила увидеть работу алгоритма в разных условиях, например при различной плотности Wi-Fi сетей. Внимательно были рассмотрены запросы, в которых отклонения между результатом алгоритма и реальным местоположением было значительное, что позволило найти «слабые места» алгоритма, которые в последствии были доработаны. Было замечено, что в густонаселенных городских районах, где Wi-Fi сети обладают высокой плотностью и разнообразием, алгоритм демонстрировал высокую точность. В то же время, в местах с небольшим количеством точек доступа или в условиях сложной топографии возникали более весомые ошибки определения геопозиции.

4.3 Заключение по результатам тестирования

В ходе тестирования алгоритма определения геопозиции были обнаружены следующие особенности и результаты:

- Точность определения: алгоритм смог достичь удовлетворительной точности определения геолокации устройства пользователя. Испытания были проведены в городских условиях и вне.
- Надежность и работоспособность: контролировалась работа алгоритма в исключительных ситуациях. В них система успешно справлялась с задачами, а запросы обрабатывались корректно, сбоев программы не наблюдалось.
- Быстродействие: время обработки запросов было в пределах норм и не нарушало комфорт использования приложения пользователем.
- Сравнение с GPS: в ходе сравнения с GPS было выяснено, что в местах, где GPS не мог точно предоставить данные о геопозиции, разработанный алгоритм справлялся успешно, демонстрируя себя как эффективное решение.

Внешние условия и задачи определения местоположения могут отличаться, что влечет более конкретную настройку алгоритма, а также проведения исследования и тестирования в новых условиях.

В результате тестирования было выяснено, что разработанный алгоритм представляет собой надежный инструмент для определения геопозиции, который может улучшить результаты GPS в некоторых условиях. Также были выяснены слабые стороны, в которых возможно его усовершенствование.

Таким образом, в различных прикладных областях, от навигации до служб экстренной помощи возможно применение разработанного алгоритма, что было доказано в результате тестирования.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы было проведено глубокое исследование в области определения геопозиции на основе видимых Wi-Fi сетей и сотовых вышек. В процессе работы были использованы такие комплексные навыки, как программирование на языках Kotlin и Python, работа с базами данных. Было разработано Android-приложение, которое собирает данные с Wi-Fi сетей и мобильных вышек, отправляет их на сервер и возвращает геолокацию на основе полученных данных.

В рамках данного исследования был разработан и опробован новый алгоритм определения геопозиции, который совмещает данные от сотовых сетей и Wi-Fi. В процессе тестирования особое внимание уделялось анализу ряда ключевых факторов, способных влиять на достоверность и точность геолокационных данных. Среди рассматриваемых аспектов были: уровень и качество сигнала Wi-Fi, потенциальные искажения и неточности в данных GPS, а также способность алгоритма корректно интерпретировать получаемую информацию в разнообразных условиях окружающей среды.

Результаты тестирования продемонстрировали, что разработанный алгоритм способен конкурировать с традиционными GPS-системами по ряду показателей. В частности, было установлено, что в условиях городской застройки, где здания и другие препятствия могут существенно влиять на чистоту и стабильность сигнала GPS, предложенный метод обеспечивает сопоставимую, а порой и более высокую, точность определения геопозиции. Это достигается за счет более эффективного использования данных о сотовых сетях и Wi-Fi, которые в совокупности позволяют компенсировать типичные для GPS искажения.

