

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего
образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

Кафедра

вычислительной техники

«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация
автоматизированных систем специального
назначения»
Специализация: «Эксплуатация вычислительных машин,
комплексов, систем и сетей специального
назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

**МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ ОПРЕДЕЛЕНИЯ
РАСПОЛОЖЕНИЯ ВЫШЕК СОТОВОЙ СВЯЗИ**

Студент Верченко Сергей Витальевич
(подпись, дата) 15.06.2025 (ФИО полностью)

Руководитель Егоров В.Ю.
(подпись, дата) 15.06.2025 (фамилия, инициалы)

Нормоконтролёр Бычков А.С.
(подпись, дата) 15.06.2025 (фамилия, инициалы)

Рецензент Сивостянов Р.Р.
науч. сотрудник
ИТ (должность, место работы) (подпись, дата) 18.06.2025 (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ 2025г. № _____)

Заведующий кафедрой Митрохин М.А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой удовлетворительно
(протокол заседания ГЭК от 19.06.2025 № 8)

Секретарь ГЭК Гурин Е.И.
(подпись) (фамилия, инициалы)

Пенза, 2025

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

Факультет
Вычислительной техники

Кафедра
Вычислительная техника

«Утверждаю»
Заведующий кафедрой
Митрохин М.А.
«6» мая 2025 г.

ЗАДАНИЕ НА ДИПЛОМНЫЙ ПРОЕКТ

1. Студент Верченко Сергей Витальевич гр. 20ВОЭ1 факультета вычислительной техники.
2. Тема работы: «Мобильное приложение для определения расположения вышек сотовой связи».

Тема утверждена приказом ПГУ №601/о от 06.05.2025 г.

3. Руководитель работы к.т.н., доцент каф. ВТ «ПГУ» Егоров В.Ю.

4. Задание на работу (назначение разработки, исходные данные и т.п.).

Разработать бесплатное мобильное приложение для определения информации о вышках сотовой связи.

Приложение должно выполнять следующие функции: отображение на интерактивной карте информации о вышках сотовой связи, возможность настройки информации о мобильных операторах из открытых источников, сохранение информации о мобильных операторах в локальной СУБД. Приложение должно иметь возможность работы с интерактивной картой. Программное обеспечение реализуется с помощью среды разработки Android Studio. В качестве СУБД используется компактная встраиваемая СУБД SQLite.

5. Перечень подлежащих разработке вопросов.

Требуется произвести анализ предметной области, определить функциональный состав программы, выбрать стек технологий для реализации программы, разработать диаграммы: вариантов использования и развертывания, спроектировать модель базы данных, разработать и реализовать архитектуру мобильного приложения.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	10	06.05.25 – 08.05.25	<i>BS</i>
2	Проектирование и отладка программного обеспечения	10	08.05.25 – 09.05.25	<i>BS</i>
3	Разработка базы данных	20	09.05.25 – 11.05.25	<i>BS</i>
4	Отладка и тестирование	10	11.05.25 – 17.05.25	<i>BS</i>
5	Разработка экономической и БЖД частей	10	17.05.25 – 19.05.25	<i>BS</i>
6	Оформление графической части	10	19.05.25 – 25.05.25	<i>BS</i>
7	Оформление выпускной работы	30	25.05.25 – 30.05.25	<i>BS</i>

Дата выдачи задания «06» мая 2025г.

Руководитель дипломной работы _____

BS 06.05.2025 Егоров К.Ю.
(подпись, дата)

Задание к исполнению принял студент _____

BS 06.05.2025 Афанасов С.В.
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____

BS 18.06.2025 Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 110 листов, 35 рисунков, 14 таблиц и 3 приложения.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ГЕОЛОКАЦИЯ, OPENCCELLID, БАЗА ДАННЫХ, СОТОВАЯ СВЯЗЬ.

Объектом исследования являются технологии определения местоположения вышек сотовой связи и их интеграция в мобильные приложения.

Цель работы – разработать мобильное приложение для визуализации и анализа информации о расположении вышек сотовой связи на основе открытой базы данных OpenCellID.

В процессе работы выполнен анализ современных решений, сформулированы требования к функционалу приложения, выбраны инструменты разработки, спроектирована архитектура системы и реализовано работоспособное приложение в соответствии с требованиями технического задания.

В результате дипломного проектирования создано мобильное приложение, предоставляющее пользователям информацию о ближайших вышках, уровне сигнала и зонах покрытия мобильного оператора связи. База данных интегрирована с API OpenCellID и Google Maps, интерфейс реализован с использованием языка программирования Kotlin и СУБД SQLite.

Разработанное приложение имеет рабочее название Cell Tower Locator.

					ПГУ1.090501.05.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата	Мобильное приложение для определения расположения вышек сотовой связи	Лит.	Лист	Листов	
Разраб.		Верченко С.В.		18.06.25					
Провер.		Егоров В.Ю.		18.06.25			2	114	
Н. Контр		Бычков А.С.		18.06.25					
Утверд.		Митрохин М.А.		18.06.25	Пояснительная записка	ФВТ, каф. ВТ			
						гр. 20В0Э1			

Содержание

Перечень сокращений и обозначений.....	7
Введение.....	8
1. Анализ предметной области	9
1.1 Актуальность предметной области и ее текущее состояние	9
1.2 Цели и задачи разрабатываемого программного обеспечения	10
1.3 Целевая аудитория и сценарии использования.....	11
1.3.1 Сценарии использования	11
1.3.2 Варианты использования приложения	15
2 Техническое задание	20
2.1 Назначение разработки.....	20
2.1.1 Требования к функциональным характеристикам	20
2.1.2 Требования к надежности	21
2.1.3 Требования к составу и параметрам технических средств.....	21
2.1.4 Требования к информационной и программной совместимости	22
2.1.5 Требования к программной документации	22
2.1.6 Порядок контроля и приемки	22
3 Анализ аналогов решений для работы с данными вышек сотовой связи	24
4. Анализ стека технологий для разработки мобильного приложения	30
4.1 Сравнение мобильных ОС	30
4.2 Сравнение средств разработки	32
4.3 Сравнение СУБД.....	33
4.4 Сравнение сред разработки (IDE)	34
5 Разработка мобильного приложения Cell Tower Locator	36
5.1 Проектирование приложения Cell Tower Locator.....	39
5.2 Проектирование базы данных приложения Cell Tower Locator.....	44
5.3 Прототипирование интерфейса приложения Cell Tower Locator	49
5.4 Программная реализация приложения Cell Tower Locator.....	52
6 Описание интерфейса приложения Cell Tower Locator	59

7 Экономическое обоснование эффективности проекта.....	70
8 Безопасность жизнедеятельности при работе инженера-программиста.....	75
8.1 Эргономические требования к рабочему месту	75
8.2 Требования к микроклимату и освещению рабочего помещения	77
8.3 Организация безопасной работы с персональным компьютером.....	78
Заключение	80
Список использованных источников	81
Приложение А – UML – диаграммы	83
Приложение А.1 – Диаграмма вариантов использования.....	84
Приложение А.2 – Диаграмма развертывания.....	85
Приложение А.3 – Диаграмма последовательности.....	86
Приложение А.4 – Диаграмма последовательности.....	87
Приложение А.5 – Диаграмма данных.....	88
Приложение Б – Листинг программы	89
Приложение В – Презентация.....	97

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

ПО – программное обеспечение

ОС – операционная система

ПК – персональный компьютер

СУБД – система управления базами данных

API – программный интерфейс, то есть описание способов взаимодействия одной компьютерной программы с другими

IDE (Integrated Development Environment) - интегрированная среда разработки

SQL – обозначение широкого класса систем управления базами данных

SDK – набор инструментов для разработки программного обеспечения, объединённый в одном пакете

UML (Unified Modeling Language) – унифицированный язык моделирования, язык графического описания для объектного моделирования

					<i>ПГУ1.090501.05.001 ПЗ</i>	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Современные технологии мобильной связи требуют точного определения зон покрытия и оптимизации сетевой инфраструктуры. Однако пользователи часто сталкиваются с проблемой слабого сигнала, что делает актуальным создание инструментов для анализа расположения вышек сотовой связи.

Существующие решения (например, Network Cell Info, OpenSignal) предоставляют базовую информацию, но имеют ограничения: закрытые API, платный доступ к данным или отсутствие детализации. Это обуславливает необходимость разработки открытого и гибкого приложения, использующего crowdsourcing-данные, например из OpenCellID.

Цель дипломной работы – создание мобильного приложения для визуализации вышек сотовой связи с функциями фильтрации по операторам, отображения зон покрытия и интеграции с картографическими сервисами.

Задачи дипломной работы:

Анализ возможностей OpenCellID и сравнение с аналогами (Mozilla Location Service, GSM Arena и иными решениями).

Проектирование архитектуры приложения.

Реализация механизма обновления данных через API OpenCellID.

Тестирование функционала на различных мобильных платформах.

Приложение предназначено для исследователей сотовой связи, инженеров-сетевиков и обычных пользователей, желающих оптимизировать использование мобильного интернета и принять участие в бесплатном распространении информации и программного обеспечения.

1. Анализ предметной области

1.1 Актуальность предметной области и ее текущее состояние

Рынок мобильных приложений для анализа сотовых сетей демонстрирует устойчивый рост, что обусловлено следующими факторами:

– рост числа мобильных устройств:

1) по данным GSMA Intelligence [1], к 2025 году количество подключенных устройств достигнет 25 млрд, включая IoT-гаджеты и смартфоны;

2) потребление мобильного трафика увеличивается на 30% ежегодно (Ericsson Mobility Report, 2024) [2];

– развертывание 4G и 5G:

1) к 2025 году 5G покроет 65% мирового населения (Statista, 2024) [3];

2) пользователи и операторы нуждаются в инструментах для оптимизации размещения вышек и анализа «мертвых зон»;

– зависимость от качества связи:

1) 78% пользователей считают стабильный сигнал критически важным для работы и повседневной жизни (Pew Research Center, 2024);

– ограничения существующих решений:

1) многие приложения (например, OpenSignal) используют проприетарные данные, что ограничивает прозрачность;

2) отсутствие интеграции с открытыми базами данных (OpenCellID, Mozilla Location Service) затрудняет кастомизацию приложений;

– технологические тренды, влияющие на актуальность дипломной работы:

1) crowdsourcing данных: OpenCellID позволяет пользователям добавлять новые вышки, что повышает точность покрытия;

2) gis-интеграция: использование картографических сервисов (Google Maps API, Mapbox) упрощает визуализацию данных;

3) open source: разработка на базе открытых технологий (Java, Flutter Sqlite, Room и другие) снижает затраты и ускоряет внедрение;

– приложение помогает:

1) пользователям найти зоны с лучшим сигналом для звонков или интернета;

2) инженерам оптимизировать размещение вышек для операторов связи;

3) исследователям анализировать распределение сетей 4G/5G в городской и сельской местности.

Разработка заявленного мобильного приложения актуальна благодаря росту спроса на бесплатные инструменты анализа сотовых сетей, доступности открытых данных и необходимости преодоления ограничений существующих коммерческих решений.

1.2 Цели и задачи разрабатываемого программного обеспечения

Цели и задачи программного обеспечения:

– возможность бесплатного анализа состояния технической инфраструктуры мобильных операторов;

– обеспечение контроля над инфраструктурой вышек сотовой связи;

в результате внедрения программного обеспечения планируется достичь следующих задач:

– повысить роль бесплатного свободно распространяемого обеспечения;

– предоставить возможность пользователям производить мониторинг покрытия операторами сотовой связи;

– предоставить инженерам доступ к свободно распространяемым данным о сотовой связи для улучшения качества ее обслуживания.

1.3 Целевая аудитория и сценарии использования

Целевая аудитория разрабатываемого программного приложения определяет круг пользователей, для которых предназначено ПО.

Приложение Cell Tower Locator предназначено для следующих групп пользователей:

- технические специалисты и инженеры телеком-компаний. Используют приложение для анализа покрытия сети, проверки расположения вышек и диагностики проблем связи;
- разработчики мобильных приложений. Интегрируют данные OpenCellID в свои приложения для геолокации или анализа качества связи;
- обычные пользователи. Оценивают качество покрытия в своем районе, находят ближайшие вышки для улучшения сигнала.

1.3.1 Сценарии использования

Для технических специалистов и инженеров целью использования является проверка инфраструктуры сети и устранение проблем покрытия.

Сценарий для технических специалистов:

- автоматический поиск вышек. Приложение собирает данные о ближайших вышках при перемещении по территории;
- анализ на карте. На карте отображаются вышки с информацией о типе сети (2G/3G/4G/5G) и операторе;
- сравнение данных. Возможность сохранить историю поиска и сравнить данные за разные периоды для выявления изменений в инфраструктуре;
- экспорт данных. Экспорт координат вышек в CSV для последующего анализа в GIS-системах;

пример реализации сценария:

					ПГУ1.090501.05.001 ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

– инженер оператора связи использует приложение во время выезда на объект. Приложение показывает отсутствие вышек 4G в заданном районе, что помогает обосновать необходимость установки новой станции;

– для разработчиков мобильных приложений целью использования ПО является интеграция данных OpenCellID в сторонние приложения;

сценарий для разработчиков приложений:

– тестирование геолокации. Использование API приложения для проверки работы геолокации без GPS;

– получение технических параметров. Запрос данных о частотах, мощности сигнала и операторах через REST API;

– кэширование данных. Сохранение данных в локальную базу SQLite для оффлайн-доступа;

– настройка уведомлений. Интеграция системы уведомлений о изменении параметров сети.

пример реализации сценария:

– разработчик навигационного приложения использует API Cell Tower Locator для определения местоположения пользователя в помещении, где GPS недоступен;

– для обычных пользователей целью использования ПО проверка качества связи и поиск оптимального места для использования мобильного интернета;

сценарий для обычных пользователей:

– поиск ближайших вышек. Пользователь получает список станций с информацией об операторе и типе сети;

– просмотр на карте. Маркеры на карте помогают визуально определить зону покрытия;

– история запросов. Сохранение истории для отслеживания изменений в сети;

– упрощенные настройки. Автоматический выбор оптимальных параметров поиска (радиус N км, приоритет текущего оператора);

					ПГУ1.090501.05.001 ПЗ	Лист
						13
Изм.	Лист	№ докум.	Подпись	Дата		

пример реализации сценария:

- пользователь, испытывающий проблемы со связью дома, находит ближайшую вышку 4G;

- меняет положение роутера для улучшения сигнала;

общие сценарии пользователей:

- обработка данных в реальном времени. Приложение автоматически обновляет данные о вышках при изменении местоположения пользователя;

- анализ скорости работы. Замер времени отклика `api opencellid` и сравнение с кэшированными данными;

- работа в оффлайн-режиме. Просмотр ранее сохраненных данных о вышках без доступа к интернету;

- интеграция с картографическими сервисами. Экспорт координат вышек в `google maps` или `яндекс.карты`.

Сценарии использования необходимо визуализировать UML диаграммой использования (рисунок 1).

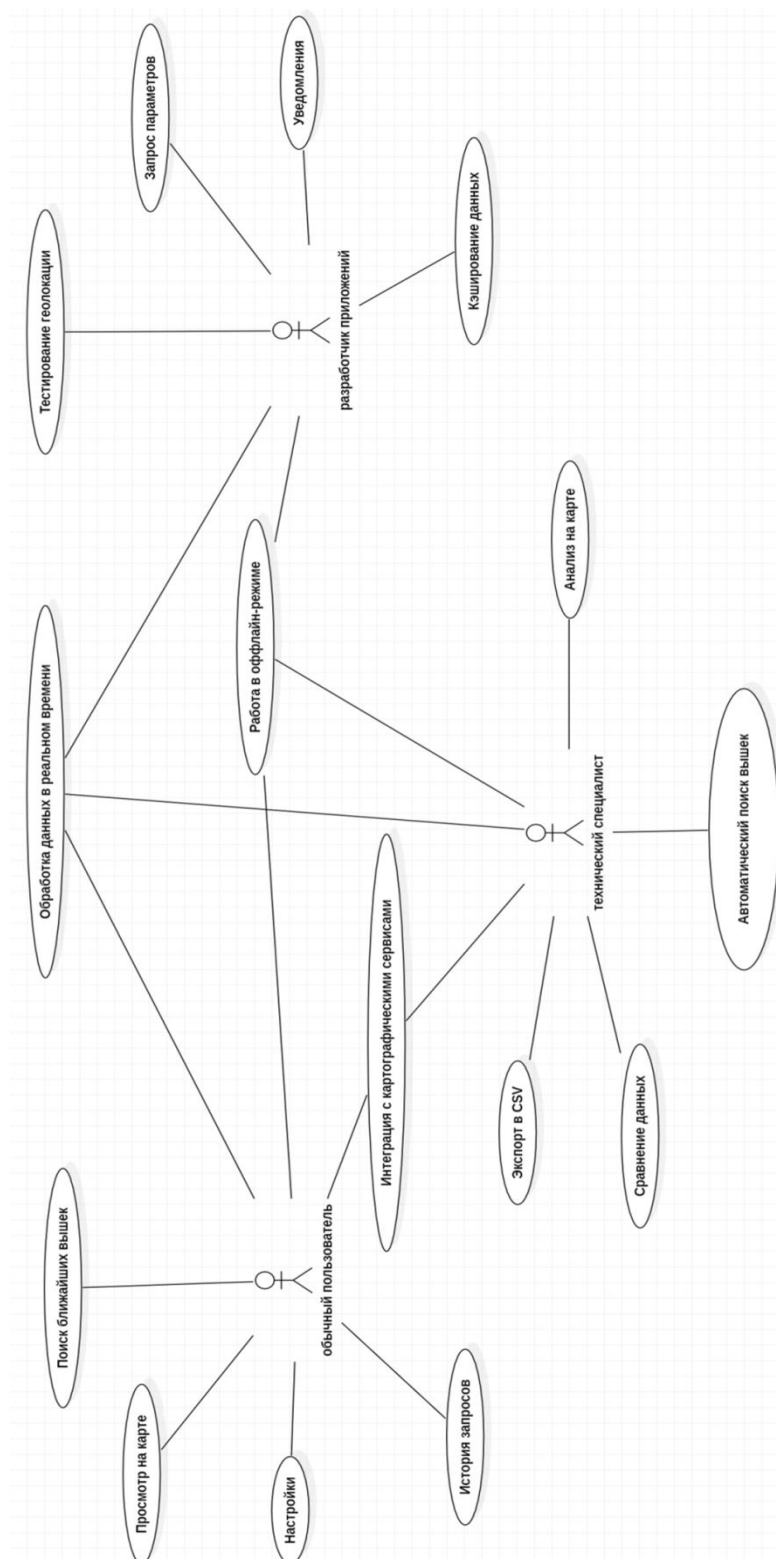


Рисунок 1 — Диаграмма вариантов использования

1.3.2 Варианты использования приложения

На основании пользовательских сценариев реализована UML диаграмма вариантов использования (рисунок 1). Опишем ее акторы и связи для детального процесса использования приложения:

– акторы:

- 1) технический специалист/Инженер. Цель: Анализ покрытия сети, диагностика проблем, обоснование установки вышек;
- 2) разработчик мобильных приложений. Цель: Интеграция данных OpenCellID в сторонние приложения;
- 3) обычный пользователь. Цель: Поиск оптимального сигнала, оценка покрытия;

сценарии использования (Use Cases):

– для технических специалистов (рисунок 2):

- 1) автоматический поиск вышек. Приложение собирает данные о вышках при перемещении;
- 2) анализ покрытия на карте. Отображение вышек с типом сети (2G/3G/4G/5G) и оператором;
- 3) сравнение исторических данных. Сохранение и сравнение данных за разные периоды;
- 4) экспорт данных в CSV. Для анализа в GIS-системах;

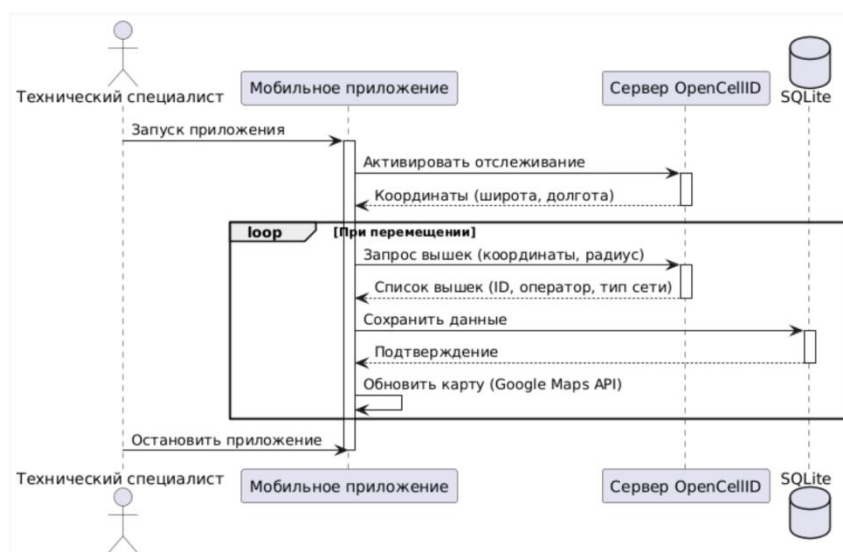


Рисунок 2 – Диаграмма последовательности. Технический специалист

- для разработчиков (рисунок 3):
- 1) тестирование геолокации через API. Проверка работы геолокации без GPS;
- 2) запрос технических параметров через REST API. Получение данных о частотах, мощности сигнала и операторах;
- 3) кэширование данных в SQLite. Для оффлайн-доступа;
- 4) настройка уведомлений о изменениях сети. Интеграция уведомлений в приложение;

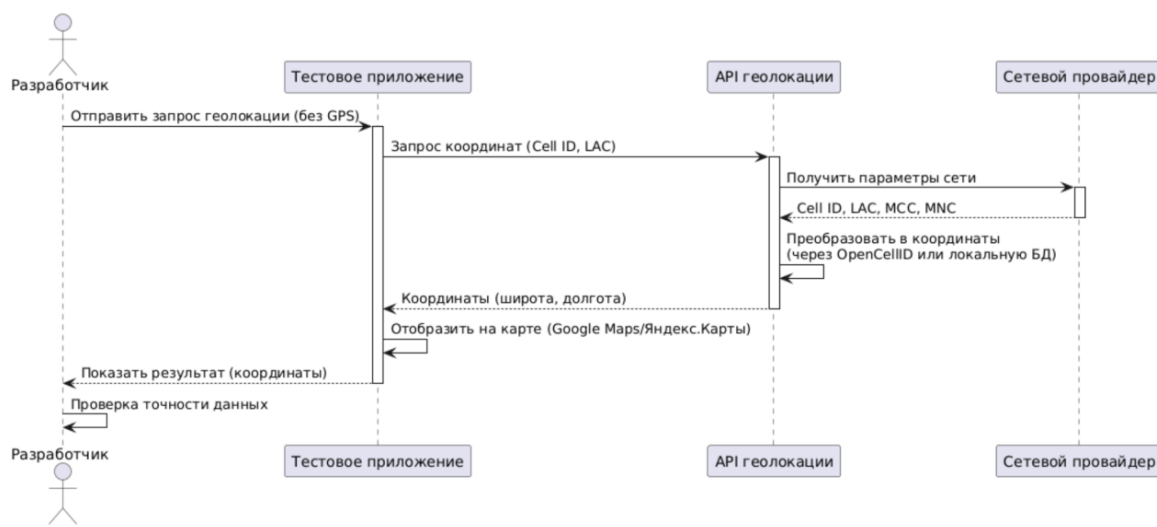


Рисунок 3 – Диаграмма вариантов использования. Разработчик

- для обычных пользователей (рисунок 4):
- 1) поиск ближайших вышек. Список станций с оператором и типом сети;
- 2) просмотр зоны покрытия на карте. Маркеры для визуализации сигнала;
- 3) сохранение истории запросов. Отслеживание изменений в сети;
- 4) автоматические настройки поиска. Радиус поиска и приоритет оператора;