Данные наблюдения подтверждают значимость и актуальность разработанного алгоритма для улучшения качества геолокационных услуг в современных мобильных и геоинформационных системах. Показанная конкурентоспособность с GPS открывает новые возможности для применения разработанного метода в широком спектре приложений, требующих высокую точность и надежность определения местоположения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Горбачёв А. Ю. Математическая модель погрешностей GPS // Авиакосмическое приборостроение. — М.: НАУЧТЕХЛИТИЗДАТ, 2018. — 41 с.
2. Анучин О.Н., Емельянцев Г.И. Интегрированные системы ориентации для морских подвижных объектов / Под ред. В. Г. Пешехонова. — 2-е изд. — СПб.: ГНЦ РФ-ЦНИИ «Электроприбор», 2019. — 390 с.
3. Samama N. Global Positioning: Technologies and Performance. — John Wiley & Sons, 2018. — 440 с.
4. Козловский Е. Искусство позиционирования // Вокруг света. — М., 2018. — 280 с.
5. Серапинас Б.Б. Глобальные системы позиционирования: учебное пособие / Серапинас Б.Б. М.: ИКФ «Каталог», 2002. — 106 с.
6. Яценков В.С. Основы спутниковой навигации / Яценков В.С. М.: Горячая линия-Телеком, 2005. - 271 с.
7. Липкин И. А. Спутниковые навигационные системы / Липкин. И. А. - 2-е изд. - М.: Вузовская книга, 2006. — 288 с.
8. Одуан К., Гино Б., Измерение времени. Основы GPS / Одуан К. М.: Техносфера, 2002. — 399 с.
9. Гончаров И. А. Основы любительской GPS-навигации / Гончаров И. А. М.: Горячая линия — Телеком, 2007. — 127 с.
10. Шебшаевич В. С., Дмитриев П. П., Иванцев Н. В. и др. Сетевые спутниковые радионавигационные системы / под ред. В. С. Шебшаевича. — 2-е изд., перераб. и доп. — М.: Радио и связь, 2020. — 408 с.

ПРИЛОЖЕНИЕ А

ПРОГРАММНЫЙ КОД БЭКЕНД ЧАСТИ

```
...
```

```
# Запрос в OpenCellID для получения позиции сотовой вышки
def get_cell_coordinates(mcc, mnc, lac, cellid):
    url = f"http://opencellid.org/cell/get?key={OPEN_CELL_ID_API_KEY}&mcc={mcc}&mnc={mnc}&lac={lac}&cellid={cellid}&format=json"

    response = requests.get(url)

    if response.status_code == 200:
        data = response.json()
        lat = data.get('lat')
        lon = data.get('lon')
        if lat is not None and lon is not None:
            return lat, lon
        else:
            raise Exception('No cell')
    else:
        raise Exception('Not 200')

# Модель Wi-Fi в БД
class Wifi(db.Model):
    bssid = db.Column(db.String(80), primary_key=True)
    lat = db.Column(db.Numeric(10, 6), nullable=False)
    lon = db.Column(db.Numeric(10, 6), nullable=False)
    level = db.Column(db.Integer, nullable=False)
    frequency = db.Column(db.Float, nullable=False)

...
```

```
# Модель GlobalLog в БД
class GlobalLog(db.Model):
    index = db.Column(db.Integer, primary_key=True, autoincrement=True)
    input_gps = db.Column(db.String(100), nullable=True)
    input_wifis = db.Column(db.String(50000), nullable=True)
    input_cells = db.Column(db.String(5000), nullable=True)
    output = db.Column(db.String(5000), nullable=False)
    error_code = db.Column(db.Integer, nullable=False)
    queries = db.Column(db.String(50000), nullable=True)
    timestamp = db.Column(db.Integer, nullable=False)
    algo = db.Column(db.String(10), nullable=False)

...
```

```
# Получение всех Wi-Fi в БД
@app.route('/get_all', methods=['GET'])
def get_all():
    wifis = Wifi.query.all()
    response = {
        "wifi": []
    }
    for wifi in wifis:
        response["wifi"].append({"lat": wifi.lat, "lon": wifi.lon})
    return jsonify(response)

# Функция для scanning - добавление новых данных в БД
@app.route('/lbs', methods = ['POST'])
def lbs():
```

```

...
for item in data:
    ...
    if not wifi:
        new_wifi = Wifi(**new_wifi_values)
        db.session.add(new_wifi)
        cnt += 1
    # Если сигнал Wi-Fi лучше - перезаписываем
    elif new_wifi_values['level'] < wifi.level:
        wifi.lat = new_wifi_values['lat']
        wifi.lon = new_wifi_values['lon']
        wifi.frequency = new_wifi_values['frequency']
        wifi.level = new_wifi_values['level']
        upd += 1

db.session.commit()

return jsonify({"count": cnt, "upd": upd}), 200

# функция гаеверсина
def dist(lat1, lon1, lat2, lon2):
    R = 6371000 # Earth radius
    f1 = lat1 * pi / 180.
    f2 = lat2 * pi / 180.
    delta_f = (lat2 - lat1) * pi / 180.
    delta_l = (lon2 - lon1) * pi / 180.
    a = sin(delta_f/2) ** 2 + cos(f1) * cos(f2) * (sin(delta_l/2) ** 2)
    c = 2 * atan2(sqrt(a), sqrt(1 - a))
    return R * c

# функция определения расстояния по FSPL
def calculateDistance(level, frequency):
    return 10.0 ** ((27.55 - (20 * log10(frequency)) - level)/20)

...
# Алгоритм демекции
def run_algo(data, filter=[True, True]):
    queries = []
    if not data:
        return get_error_output()

    cells = []
    for item in data['cellList']:
        mcc = item.get('mcc')
        mnc = item.get('mnc')
        lac = item.get('lac')
        cell_id = item.get('cellId')
        if mcc != 2147483647:
            try:
                coordinates = get_cell_coordinates(mcc, mnc, lac, cell_id)
            except Exception:
                continue
            queries.append({"cell": str(coordinates)})
            # , "level": float(item.get('level'))
            cells.append({"lat": float(coordinates[0]), "lon": float(coordinates[1])})

    wifis = []