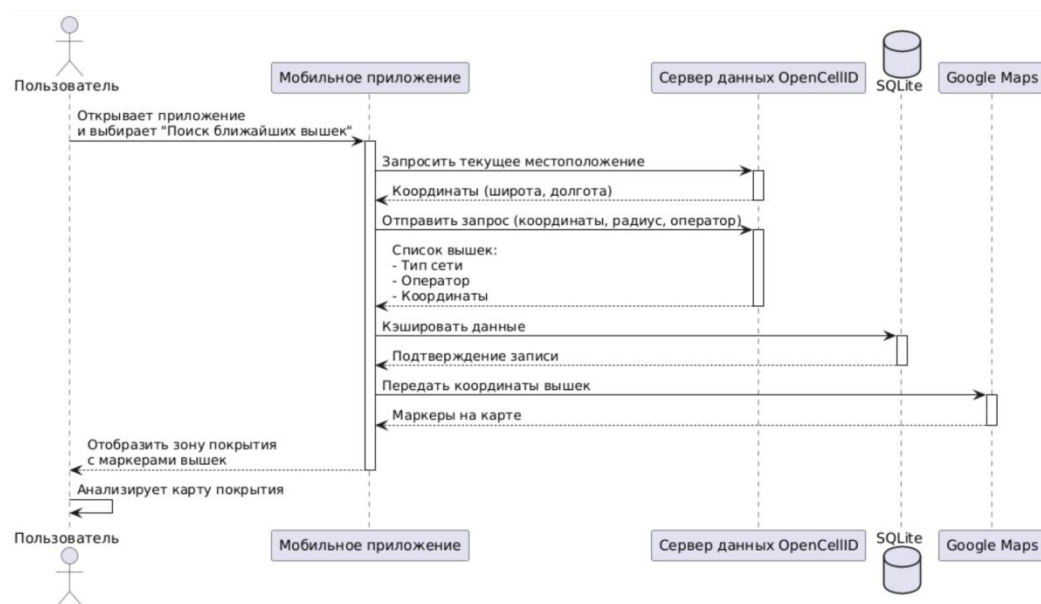


Рисунок 4 – Диаграмма последовательности. Пользователь

– общие сценарии (доступны всем акторам):

- 1) обработка данных в реальном времени. Обновление информации при изменении местоположения;
- 2) анализ скорости работы API. Сравнение времени отклика OpenCellID и кэшированных данных;
- 3) работа в оффлайн-режиме. Просмотр сохраненных данных без интернета;
- 4) интеграция с картографическими сервисами. Экспорт координат в Google Maps/Яндекс.Карты;

связи:

– технический специалист связан с:

- 1) автоматический поиск вышек;
- 2) анализ покрытия;
- 3) сравнение данных;
- 4) экспорт в CSV;

– общие:

- 1) обработка данных;
- 2) оффлайн-режим;

- 3) интеграция с картами;
- разработчик связан с:
 - 1) тестирование геолокации;
 - 2) запрос параметров;
 - 3) кэширование;
 - 4) уведомления;
- общие:
 - 1) анализ скорости API;
 - 2) оффлайн-режим;
- обычный пользователь связан с:
 - 1) поиск вышек;
 - 2) просмотр на карте;
 - 3) история запросов;
 - 4) автонастройки;
- общие:
 - 1) обработка данных;
 - 2) оффлайн-режим;
 - 3) интеграция с картами.

Исходя из выше реализованной аналитики, выделим основные группы пользователей и цели использования реализуемого приложения (таблица 1).

Таблица 1 – Особенности для разных групп пользователей

Группа	Особенности использования
Технические специалисты	Расширенная аналитика, экспорт сырых данных
Разработчики	Доступ к API, кастомизация логики запросов
Обычные пользователи	Упрощенный интерфейс, автоматические настройки

После определения вариантов использования приложения требуется определить стек технологий, которые позволят реализовать ПО. Для этого требуется провести сравнительный анализ имеющихся возможностей реализовать мобильное приложение, включая решения для работы с данными вышек сотовой связи.

					ПГУ1.090501.05.001 ПЗ	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

2 Техническое задание

2.1 Назначение разработки

2.1.1 Требования к функциональным характеристикам

Программа предназначена для поиска и отображения ближайших базовых станций (вышек сотовой связи) на основе текущего местоположения пользователя. Приложение помогает пользователям находить информацию о базовых станциях, таких как их координаты, оператор связи и тип сети (2G, 3G, 4G, 5G).

Требуется создать удобное и функциональное мобильное приложение для пользователей, инженеров и разработчиков, использующего SQLite для хранения данных о станциях и Google Maps API для визуализации их расположения.

Разрабатываемая программа должна выполнять следующие функции:

- главный экран (mainactivity) должен: предоставлять навигацию между режимами «пользователь», «инженер», «разработчик»;
- режим пользователя (useractivity) должен: отображать текущее местоположение пользователя. поиск ближайших базовых станций через opencellid api или локальную базу данных sqlite. передача данных о станциях на карту для визуализации;
- карта (mapactivity) должна: отображать маркеры базовых станций на карте. позволять пользователю взаимодействовать с картой (масштабирование, перемещение). отображать детальную информацию о станции при нажатии на маркер;
- модуль «детали станции» должен: отображать подробную информацию о выбранной базовой станции: id, координаты, оператор, тип сети. поддерживать редактирование данных о станции (для инженеров);
- настройки (settingsactivity) поддерживает включение/выключение автоматического поиска станций;

– база данных (sqlite) должна: хранение данных о базовых станциях (координаты, оператор, тип сети). хранение истории поиска (дата, время, координаты пользователя, найденные станции).

Входными данными являются данные о местоположении пользователя и информация о базовых станциях, получаемые через OpenCellID API или локальную базу данных. Выходными данными являются отображаемая информация о станциях, карта с маркерами и история поиска.

2.1.2 Требования к надежности

Программное средство должно обладать высокой степенью устойчивости к ошибкам и сбоям во время работы:

– обработка исключений: Программа должна корректно обрабатывать ошибки, такие как отсутствие доступа к интернету, проблемы с GPS или ошибки при работе с базой данных;

– валидация данных: Все поля ввода должны проверяться на корректность перед сохранением в базу данных;

– защита целостности данных: База данных SQLite должна использовать механизмы контроля целостности (например, уникальные ключи и триггеры);

– устойчивость к аварийному завершению: При возникновении нестандартной ситуации выполнение критических операций блокируется, а система возвращается в стабильное состояние без потери данных.

2.1.3 Требования к составу и параметрам технических средств

Программа должна быть предназначена для использования на мобильных устройствах под управлением Android. Минимальные требования к устройству:

– операционная система: Android 8.0 (API 26) и выше;

– процессор: совместимый с архитектурой ARM или x86;

– оперативная память: не менее 2 ГБ;

– объем свободного места на диске: не менее 50 МБ (для установки приложения и хранения базы данных);

					ПГУ1.090501.05.001 ПЗ	Лист
						22
Изм.	Лист	№ докум.	Подпись	Дата		

– доступ к интернету: для работы с OpenCellID API и Google Maps API;

– доступ к GPS: для определения текущего местоположения пользователя.

2.1.4 Требования к информационной и программной совместимости

Операционная система: Android 8.0 (API 26) и выше. СУБД: SQLite (встроенная в Android).

Используемое API: Google Maps API для работы с картами. OpenCellID API для получения данных о базовых станциях. Язык программирования: Kotlin. Архитектура: MVVM (Model-View-ViewModel).

2.1.5 Требования к программной документации

Программная документация к разрабатываемому приложению должна содержать следующие разделы:

– техническое задание: Полное описание требований к приложению;

– диаграмма вариантов использования: UML-диаграмма, показывающая взаимодействие пользователя с приложением;

– диаграмма классов: UML-диаграмма, демонстрирующая структуру классов приложения;

– описание логического проектирования БД: Схема таблиц и связей в базе данных;

– проектирование приложения: Описание архитектуры приложения (MVVM);

– описание применения: Инструкция по использованию приложения;

– текст программы: Исходный код приложения.

2.1.6 Порядок контроля и приемки

Для проверки исполнения программы и всех ее функций в соответствии с техническим заданием следует протестировать реакцию программы на возможные действия пользователя:

					ПГУ1.090501.05.001 ПЗ	Лист
						23
Изм.	Лист	№ докум.	Подпись	Дата		

- запуск приложения;
- определение текущего местоположения;
- поиск ближайших базовых станций;
- переход между активностями;
- работа с картой (масштабирование, перемещение, клики по маркерам);
- настройка параметров приложения.

Приложение соответствует требованиям технического задания, если оно выполняет все заданные функции при различных наборах входных данных из тестовых примеров.

Приемка программы должна осуществляться при выполнении всех ее функций, а также при наличии полной документации на программу (пояснительная записка с приложениями: текстом программы и результатами тестовых примеров).

					ПГУ1.090501.05.001 ПЗ	Лист
						24
Изм.	Лист	№ докум.	Подпись	Дата		

3 Анализ аналогов решений для работы с данными вышек сотовой связи

Подробный анализ существующих решений для работы с данными вышек сотовой связи позволит интегрировать в реализуемое приложение бесплатную и эффективную библиотеку.

Библиотека OpenSignal (рисунок 5) - инструмент для анализа качества мобильной связи, тестирования скорости интернета и визуализации зон покрытия операторов.

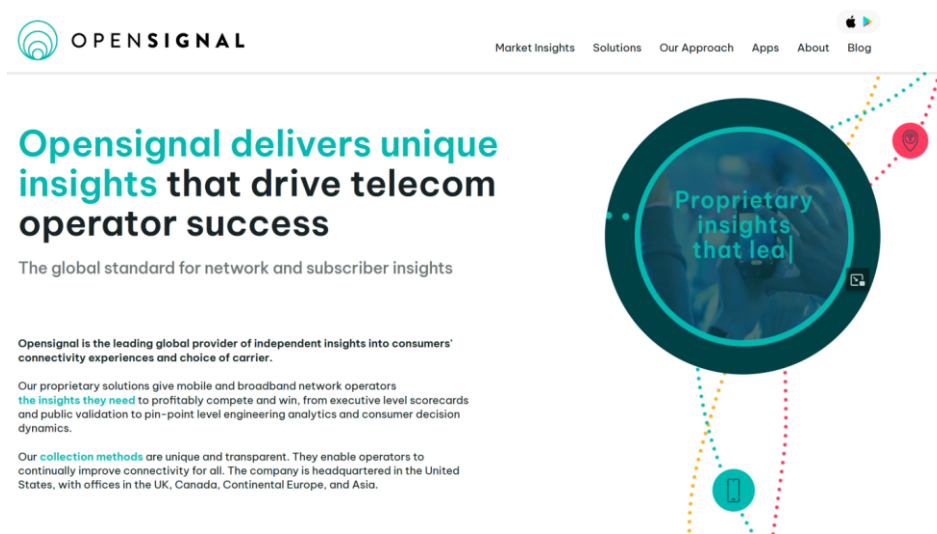


Рисунок 5 – Официальный сайт OpenSignal

Преимущества:

- реал-тайм карта покрытия с цветовой индикацией силы сигнала (2g/3g/4g/5g);
- тест скорости интернета с сохранением истории результатов;
- глобальная база данных (охват 190+ стран);
- интеграция с google maps для наложения данных;

недостатки:

- доступ к raw-данным (lac, cid, rssi) ограничен платной подпиской (от \$20/мес);
- невозможно экспортировать данные в сторонние приложения;
- закрытый api для разработчиков (только для партнеров);

					ПГУ1.090501.05.001 ПЗ	Лист
						25
Изм.	Лист	№ докум.	Подпись	Дата		

обновление данных:

- автоматически через измерения пользователей (фоновая работа приложения);

- ручное добавление информации от операторов связи;

интеграция:

- для некоммерческих проектов доступен ограниченный api (требуется запрос на opensignal api access);

- пример использования: получение средней скорости интернета по регионам.

Библиотека Network Cell Info Lite (рисунок 6) - инструмент для технического анализа сотовых сетей с отображением параметров вышек.

Преимущества:

- детальная информация:

- 1) LAC (Location Area Code), CID (Cell ID);

- 2) уровень сигнала (RSSI, RSRP, RSRQ);

- 3) частота и тип сети (LTE, NR, GSM);

- 4) карта с точным расположением вышек (для Android 7.0+);

- недостатки:

- 1) закрытый исходный код;

- 2) рекламные баннеры в бесплатной версии;

- 3) нет публичного api для интеграции;

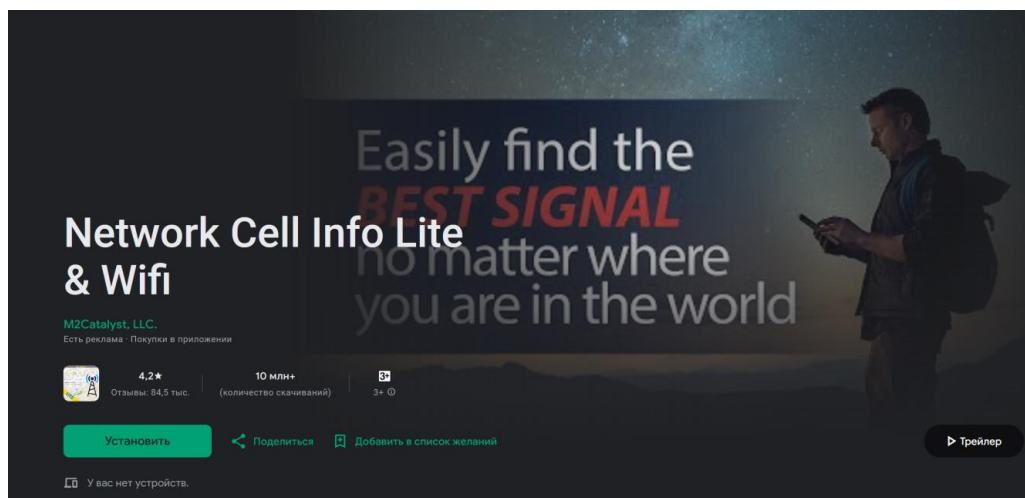


Рисунок 6 – Официальный сайт Network Cell Info Lite

					ПГУ1.090501.05.001 ПЗ	Лист
						26
Изм.	Лист	№ докум.	Подпись	Дата		

- обновление данных:
 - 1) через пользовательские устройства (только для Android);
 - 2) база данных обновляется раз в квартал;
- интеграция:
 - 1) невозможно напрямую подключиться к данным приложения;
 - 2) альтернатива: парсинг экрана через ADB (нарушает лицензионное соглашение).

Библиотека OpenCellID (рисунок 7) - открытая база данных с координатами вышек сотовой связи, создаваемая сообществом.

Преимущества:

- бесплатный доступ через REST API;
- 50 млн+ записей (охват 200+ стран);
- поддержка форматов JSON/CSV для экспорта;
- интеграция с OpenStreetMap, Google Maps, Leaflet;

недостатки:

- зависимость от пользовательских данных;
- в сельской местности данные могут быть устаревшими;
- нет автоматической верификации вводимой информации;
- отсутствие данных о уровне сигнала (только геолокация вышек).

Statistics

The following table shows the number of logical cells per radio standard and the total number of cells observed in the history of the project. The exports only include data observed in the last 18 months. Logical network numbers are only loosely correlated with physical infrastructure like cell towers.

Show entries Search:

Country Name	Total Cells	GSM	CDMA	UMTS	LTE	NR
United States of America	7,696,010	292,374	604,579	1,634,062	5,085,678	79,317
Germany	3,075,979	1,310,662	1	1,189,654	574,254	1,408
India	2,758,318	1,540,966	16	708,280	502,929	6,127
Japan	2,576,419	558	514	366,329	2,188,849	20,169
France	2,468,287	554,819	21	1,139,757	770,501	3,189
Russian Federation	2,232,968	588,214	8	1,180,976	463,761	9
Brazil	1,890,567	330,597	0	1,267,580	292,290	100
United Kingdom	1,771,990	334,873	0	992,943	441,917	2,257
Poland	1,551,246	470,860	0	756,294	324,074	18
Italy	1,511,331	260,273	0	844,362	406,676	20
Indonesia	1,434,773	328,519	971	875,340	229,940	3

Рисунок 7 – Официальный сайт OpenCellID

обновление данных:

- пользователи добавляют новые вышки через мобильное приложение или веб-интерфейс;

- данные обновляются ежедневно;

интеграция:

- полная совместимость с собственным приложением;
- для доступа требуется бесплатный API-ключ (регистрация на сайте).

Пример интеграции библиотеки в код JavaScript страницы:

```
// Пример запроса к API OpenCellID
fetch(`https://opencellid.org/cell/get?key=API_KEY&mcc=250&mnc=01&lac=123&cellid=456`)
  .then(response => response.json())
  .then(data => console.log(data.lat, data.lon));
```

Таблица 2 – Сравнение возможностей интеграции

Параметр	OpenSignal	Network Cell Info Lite	OpenCellID
Тип лицензии	Проприетарная	Проприетарная	Открытая
API для разработки	Ограниченный доступ	Отсутствует	Полный доступ
Стоимость	Платный (от \$20/мес.)	Бесплатно с рекламой	Бесплатно
Данные для экспорта	Только агрегированные	Нет	Координаты вышек
Пример использования	Анализ скорости интернета	Техническая диагностика	Визуализация на карте

Выводы для интеграции в собственное приложение:

- OpenSignal подходит для коммерческих проектов с бюджетом, но требует согласования с правообладателем;
 - Network Cell Info Lite не рекомендуется из-за закрытости и юридических ограничений;
 - OpenCellID является оптимальным выбором для некоммерческих и образовательных проектов. Позволяет свободно использовать данные, модифицировать код и добавлять новые функции (например, фильтрацию по операторам);
- также имеет смысл для мобильного приложения на базе OpenCellID целесообразно:
- реализовать синхронизацию с API для получения актуальных данных;
 - добавить модуль crowdsourcing, чтобы пользователи могли вносить новые вышки;

– интегрировать картографический SDK (Mapbox или Google Maps) для визуализации.

OpenCellID выбран как наиболее полная открытая база с REST API для получения данных в формате JSON. Для кроссплатформенной разработки используется React Native, что позволяет сократить время выпуска версий под iOS и Android.

					ПГУ1.090501.05.001 ПЗ	Лист
						30
Изм.	Лист	№ докум.	Подпись	Дата		

4. Анализ стека технологий для разработки мобильного приложения

Анализ стека технологий включает выбор операционной системы, языка программирования и средств разработки приложения.

4.1 Сравнение мобильных ОС

При выборе платформы для разработки приложения Cell Tower Locator ключевым фактором является охват аудитории и технические возможности операционных систем. Рассмотрим две доминирующие на рынке ОС — Android и iOS [4].

Android, разработанная Google, занимает лидирующие позиции на глобальном рынке с долей около 70% (StatCounter, 2024). Ее распространенность объясняется следующими преимуществами:

- открытая экосистема: разработчики имеют доступ к исходному коду ОС, что позволяет глубоко кастомизировать приложения и интегрировать их с аппаратными компонентами устройств;
- поддержка сторонних магазинов приложений (например, Huawei AppGallery) расширяет возможности дистрибуции;
- гибкость: широкая совместимость с устройствами разных ценовых категорий (от бюджетных моделей до флагманов);
- возможность адаптации интерфейса под специфические требования пользователей;
- масштабируемость: интеграция с сервисами Google (Maps, Firebase) упрощает реализацию функций геолокации и аналитики.

Однако у Android есть существенные недостатки. Ниже перечислены основные из них.

Фрагментация версий и устройств. Одновременно на рынке присутствуют устройства с разными версиями ОС (от Android 8 до Android 14), что усложняет тестирование и оптимизацию приложения.

Проблемы безопасности. Открытость платформы повышает риски уязвимостей, особенно на устройствах с устаревшими версиями ОС.

Операционная система iOS, эксклюзивная ОС для устройств Apple, занимает около 28% рынка, доминируя в регионах с высокой платежеспособностью пользователей (США, Западная Европа). Ее ключевые преимущества:

- стабильность и безопасность: единая экосистема Apple обеспечивает строгий контроль качества приложений;
- регулярные обновления ОС и sandbox-архитектура минимизируют угрозы утечек данных;
- единообразие устройств: ограниченный модельный ряд iPhone и iPad упрощает тестирование и оптимизацию;
- высокая производительность даже на устройствах предыдущих поколений;
- монетизация: пользователи iOS чаще готовы платить за приложения и подписки;

недостатки iOS:

- закрытая экосистема. Публикация приложений возможна только через App Store, а процесс модерации занимает до 7-14 дней;
- ограниченная кастомизация. Разработчики не имеют доступа к низкоуровневым функциям системы;

несмотря на преимущества iOS, для приложения Cell Tower Locator целевой платформой выбрана Android по следующим причинам:

- максимальный охват аудитории;
- ОС Android доминирует в РФ и развивающихся странах (Индия, Бразилия, страны Африки), где потребность в анализе покрытия сотовой связи наиболее высока;
- гибкость интеграции с оборудованием: доступ к API телефона для сбора данных о сигнале (уровень RSSI, идентификаторы вышек) проще реализовать на Android;

					ПГУ1.090501.05.001 ПЗ	Лист
						32
Изм.	Лист	№ докум.	Подпись	Дата		

– снижение затрат на тестирование: использование кроссплатформенных инструментов позволяет в будущем портировать приложение на iOS с минимальными изменениями кода.

Таким образом, разработка под ОС Android [5] обеспечивает баланс между охватом пользователей, функциональностью и экономической эффективностью разработки.

4.2 Сравнение средств разработки

В данном разделе проводится оценка инструментов и технологий, необходимых для реализации ключевых функций приложения: работы с картами, оффлайн-режима и интеграции API [6]. Рассматриваются языки программирования и их совместимость с фреймворками, системами управления базами данных (СУБД) и средами разработки (IDE).

Для разработки мобильных приложений наиболее актуальны следующие языки программирования (таблица 3).

Таблица 3 – Сравнение технологий разработки

Критерий	Kotlin (Android)	Swift (iOS)	Java
Тип разработки	Нативная	Нативная	Нативная
Производительность	Высокая (прямой доступ к железу)	Высокая (оптимизация под Apple Silicon)	Средняя (устаревшая виртуальная машина)
Интеграция с картами	Google Maps SDK, Mapbox	MapKit, Google Maps API	Google Maps SDK
Оффлайн-режим	SQLite, Room	CoreData, SQLite.swift	SQLite, RealmDB
Работа с API	Retrofit, OkHttp	URLSession, Alamofire	Retrofit, Volley
Кроссплатформе	Только Android	Только iOS	Только Android

нность			
Сообщество	Активное (официальный язык Android)	Активное (экосистема Apple)	Устаревающее
Сложность обучения	Средняя (совместим с Java)	Высокая (требуется знания SwiftUI)	Низкая (документация)

В качестве базы для разработки выбран язык программирования Kotlin для обеспечения максимальной производительности для нативной разработки под Android. Язык Java уступает Kotlin [6] в удобстве и современных возможностях (например, корутины), несмотря на кроссплатформенность.

4.3 Сравнение СУБД

Для хранения данных в оффлайн-режиме рассматриваются следующие решения (таблица 4).

Таблица 4 – Сравнение СУБД

СУБД	Тип	Плюсы	Минусы	Интеграция
SQLite	Реляционная	Легковесная, встроенная в ОС	Нет поддержки NoSQL	Нативно в Android/iOS, sqlite в Flutter
RealmDB	Объектно- ориентирован ная	Высокая скорость, синхронизация с облаком	Большой размер приложения	Требуется нативных библиотек
Firebase	Cloud-NoSQL	Автосинхронизация, аутентификация	Требуется интернет- соединения	Облачная зависимость

Для реализации приложения оптимальный выбор SQLite [7]:

– поддерживается всеми платформами;

					ПГУ1.090501.05.001 ПЗ	Лист
						34
Изм.	Лист	№ докум.	Подпись	Дата		

- минимальные требования к ресурсам (важно для старых устройств);
- совместима с GIS-системами для экспорта данных в CSV;
- СУБД RealmDB и Firebase избыточны из-за зависимости от облака.

4.4 Сравнение сред разработки (IDE)

Выбор среды разработки (IDE) играет ключевую роль в эффективности процесса создания приложения. В данном разделе проводится оценка популярных IDE с учетом их совместимости с выбранными технологиями (таблица 5), функциональности и требований проекта Cell Tower Locator.

Для анализа использовались следующие критерии:

- поддержка технологий — совместимость с языками и фреймворками (Kotlin, Swift, Flutter, СУБД, плагинов);
- производительность — скорость работы, потребление ресурсов;
- интеграция с инструментами — наличие встроенных эмуляторов, отладчиков, профилировщиков;
- гибкость настройки — возможность адаптации под требования проекта;
- доступность — поддержка операционных систем, стоимость лицензии.