```

```

for item in data['wifiList']:
    bssid = item.get('BSSID')
    wifi = Wifi.query.get(bssid)
    if wifi is not None:
        queries.append({"wifi": wifi})
        wifis.append({"lat": float(wifi.lat), "lon": float(wifi.lon), "level": float(item.get('level')), "frequency": float(item.get('frequency'))})
    if len(wifis) == 0:
        return get_error_output(code=404)

levels = []
for index, first in enumerate(wifis):
    dist_from_level = calculateDistance(first["level"], first["frequency"])
    if len(wifis) <= 5 or dist_from_level <= 100.0:
        levels.append((dist_from_level, index))
levels = sorted(levels)
cnt = max(1, int(float(len(levels)) * 0.68)) if filter[0] else len(levels)
new_wifis = []
for c in range(cnt):
    new_wifis.append(wifis[levels[c][1]])
wifis = new_wifis

distances = []
for index, first in enumerate(wifis):
    summary = 0.
    for second in wifis:
        summary += dist(first["lat"], first["lon"], second["lat"], second["lon"])
    distances.append((summary, index))
distances = sorted(distances)

cnt = max(1, int(float(len(distances)) * 0.68)) if filter[1] else len(distances)

new_wifis = []
sumlat = 0.
sumlon = 0.
for i in range(cnt):
    sumlat += wifis[distances[i][1]]["lat"]
    sumlon += wifis[distances[i][1]]["lon"]
    new_wifis.append(wifis[distances[i][1]])

return {"lat": sumlat / cnt, "lon": sumlon / cnt, "count": cnt, "wifi": new_wifis, "cells": cells}, 200, queries
...
# Запуск алгоритма с разными вариациями фильтрации
def run_algo_v0(data, filter=[True, True]):
    output = run_algo(data, filter)
    fill_log(f"v0_{filter[0]}_{filter[1]}", data, output)
    db.session.add(log)
    db.session.commit()
    return output