Таблица 5 – Сравнение IDE

Инструмент	Поддерживаемые технологии	Плюсы	Минусы
Android Studio	Kotlin, Java, Flutter	Глубокая интеграция с Android SDK, эмуляторы	Высокое потребление ресурсов
Visual Studio Code	Flutter, Dart, JavaScript	Легковесный, плагины для Git,	Требует ручной настройки

		REST-клиент	
IntelliJ IDEA	Kotlin, Java	Умный анализ кода, рефакторинг	Платная версия для полного функционала

Учитывая выбранный стек технологий (Kotlin, SQLite, интеграция с OpenCellID), оптимальным инструментом является Android Studio для разработки под Android, тестирования на эмуляторах и профилирования. Android Studio обеспечивает глубокую интеграцию с Google Maps SDK [8] и SQLite, что критично для функций картографии и оффлайн-режима.

В результате аналитики и исследования предметной области, потенциальных пользователей и стека технологий формируются следующие промежуточные выводы. Цель реализации дипломного проекта является создание альтернативы коммерческим решениям с акцентом на:

- открытость данных;
- визуализацию вышек на карте с фильтрами (2g/3g/4g/5g);
- возможность анализа доступности сервиса opencellid [9, 10];

целевая аудитория для использования ПО разделена на две части:

- основная:
 - 1) пользователи смартфонов, испытывающие проблемы с связью;
- профессиональная:
 - 1) инженеры связи;
 - 2) урбанисты;
 - 3) разработчики IoT-устройств.

5 Разработка мобильного приложения Cell Tower Locator

На основании исследования, проведенного в первой главе дипломной работы составлена диаграмма разворачивания (рисунок 8).

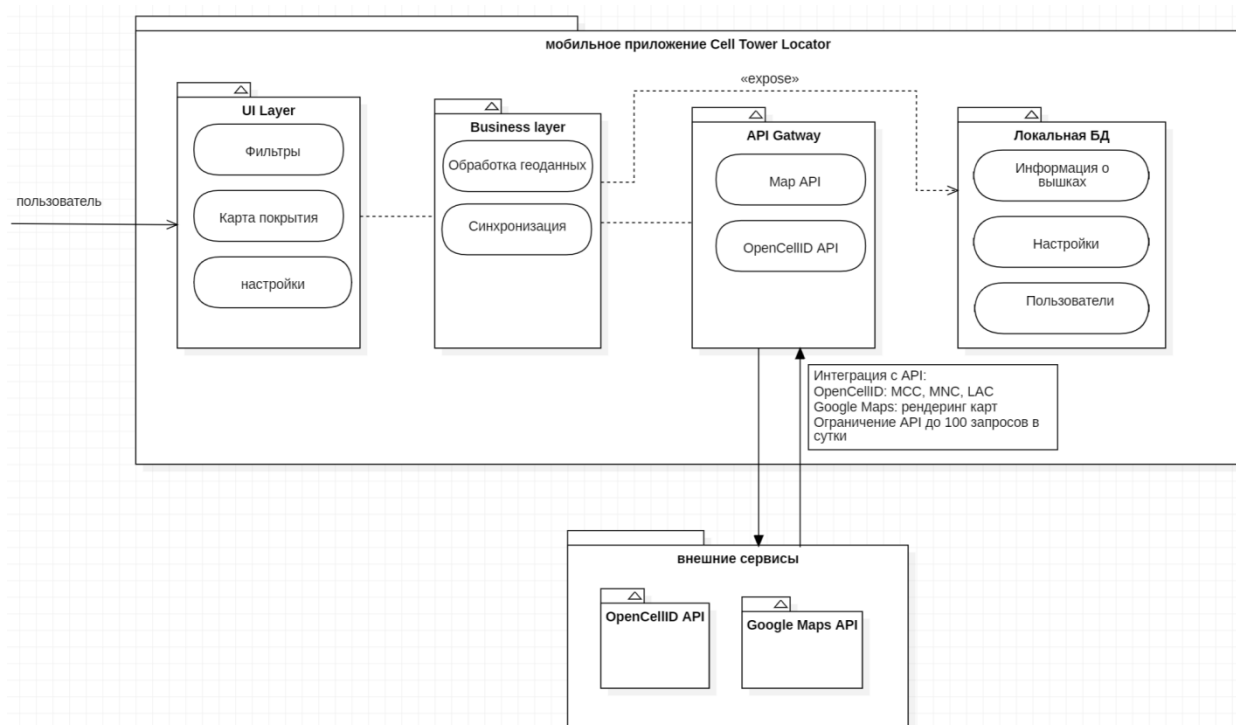


Рисунок 8 – Диаграмма разворачивания

На основе предоставленной диаграммы разворачивания, можно составить структурированное описание системы, которое включает все ключевые элементы программного приложения, их взаимодействие и особенности реализации.

Система состоит из нескольких логически организованных слоев, каждый из которых выполняет определенные задачи.

Первым слоем системы является мобильное приложение:

- пользовательский интерфейс (UI Layer). Этот слой предоставляет интуитивно понятный интерфейс для взаимодействия с системой. Он адаптирован для трех основных ролей пользователей. Конечный

пользователь: может просматривать карту покрытия, фильтровать данные по операторам связи (2G/3G/4G/5G) и анализировать доступные базовые станции;

– технический специалист:

- 1) имеет возможность добавлять пользовательские метки;
- 2) анализировать параметры вышек (например, LAC, CID, RSSI);
- 3) управлять настройками;

– разработчик:

1) получает доступ к режиму отладки и логированию данных для тестирования и анализа работы приложения;

– основные экраны включают:

- 1) карту покрытия;
- 2) фильтры;
- 3) настройки;

интерфейс разработан с учетом требований удобства и производительности;

Вторым слоем системы является бизнес-логика (Business Logic), состоящая функционала обработки данных и управление их потоками;

ключевые функции включают:

- 1) обработка геоданных;
- 2) нормализация координат;
- 3) преобразование форматов;
- 4) расчет расстояний до базовых станций;

– кэширование данных:

1) минимизация нагрузки на внешние API за счет сохранения кэшированных данных на локальном устройстве;

– синхронизация данных:

1) обеспечение актуальности информации путем регулярного обмена данными между локальной базой данных и облачными сервисами.

					ПГУ1.090501.05.001 ПЗ	Лист
						38
Изм.	Лист	№ докум.	Подпись	Дата		

Третьим слоем системы является API Gateway (Шлюз API). Этот компонент управляет взаимодействием с внешними сервисами:

- OpenCellID Client: отправляет REST-запросы для получения данных о базовых станциях (MCC/MNC/LAC);
- Google Maps SDK: обеспечивает визуализацию карт, геокодирование и расчет расстояний.

Шлюз API абстрагирует сложность взаимодействия с внешними сервисами, предоставляя унифицированный интерфейс для бизнес-логики.

Четвертым слоем системы является Локальная БД (SQLite). Локальная база данных играет важную роль в обеспечении автономной работы приложения. Она хранит следующие типы данных:

- кэшированные данные OpenCellID: (информация о базовых станциях (координаты, MCC/MNC));
- пользовательские метки и комментарии: (отметки, добавленные техническими специалистами или пользователями);
- персонализированные настройки: (такие как тема приложения, частота обновления данных и предпочтения фильтрации).

Благодаря локальному хранилищу, приложение может работать без подключения к интернету, используя ранее загруженные данные.

Внешние сервисы играют ключевую роль в обеспечении функциональности приложения:

- OpenCellID API. Это основной источник данных о базовых станциях. Сервис содержит более 55 миллионов записей и предоставляет информацию о координатах, операторах связи и типах сетей. На бесплатном тарифе доступно до 100 запросов в день, что делает кэширование данных особенно важным для минимизации нагрузки на API;
- Google Maps API. Используется для отображения карт, геокодирования и расчета расстояний. Google Maps SDK предоставляет мощные инструменты для визуализации данных, такие как маркеры, слои и интерактивные элементы;

ключевые особенности архитектуры:

- автономный режим. Локальная база данных SQLite позволяет приложению работать без подключения к интернету. Все необходимые данные кэшируются на устройстве и могут быть использованы повторно в течение 24 часов;
- кэширование данных. Для минимизации количества запросов к OpenCellID API данные сохраняются локально. Это не только снижает нагрузку на внешний сервис, но и улучшает производительность приложения;
- ролевая модель. Разделение функционала для разных типов пользователей (конечные пользователи, технические специалисты, разработчики) повышает удобство использования и позволяет каждому пользователю выполнять свои задачи максимально эффективно;
- гибкость и масштабируемость. Архитектура приложения обеспечивает легкость добавления новых функций и интеграции с другими сервисами. Например, можно легко добавить поддержку дополнительных API или расширить возможности бизнес-логики;
- архитектура мобильного приложения Cell Tower Locator обеспечивает надежное и эффективное взаимодействие всех компонентов системы.

5.1 Проектирование приложения Cell Tower Locator

Наличие диаграммы развертывания (рисунок 8) и диаграммы вариантов использования (рисунок 1) позволяет реализовать диаграммы (рисунок 9) последовательности с точки зрения реализации приложения.

					ПГУ1.090501.05.001 ПЗ	Лист
						40
Изм.	Лист	№ докум.	Подпись	Дата		

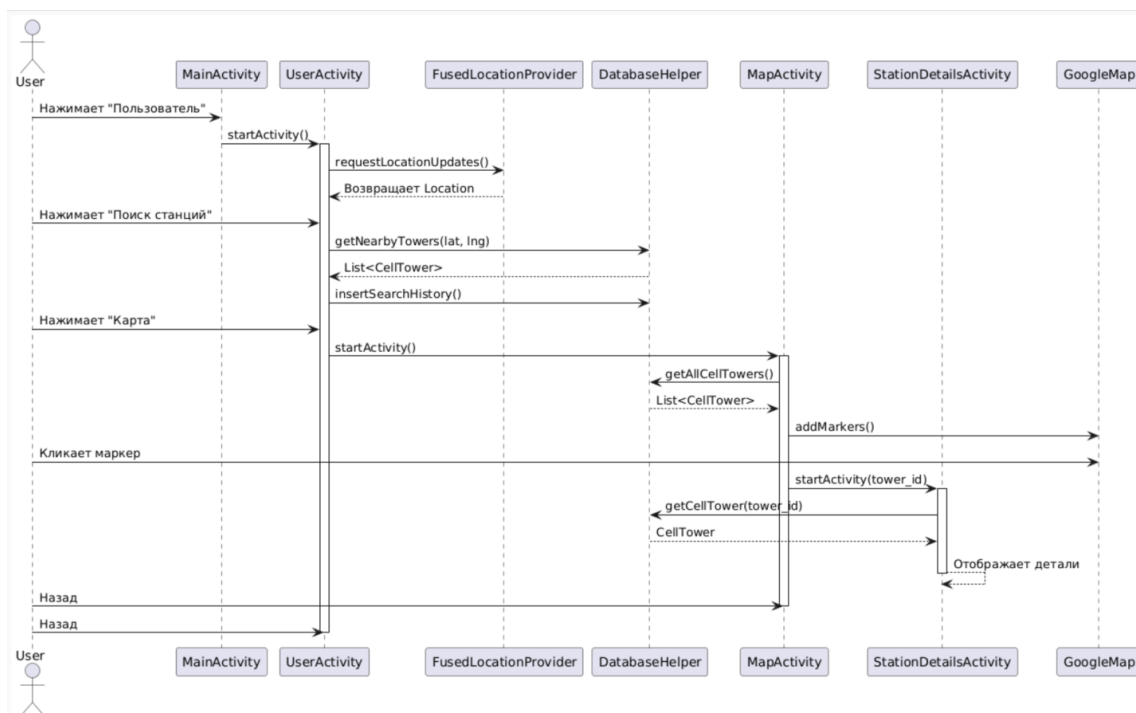


Рисунок 9 – Диаграмма последовательности. Пользователь

Пользовательский режим приложения Cell Tower Locator ориентирован на решение задач поиска и визуализации базовых станций. Основные функции включают:

- определение текущего местоположения пользователя;
- поиск ближайших базовых станций в радиусе 2 км;
- отображение станций на интерактивной карте;
- просмотр детальной информации о выбранной станции;
- сохранение истории поиска для последующего анализа.