```

```

# Функция для детекции расположения пользователя
@app.route('/detect', methods = ['POST'])
def detect():
    data = request.get_json()
    output = None
    for filter_0 in [True, False]:
        for filter_1 in [True, False]:
            cur_output = run_algo_v0(data, filter=[filter_0, filter_1])
            if not output:
                output = cur_output
    return jsonify_response(output)

# Функция отображения главной страницы
@app.route('/')
def index():
    page = request.args.get('page', 1, type=int)
    per_page = 5
    logs_pagination = GlobalLog.query.paginate(page, per_page, error_out=False)
    return render_template('index.html', logs_pagination=logs_pagination)

# Функция для просмотра каждого запроса
@app.route('/item/<int:index>')
def item(index):
    ...
    for v in output['wifi']:
        ll.add((v['lat'], v['lon']))
    r_queries = []
    # Вычисление Wi-Fi поэлияющих на результатах
    for d in queries:
        for k, v in d.items():
            if not (k == "wifi" and (v['lat'], v['lon']) in ll):
                r_queries.append(d)

    ...
    return render_template('item.html', log=dict_log)

# Функция для посчета статистики
def fetch_data():
    unique_algos = db.session.query(GlobalLog.algo).distinct().all()
    results = []
    for algo in unique_algos:
        algo_name = algo[0]
        entries = GlobalLog.query.filter_by(algo=algo_name).order_by(
            GlobalLog.timestamp.desc()).limit(1000).all()
        distances = []
        for entry in entries:
            input_gps = json.loads(entry.input_gps.replace("\'", "\""))
            output = json.loads(entry.output.replace("\'", "\""))
            distance = dist(input_gps['lat'], input_gps['lon'], output['lat'],
                output['lon']))
            distances.append(distance)
        percentiles = [f"{p:.2f}" for p in np.percentile(distances, [25, 50,
            75, 90, 95, 99])]
        results.append({'algo': algo_name, 'percentiles': percentiles})
    return results

# Страница подсчета статистики
@app.route('/stats')
def stats():

```

```
data = fetch_data()
return render_template('stats.html', data=data)

# Функция карты - отображения всех Wi-Fi
@app.route('/map', methods=['GET'])
def map():
    wifis = Wifi.query.all()
    return render_template('map.html', wifis=wifis)
```

ПРИЛОЖЕНИЕ Б

ПРОГРАММНЫЙ КОД ANDROID-ПРИЛОЖЕНИЯ

...

```
class MainActivity : AppCompatActivity() {
    ...

    // Функция добавления точки на карту
    private fun setPoint(lat: Double, lon: Double, color: Int = Color.RED,
    setCenter: Boolean = false) {
        mapController.setZoom(18.0)
        val myLocation = GeoPoint(
            lat,
            lon
        )
        if (setCenter) {
            mapController.setCenter(myLocation)
        }
        val myMarker = OverlayItem("Title", "Description", myLocation)
        val d = ShapeDrawable(OvalShape())
        d.intrinsicHeight = 50
        d.intrinsicWidth = 50
        d.paint.color = color
        myMarker.setMarker(d)

        val markersOverlay = ItemizedOverlayWithFocus<OverlayItem>(
            this,
            listOf(myMarker),
            object :
                ItemizedIconOverlay.OnItemGestureListener<OverlayItem> {
                override fun onItemSingleTapUp(
                    index: Int,
                    item: OverlayItem
                ): Boolean {
                    return true
                }

                override fun onItemLongPress(
                    index: Int,
                    item: OverlayItem
                ): Boolean {
                    return false
                }
            }
        )
        markersOverlay.setFocusItemsOnTap(true)
        map.overlays.add(markersOverlay)
    }

    //Функция для получения разрешений Android-приложению
    override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions,
        grantResults)
    }

    // Функция для сканирования вышек
    private fun scanCellTowers(): List<GsmResultVO> {
        val results = mutableListOf<GsmResultVO>()
```

```

        val telephonyManager = getSystemService(Context.TELEPHONY_SERVICE)
        as TelephonyManager
        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.
        ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED ||
            ActivityCompat.checkSelfPermission(this, Manifest.permission.
        ACCESS_COARSE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
            val cellInfoList: List<CellInfo>? = telephonyManager.allCellInfo
            cellInfoList?.forEach { cellInfo ->
                if (cellInfo is CellInfoGsm) {
                    val cellIdentity = cellInfo.cellIdentity
                    val cellSignalStrength = cellInfo.cellSignalStrength
                    results.add(
                        GsmResultVO(
                            mcc = cellIdentity.mcc,
                            mnc = cellIdentity.mnc,
                            lac = cellIdentity.lac,
                            cellId = cellIdentity.cid,
                            level = cellSignalStrength.dbm
                        )
                    )
                } else if (cellInfo is CellInfoLte) {
                    val cellIdentity = cellInfo.cellIdentity
                    val cellSignalStrength = cellInfo.cellSignalStrength
                    results.add(
                        GsmResultVO(
                            mcc = cellIdentity.mcc,
                            mnc = cellIdentity.mnc,
                            lac = cellIdentity.tac,
                            cellId = cellIdentity.ci,
                            level = cellSignalStrength.dbm
                        )
                    )
                }
            }
        }
    }

    return results
}

// Отрисовка интерфейса
@SuppressLint("UseSwitchCompatOrMaterialCode")
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    ...

    // Логика нажатия на основную кнопку
    button.setOnClickListener {
        val wifiList = getWifiList()
        fusedLocationClient.lastLocation
            .addOnSuccessListener { location: Location? ->
                if (location == null) {
                    return@addOnSuccessListener
                }
                if (!switch.isChecked) {
                    map.overlays.clear()
                    setPoint(location.latitude, location.longitude,
                    setCenter = true)
                    val accuracyInMeters = location.accuracy
                }
            }
    }
}

```

```

        if (accuracyInMeters > 70) {
            makeToast("Too inaccurate: $accuracyInMeters")
            return@addOnSuccessListener
        }
    }

    if (switch.isChecked) {
        postDetect(wifiList, location)
    } else {
        postDataToServer(wifiList, location)
    }
}
}

// Логика нажатия на кнопку отображения всех Wi-Fi
buttonShowAll.setOnClickListener {
    getAllWifi()
}
}

...
}

// Получения списка Wi-Fi
private fun getWifiList(): List<ScanResult> {
    val wifiManager = applicationContext.getSystemService(Context.WIFI_SERVICE) as WifiManager
    if ((ActivityCompat.checkSelfPermission(
            this,
            Manifest.permission.ACCESS_WIFI_STATE
        ) != PackageManager.PERMISSION_GRANTED) && (ActivityCompat.checkSelfPermission(
            this,
            Manifest.permission.ACCESS_FINE_LOCATION
        ) != PackageManager.PERMISSION_GRANTED)) {
        ActivityCompat.requestPermissions(this@MainActivity, arrayOf(
            Manifest.permission.ACCESS_WIFI_STATE, Manifest.permission.ACCESS_FINE_LOCATION), 1)
    }
    return wifiManager.scanResults;
}

// Схема результатов от сканирования Wi-Fi
data class ScanResultVO(
    val SSID: String,
    val BSSID: String,
    val capabilities: String,
    val frequency: Int,
    val level: Int,
    val lat: Double,
    val lon: Double
) {
    ...
}

// Схема геопозиции по GPS
data class InputGps(
    val lat: Double,
    val lon: Double
)