На рисунке 10 представлена UML-диаграмма последовательности для ключевых сценариев работы в пользовательском режиме.

Основные этапы:

- инициализация пользовательского интерфейса:
 - 1) пользователь нажимает кнопку «Пользователь» в MainActivity;
 - 2) запускается UserActivity через метод startActivity();
- получение геолокации:
 - 1) useractivity запрашивает местоположение через fusedlocationproviderclient;

2) система возвращает координаты (широта/долгота) с точностью до 10 метров;

– поиск ближайших станций:

1) при нажатии кнопки «поиск станций»;

2) `useractivity` вызывает метод `getnearbytowers(lat, lng)` класса `databasehelper`;

3) выполняется sql-запрос с использованием формулы гаверсинусов для расчета расстояний;

4) результат (список объектов `CellTower`) возвращается и сохраняется в истории поиска через `insertSearchHistory()`;

– работа с картой:

1) при переходе в `mapactivity`;

2) загружаются все станции из БД через `getallcelltowers()`;

3) маркеры с иконками (4g/5g) добавляются на карту `google maps`;

– просмотр деталей станции:

1) при клике на маркер;

2) запускается `stationdetailsactivity` с передачей идентификатора `tower_id`;

3) детальная информация (оператор, тип сети) запрашивается из БД и отображается в виде карточки.

Режим инженера предназначен для управления данными о базовых станциях и обеспечивает:

– добавление новых станций вручную:

1) валидацию вводимых параметров (координаты, оператор, тип сети);

2) интеграцию с основной базой данных приложения. (доступ к режиму защищён авторизацией и доступен только пользователям с ролью «Инженер»).

На рисунке 10 представлена UML-диаграмма последовательности для сценария добавления базовой станции.

					ПГУ1.090501.05.001 ПЗ	Лист
						42
Изм.	Лист	№ докум.	Подпись	Дата		

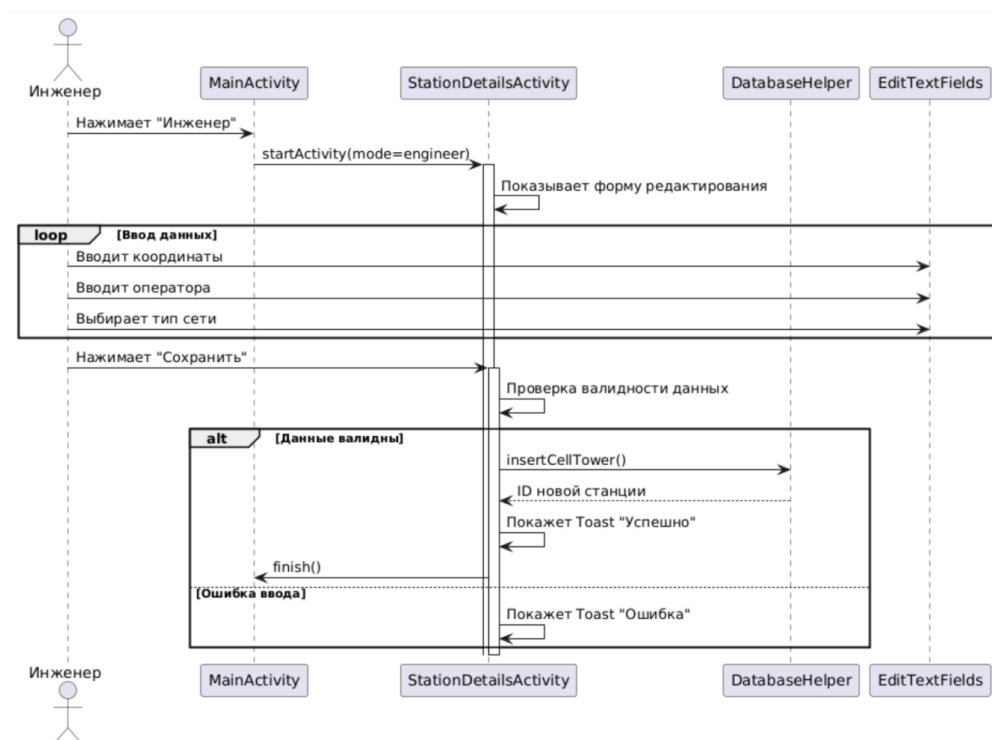


Рисунок 10 – Диаграмма последовательности. Инженер

основные этапы:

- активация режима редактирования:
 - 1) инженер выбирает кнопку «инженер» в MainActivity;
 - 2) система запускает stationdetailsactivity с параметром mode=engineer;
- отображение формы ввода:
 - 1) stationdetailsactivity отображает интерфейс с полями;
 - 2) координаты (широта/долгота);
 - 3) название оператора;
 - 4) выпадающий список типов сети (3g/4g/5g);
- процесс ввода данных:
 - 1) инженер последовательно заполняет обязательные поля;
 - 2) координаты в формате decimal degrees (например: 55.7558, 37.6173);
 - 3) оператор связи (минимум 3 символа);
 - 4) тип сети из predetermined списка;
- сохранение данных:

- 1) при нажатии «сохранить» выполняется;
- 2) проверка корректности координат (диапазон -90...+90 для широты, -180...+180 для долготы);
- 3) валидация ввода данных (допустимые символы: a-z, 0-9, пробел);
- 4) при успешной проверке;
- 5) вызов метода *insertcelltower()* в *databasehelper*;
- 6) показ toast-уведомления «станция добавлена»;
- 7) при ошибке;
- 8) подсветка невалидных полей;
- 9) toast с описанием проблемы.

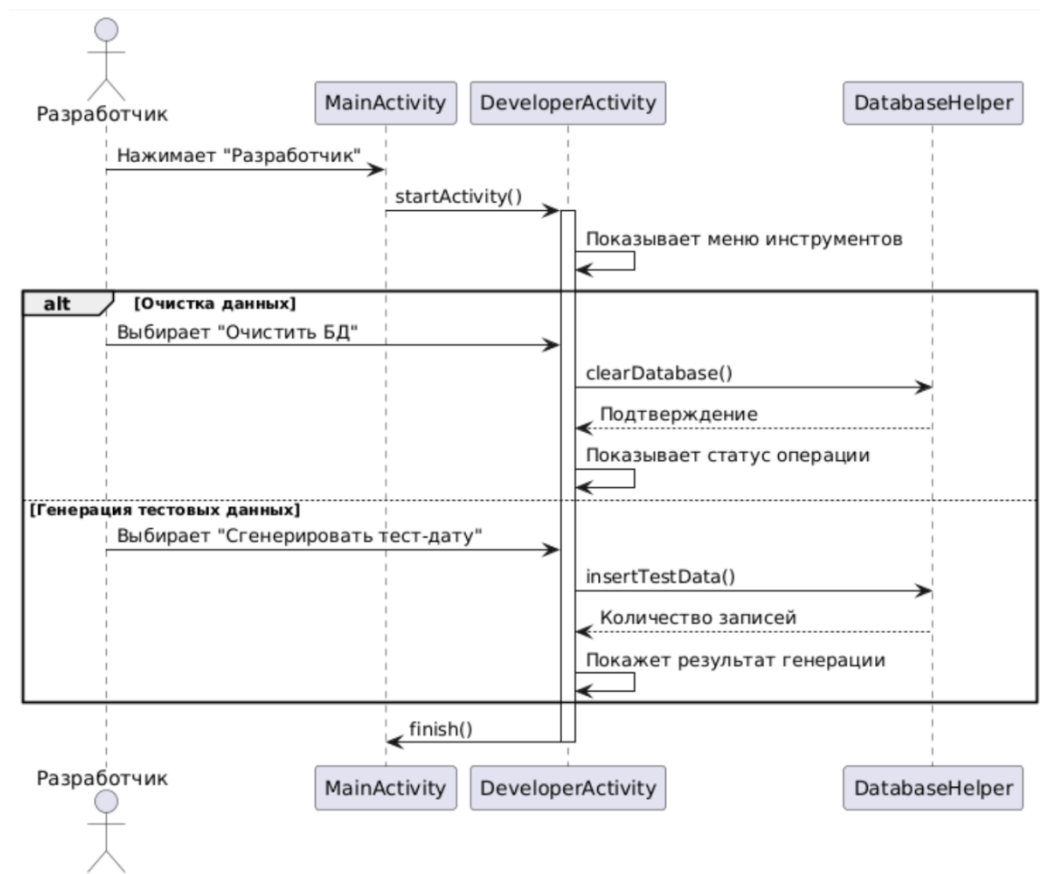


Рисунок 11 – Диаграмма последовательности. Разработчик

Режим разработчика (рисунок 11) в приложении Cell Tower Locator предоставляет инструменты для тестирования, отладки и управления данными в условиях разработки. Он включает функционал для:

- очистки всей базы данных (БД) от тестовых или устаревших записей;
- генерации тестовых данных для проверки работы функций приложения, связанных со связью с OpenCellID.

5.2 Проектирование базы данных приложения Cell Tower Locator

База данных приложения Cell Tower Locator (рисунок 12) реализована на SQLite и состоит из четырех нормализованных таблиц, связанных через внешние ключи. Структура таблиц обеспечивает:

- хранение информации о пользователях и их ролях;
- управление данными о базовых станциях;
- фиксацию истории поиска;
- персонализированные настройки.

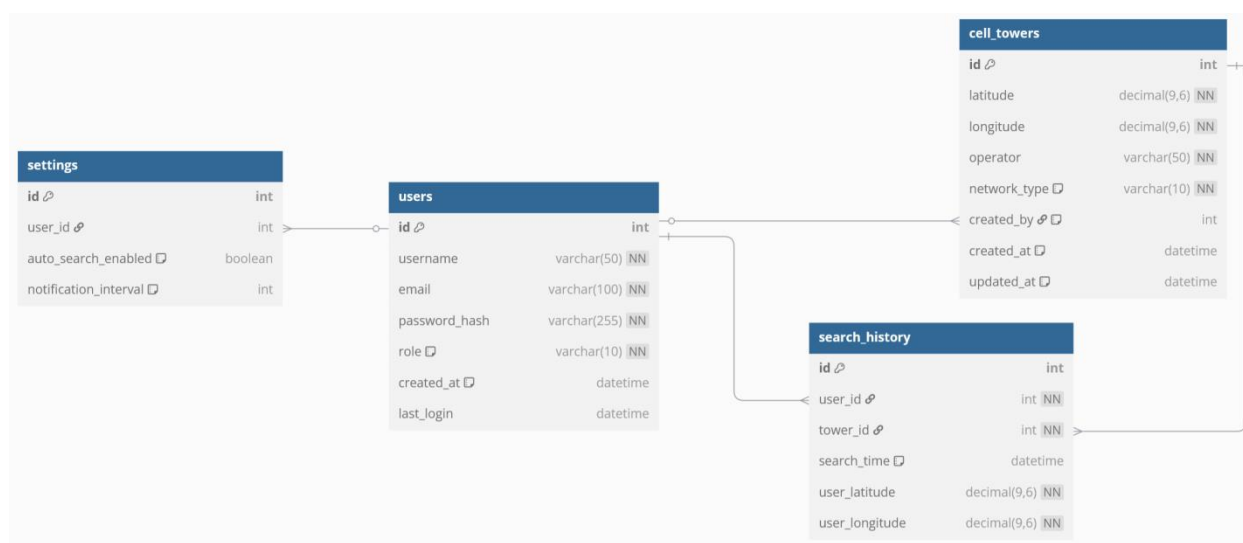


Рисунок 12 – Схема базы данных ПО Cell Tower Locator

Таблица 6 – Таблица users (пользователи)

Поле	Тип данных	Ограничения	Описание
id	INTEGER	PRIMARY KEY,	Уникальный

		AUTOINCREMENT	идентификатор
username	VARCHAR (50)	NOT NULL, UNIQUE	Логин пользователя
email	VARCHAR (100)	NOT NULL, UNIQUE	Электронная почта
password_hash	VARCHAR (255)	NOT NULL	Хэш пароля
role	VARCHAR (10)	NOT NULL	Роль: user/engineer/developer
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Дата регистрации
last_login	DATETIME	—	Время последнего входа

назначение:

- авторизация и аутентификация пользователей;
- разграничение прав доступа (ролевая модель);

Таблица 7 – Таблица cell_towers (базовые станции)

Поле	Тип данных	Ограничения	Описание
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Уникальный идентификатор
latitude	DECIMAL(9,6)	NOT NULL	Широта (-90.0...90.0)
longitude	DECIMAL(9,6)	NOT NULL	Долгота (- 180.0...180.0)
operator	VARCHAR(50)	NOT NULL	Оператор связи

network_type	VARCHAR(10)	NOT NULL	Тип сети (2G или 3G или 4G или 5G)
created_by	INTEGER	FOREIGN KEY (users.id)	Автор записи (инженер/разработчик)
created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Дата создания
updated_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	Дата изменения

Назначение:

- Хранение геоданных и технических параметров базовых станций;
- Аудит изменений (поле в программном обеспечении updated_at).