```

```

// Схема с комбинацией разных типов сканирования
data class CombinedScanResult(
    val wifiList: List<ScanResultVO>,
    val cellList: List<GsmResultVO>,
    val inputGps: InputGps
) {
}

// Схема сканирования GSM Cells
data class GsmResultVO(
    val mcc: Int,
    val mnc: Int,
    val lac: Int,
    val cellId: Int,
    val level: Int,
    val lat: Double = 0.0,
    val lon: Double = 0.0
) {
}

// Схема ответа сервера
data class LbsResponse(
    var count: String,
    var upd: String
)

// Функция запроса к серверу для сканирования
private fun postDataToServer(wifiList: List<ScanResult>, location: Location?) {
    ...

    val retrofit = Retrofit.Builder()
        .baseUrl("https://kmekhovichlbs.pythonanywhere.com/")
        .addConverterFactory(GsonConverterFactory.create())
        .client(client)
        .build()

    val wifiScanResultVOList = wifiList.map { ScanResultVO(it, location) }
    val service = retrofit.create(ApiService::class.java)

    val call = service.sendWifiScanResult(wifiScanResultVOList)

    call.enqueue(object : retrofit2.Callback<LbsResponse> {
        override fun onResponse(call: retrofit2.Call<LbsResponse>, response: Response<LbsResponse>) {
            if (response.isSuccessful) {
                val lbsResponse = response.body()

                if (lbsResponse != null) {
                    makeToast("Scanned ${lbsResponse.count} new wifi.
Updated ${lbsResponse.upd}")
                }
            }
        }

        override fun onFailure(call: retrofit2.Call<LbsResponse>, t: Throwable) {
            makeToast("got error $t")
        }
    })
}

```

```

        })
    }

    // Схема получения Wi-Fi
    data class Wifi(
        var lat: String,
        var lon: String
    )

    // Схема получения GSM Cells
    data class Cell(
        var lat: String,
        var lon: String
    )

    // Функция для отрисовки Wi-Fi на карте
    private fun drawWifi(wifiList: List<Wifi>, color: Int = Color.BLUE) {
        for (wifi in wifiList) {
            val myLocation = GeoPoint(
                wifi.lat.toDouble(),
                wifi.lon.toDouble()
            )
            val myMarker = OverlayItem("Title", "Description", myLocation)
            val d = ShapeDrawable(OvalShape())
            d.intrinsicHeight = 10
            d.intrinsicWidth = 10
            d.paint.color = color
            myMarker.setMarker(d)
            val markersOverlay = ItemizedOverlayWithFocus<OverlayItem>(
                this,
                listOf(myMarker),
                object :
                    ItemizedIconOverlay.OnItemGestureListener<OverlayItem> {
                    override fun onItemSingleTapUp(
                        index: Int,
                        item: OverlayItem
                    ): Boolean {
                        return true
                    }

                    override fun onItemLongPress(
                        index: Int,
                        item: OverlayItem
                    ): Boolean {
                        return false
                    }
                }
            )
            map.overlays.add(markersOverlay)
        }
    }

    // Функция для отрисовки GSM Cells на карте
    private fun drawCells(cellList: List<Cell>, color: Int = Color.BLACK) {
        for (cell in cellList) {
            val myLocation = GeoPoint(
                cell.lat.toDouble(),
                cell.lon.toDouble()
            )
            val myMarker = OverlayItem("Title", "Description", myLocation)
        }
    }
}

```

```

        val d = ShapeDrawable(OvalShape())
        d.intrinsicHeight = 50
        d.intrinsicWidth = 50
        d.paint.color = color
        myMarker.setMarker(d)
        val markersOverlay = ItemizedOverlayWithFocus<OverlayItem>(
            this,
            listOf(myMarker),
            object :
                ItemizedIconOverlay.OnItemGestureListener<OverlayItem> {
                    override fun onItemSingleTapUp(
                        index: Int,
                        item: OverlayItem
                    ): Boolean {
                        return true
                    }

                    override fun onItemLongPress(
                        index: Int,
                        item: OverlayItem
                    ): Boolean {
                        return false
                    }
                }
        )
        map.overlays.add(markersOverlay)
    }

// Схема ответа на запрос детекции
data class LocationResponse(
    var lat: String,
    var lon: String,
    var count: String,
    var wifi: List<Wifi>,
    var cells: List<Cell>
)

// Функция гаверсинуса для расчета ошибки
private fun Distance(lat1: Double, lon1: Double, lat2: Double, lon2: Double): Double {
    val R = 6371000
    val f1 = lat1 * PI / 180.0
    val f2 = lat2 * PI / 180.0
    val delta_f = (lat2 - lat1) * PI / 180.0
    val delta_l = (lon2 - lon1) * PI / 180.0
    val a = sin(delta_f / 2) * sin(delta_f / 2) + cos(f1) * cos(f2) * (
        sin(delta_l / 2) * sin(delta_l / 2))
    val c = 2 * atan2(sqrt(a), sqrt(1 - a))
    return R * c
}

// Функция для запроса детекции
private fun postDetect(wifiList: List<ScanResult>, location: Location) {
    val client = OkHttpClient.Builder().build()

    val retrofit = Retrofit.Builder()
        .baseUrl("https://kmekhovichlbs.pythonanywhere.com/")
        .addConverterFactory(GsonConverterFactory.create())
        .client(client)
}

```

```

        .build()

    val wifiScanResultVOList = wifiList.map { ScanResultVO(it) }
    val cellTowersResultVOList = scanCellTowers()
    val service = retrofit.create(ApiService::class.java)

    val combinedScanResult = CombinedScanResult(wifiScanResultVOList,
    cellTowersResultVOList, InputGps(location.latitude, location.longitude))
    val call = service.sendDetect(combinedScanResult)

    call.enqueue(object : retrofit2.Callback<LocationResponse> {
        override fun onResponse(call: retrofit2.Call<LocationResponse>,
        response: Response<LocationResponse>) {
            Log.e("r", response.toString())
            if (response.isSuccessful) {
                val locationResponse = response.body()

                if (locationResponse != null) {
                    val lat = locationResponse.lat.toDouble()
                    val lon = locationResponse.lon.toDouble()

                    map.overlays.clear()
                    setPoint(location.latitude, location.longitude,
setCenter = !switch.isChecked)
                    setPoint(lat, lon, Color.YELLOW, setCenter = true)

                    makeToast("Location based on ${locationResponse.
count} wifi. Error is %.2f meters. Found ${locationResponse.cells.size}
cells".format(Distance(lat, lon, location.latitude, location.longitude)))
                    drawWifi(locationResponse.wifi.toList())
                    drawCells(locationResponse.cells.toList())
                }
            }
        }

        override fun onFailure(call: retrofit2.Call<LocationResponse>, t
: Throwable) {
            makeToast("got error $t")
        }
    })
}

// Функция для запроса по списку всех Wi-Fi
private fun getAllWifi() {
    val client = OkHttpClient.Builder().build()

    val retrofit = Retrofit.Builder()
        .baseUrl("https://kmekhovichlbs.pythonanywhere.com/")
        .addConverterFactory(GsonConverterFactory.create())
        .client(client)
        .build()

    val service = retrofit.create(ApiService::class.java)

    val call = service.getAllWifi()

    call.enqueue(object : retrofit2.Callback<LocationResponse> {
        override fun onResponse(call: retrofit2.Call<LocationResponse>,
        response: Response<LocationResponse>) {
            if (response.isSuccessful) {

```

```

        val locationResponse = response.body()

        if (locationResponse != null) {
            var wifiList = locationResponse.wifi.toList()
            makeToast("Got ${wifiList.size} wifis")
            drawWifi(wifiList, Color.GREEN)
        }
    }
}

override fun onFailure(call: retrofit2.Call<LocationResponse>, t
: Throwable) {
    makeToast("got error $t")
}
}

...
// API общения с сервером
interface ApiService {
    @POST("lbs")
    fun sendWifiScanResult(@Body wifiList: List<ScanResultVO>): retrofit2.Call<LbsResponse>

    @POST("detect")
    fun sendDetect(@Body combinedScanResult: CombinedScanResult): retrofit2.Call<LocationResponse>

    @GET("get_all")
    fun getAllWifi(): retrofit2.Call<LocationResponse>
}
}

```