Таблица 8 – Таблица search_history (история поиска)

Поле	Тип данных	Ограничения	Описание
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Уникальный идентификатор
user_id	INTEGER	FOREIGN KEY (users.id), NOT NULL	Связь с пользователем
tower_id	INTEGER	FOREIGN KEY (cell_towers.id), NOT NULL	Связь со станцией
search_time	DATETIME	DEFAULT CURRENT_TIMESTAMP	Время выполнения поиска
user_latitude	DECIMAL(9,6)	NOT NULL	Широта пользователя

			при поиске
user_longitude	DECIMAL(9,6)	NOT NULL	Долгота пользователя при поиске

назначение:

- фиксация геоконтекста поисковых операций;
- анализ пользовательской активности;

Таблица 9 – Таблица settings (настройки)

Поле	Тип данных	Ограничения	Описание
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Уникальный идентификатор
user_id	INTEGER	FOREIGN KEY (users.id), UNIQUE	Связь с пользователем
auto_search_enabled	BOOLEAN	DEFAULT TRUE	Автоматический поиск
notification_interval	INTEGER	DEFAULT 15	Интервал уведомлений (минуты)

назначение:

- персонализация поведения приложения;
- управление push-уведомлениями (отправка сообщений сервису OpenCellID, до 100 шт. в сутки).

Таблица 10 – Связи между таблицами

Таблица	Поле	Целевая таблица	Тип связи	Описание
cell_towers	created_by	users	Многие-к-одному	Станцию добавляет пользователь
search_history	user_id	users	Многие-к-	Поиск выполняет

			одному	пользователь
search_history	tower_id	cell_towers	Многие-к-одному	Результат поиска — станция
settings	user_id	users	Один-к-одному	Уникальные настройки пользователя

База данных приложения соответствует третьей нормальной форме (3NF). Это определяется следующими критериями:

- первая нормальная форма (1NF):
 - 1) соблюдено правило;
 - 2) все таблицы имеют значения без повторяющихся групп;
- пример:
 - 1) поле network_type содержит одно значение (например, "4g"), а не список;

- 2) координаты разделены на latitude и longitude;

- вторая нормальная форма (2NF);

- 1) соблюдено все;
- 2) неключевые поля полностью зависят от первичного ключа;

пример:

- 1) в таблице cell_towers поля operator, network_type зависят от id, а не от части ключа;

- 2) в таблице search_history поля user_latitude/user_longitude зависят от id, а не от user_id или tower_id;

третья нормальная форма (3NF):

- 1) соблюдено;
- 2) отсутствуют транзитивные зависимости между неключевыми полями;

пример:

- 1) в таблице users поле role не зависит от email (оба зависят от id);

2) в таблице *settings* поле *notification_interval* не связано с *auto_search_enabled*.

5.3 Прототипирование интерфейса приложения Cell Tower Locator

Интерфейс приложения Cell Tower Locator реализован с соблюдением единства стиля всех диалогов активностей.

Интерфейс приложения реализован на основе навигации с явным разделением функционала для ролей пользователей (пользователь, инженер, разработчик). Все экраны связаны через систему навигации. Опишем прототип интерфейса (рисунок 13) на основании которого реализовано приложение.

Главный экран (MainActivity) является точкой входа в приложение. MainActivity предлагает выбор режима работы (пользователь, инженер, разработчик).

Элементы интерфейса экрана MainActivity:

- Кнопка «Пользователь»: переход на UserActivity;
- Кнопка «Инженер»: запуск экрана StationDetailsActivity в режиме редактирования карты и информации о вышках сотовой связи;
- Кнопка «Разработчик»: Переход на экран DeveloperActivity для работы с базой данных и соединением с онлайн ресурсом OPENCCELLID.

Экран пользователя (UserActivity) отображает текущее местоположение. Предназначен для поиска ближайших базовых станций, навигации по карте.

Элементы интерфейса.

Карта на основе Google Maps с маркером текущей позиции или аналог.

При изменении локации происходит обновление индикатора координат. Формат индикатора координат. Широта: 55.7558. Долгота: 37.6173. Возможен запрос к SQLite в офлайн-режиме.

					ПГУ1.090501.05.001 ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

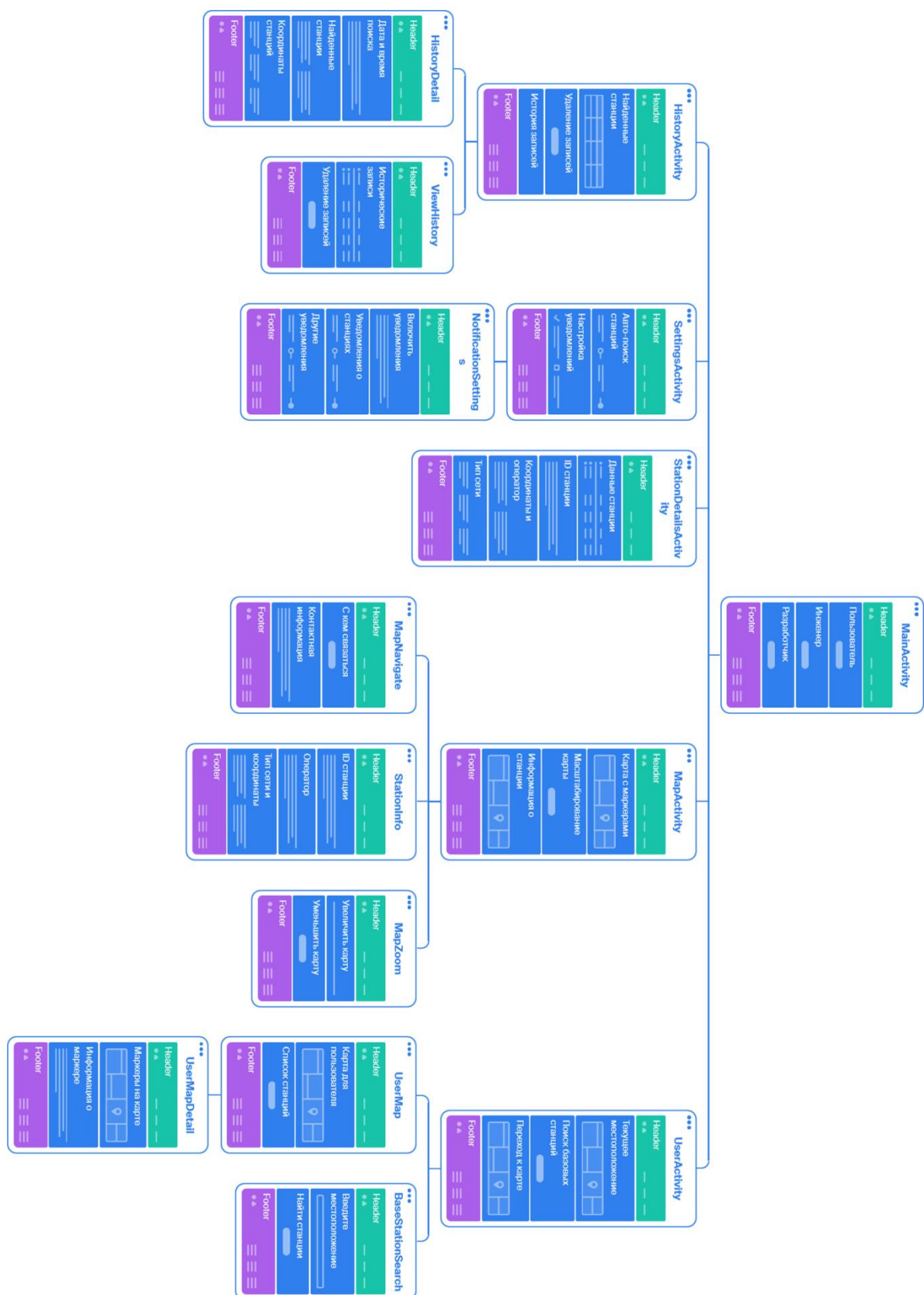


Рисунок 13 – Структура мобильного приложения Cell Tower Locator

Экран карты (MapActivity) предназначен для визуализации базовых станций на карте и интерактивного взаимодействия с объектами.

Элементы интерфейса карты Google Maps, кастомные маркеры с иконками (2G/3G/4G/5G). Предусмотреть возможность отображения информации об операторе и типе сети. имеется переход на активность StationDetailsActivity для получения подробной информации об объектах.

Экран деталей информации о станции (StationDetailsActivity) необходим для отображения технических параметров выбранной станции.

Элементы интерфейса:

- карточка станции с полями;
- ID: TextView с моноширинным шрифтом;
- координаты: Ссылка для открытия в Google Maps;
- оператор: Логотип оператора (загружается через элемент Glide);
- тип сети: Badge с цветовой индикацией (например 4G — синий, 5G — зеленый);

кнопки навигации:

- экран настроек (SettingsActivity) предназначен для управления параметрами приложения;

элементы интерфейса:

- автоматический поиск базовых станций в онлайн режиме;
- интервал уведомления;
- синхронизация данных в SQLite.

Экран истории (HistoryActivity) предназначен для просмотра и управления историей поиска.

Элементы интерфейса:

- просмотрщик структурированных записей;
- разделители между элементами.

Элемент списка: дата поиска: 12.05.2025 14:30., количество найденных станций.

Интерфейс приложения должен обеспечивать: разделение ролей, интуитивную навигацию, доступ к ключевым функциям, интеграцию с сервисами (карты и сервисы OPENCCELLID).

5.4 Программная реализация приложения Cell Tower Locator

Диаграмма классов (рисунок 14) показывает структуру и функциональность реализованного приложения Cell Tower Locator.

Приложение реализовано в архитектуре MVVM (Model-View-ViewModel), что обеспечивает разделение логики приложения и пользовательского интерфейса, упрощение тестирования кода.

В реализованном приложении имеются следующие слои:

– слой Model представлен классами:

- 1) CellTower;
- 2) HistoryEntry;
- 3) DatabaseHelper — отвечают за хранение и управление данными.

– Слой View представлен классами:

- 1) MainActivity;
- 2) UserActivity;
- 3) MapActivity;
- 4) StationDetailsActivity;
- 5) SettingsActivity;
- 6) HistoryActivity — отвечают за отображение данных

пользователю.

Слой ViewModel явно не описан в UML диаграмме (рисунок 14), но имеется в приложении для взаимодействия между View и Model.

Главный экран приложения (MainActivity), предоставляет пользователю возможность выбора режима работы: (Пользователь, Инженер, Разработчик). MainActivity отвечает за навигацию между активностями приложения с помощью методов:

					ПГУ1.090501.05.001 ПЗ	Лист
						53
Изм.	Лист	№ докум.	Подпись	Дата		

- onCreate(): Инициализирует интерфейс и устанавливает обработчики событий для кнопок;
- onOptionsItemSelected(): Обрабатывает выбор пунктов меню (если реализовано).

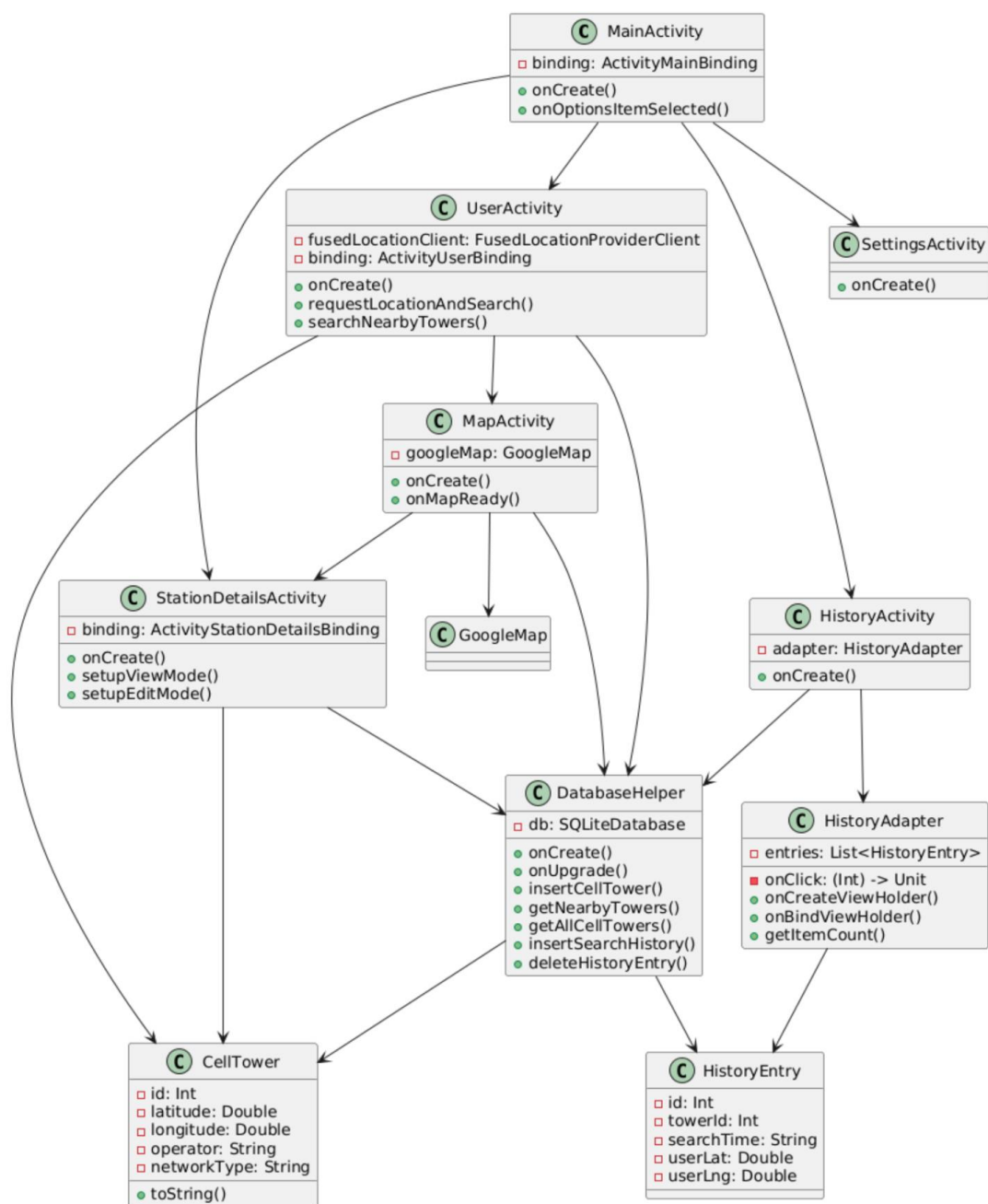


Рисунок 14 – Диаграмма классов приложения Cell Tower Locator

- атрибут `binding`: Связующий объект для взаимодействия с UI через View Binding;
- метод `onCreate()`: Инициализирует интерфейс и настраивает слушатели событий;
- метод `requestLocationAndSearch()`: Запрашивает текущее местоположение пользователя и инициирует поиск ближайших станций;
- метод `searchNearbyTowers()`: Выполняет запрос к базе данных или OpenCellID API для поиска станций.

Модуль `MapActivity` отображает карту с маркерами базовых станций. Позволяет пользователю взаимодействовать с картой (масштабирование, перемещение) и просматривать информацию о станциях.

- атрибут `googleMap`: Экземпляр Google Maps для отображения карты;
- метод `onCreate()`: Инициализирует карту;
- метод `onMapReady()`: Настраивает карту, добавляет маркеры и обрабатывает клики по ним.

Модуль `StationDetailsActivity` отображает подробную информацию о выбранной базовой станции. Предоставляет режим просмотра и редактирования данных. Имеет следующие атрибуты и методы:

- атрибут `binding`: Связующий объект для взаимодействия с UI;
- метод `onCreate()`: Инициализирует интерфейс и загружает данные станции;
- метод `setupViewMode()`: Настраивает интерфейс для просмотра данных;
- метод `setupEditMode()`: Настраивает интерфейс для редактирования данных.

Модуль `SettingsActivity` предоставляет пользователю возможность настроить параметры приложения, такие как автоматический поиск станций и уведомления.

Модуль HistoryActivity отображает историю поиска базовых станций.

Позволяет пользователю просматривать ранее найденные станции и удалять записи. Имеет следующие атрибуты и методы:

- атрибут adapter: Адаптер для отображения списка записей истории;
- метод onCreate(): Инициализирует интерфейс и загружает данные из базы данных.

Класс DatabaseHelper необходим для работы с базой данных SQLite. Класс управляет операциями вставки, удаления и выборки данных (CRUD) о базовых станциях и истории поиска. Имеет следующие методы:

- метод onCreate(): Создает таблицы базы данных при первом запуске;
- метод onUpgrade(): Обновляет структуру базы данных при изменении версии;
- метод insertCellTower(): Вставляет данные о новой базовой станции;
- метод getNearbyTowers(): Возвращает список ближайших станций на основе координат пользователя;
- метод getAllCellTowers(): Возвращает все сохраненные базовые станции;
- метод insertSearchHistory(): Вставляет запись в историю поиска;
- метод deleteHistoryEntry(): Удаляет запись из истории поиска.

Класс CellTower является моделью данных для представления информации о базовой станции. Имеет следующие атрибуты и методы:

- атрибут id: Уникальный идентификатор станции;
- атрибут latitude: Широта расположения станции;
- атрибут longitude: Долгота расположения станции;
- атрибут operator: Название оператора связи;
- атрибут networkType: Тип сети (2G, 3G, 4G, 5G);

– метод toString(): Возвращает строковое представление объекта для удобства отладки.

Модель данных для представления записи в истории поиска находится в классе HistoryEntry. Имеет следующие атрибуты:

- атрибут id: Уникальный идентификатор записи;
- атрибут towerId: Идентификатор связанной базовой станции;
- атрибут searchTime: Время выполнения поиска;
- атрибут userLat: Широта местоположения пользователя во время поиска;
- атрибут userLng: Долгота местоположения пользователя во время поиска.

Класс HistoryAdapter необходим для формирования списка записей истории поиска. Имеет следующие атрибуты и методы:

- атрибут entries: Список записей истории;
- атрибут onClick: Функция обратного вызова для обработки кликов по элементам списка;
- метод onCreateViewHolder(): Создает новый ViewHolder для элемента списка;
- метод onBindViewHolder(): Привязывает данные к элементу списка;
- метод getItemCount(): Возвращает количество элементов в списке.

Карта взаимодействия между классами представлена следующими списками:

- MainActivity;
- UserActivity;
- StationDetailsActivity;
- SettingsActivity;
- HistoryActivity.

– главный экран предоставляет навигацию к другим экранам приложения;

– MainActivity;

– MapActivity;

– DatabaseHelper;

– CellTower;

– экран пользователя взаимодействует с базой данных для поиска станций и передает данные на карту;

– MapActivity;

– StationDetailsActivity;

– DatabaseHelper;

– GoogleMap;

– карта отображает маркеры станций и позволяет перейти к детальной информации о них;

– StationDetailsActivity;

– DatabaseHelper;

– CellTower;

– экран деталей взаимодействует с базой данных для загрузки и редактирования данных станции;

– HistoryActivity;

– DatabaseHelper;

– HistoryAdapter;

– экран истории загружает данные из базы данных и отображает их через адаптер;

– DatabaseHelper;

– CellTower;

– HistoryEntry;

– база данных управляет данными о базовых станциях и записях истории;

					ПГУ1.090501.05.001 ПЗ	Лист
						58
Изм.	Лист	№ докум.	Подпись	Дата		

- HistoryAdapter;
- HistoryEntry;
- адаптер работает с моделью данных для отображения списка записей.

В результате процесса реализации создано приложение, интерфейс и описание процесса, использования которого будет показано в следующем разделе.

					ПГУ1.090501.05.001 ПЗ	Лист
						59
Изм.	Лист	№ докум.	Подпись	Дата		

6 Описание интерфейса приложения Cell Tower Locator

Интерфейс приложения реализован по принципу одностраничной навигации с центральным хабом (MainActivity) (рисунок 15). Все функциональные модули доступны через главный экран, что обеспечивает: четкое разделение ролей (Пользователь, Инженер, Разработчик), минимальное количество шагов для доступа к функциям, единую точку входа для настроек и истории. Ниже представлено описание основных диалогов приложения.

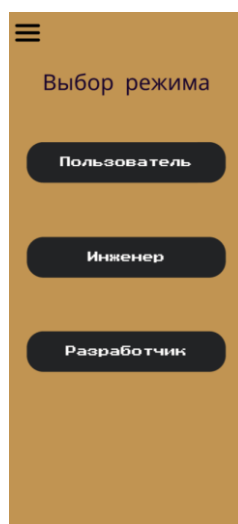


Рисунок 15 – Главное окно приложения Cell Tower Locator

По нажатию на кнопку «Пользователь» диалога *MainActivity* происходит переход на диалог сканирования текущего положения (рисунок 16).



Рисунок 16 – Режим Пользователя. Сканирование локации

По нажатию на кнопку «*Scanning*» происходит формирование карты местности с количеством вышек (рисунок 17).

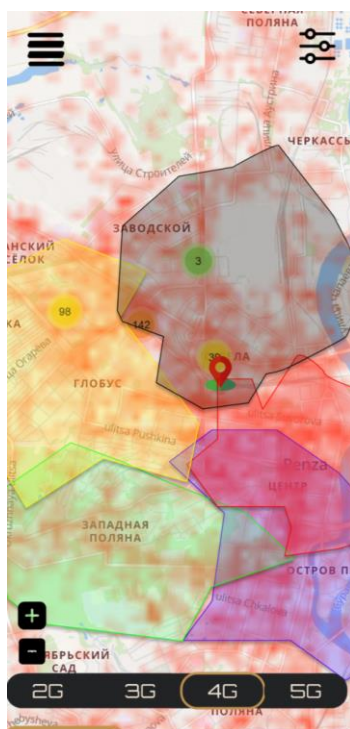


Рисунок 17 – Режим Пользователя. Карта города 4G

Сформированные карты (рисунки 18, 19) визуализируют области покрытия в зависимости от выбранной технологии мобильной связи (2G, 3G, 4G, 5G).

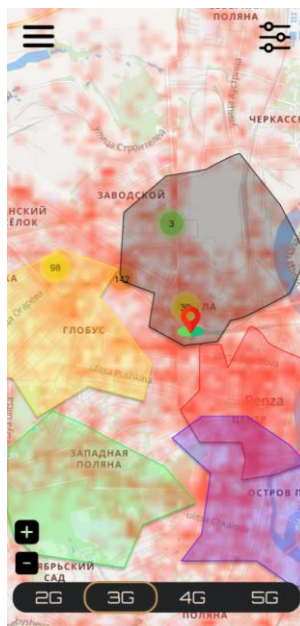


Рисунок 18 – Режим Пользователя. Карта города 3G

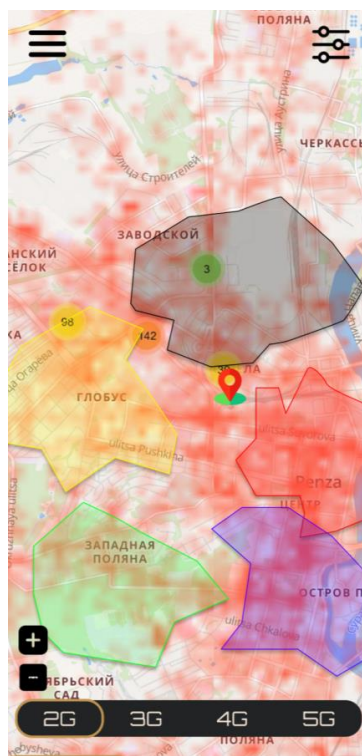


Рисунок 19 – Режим Пользователя. Карта города 2G

Также у пользователя имеется возможность изменения масштаба карты (рисунок 20) благодаря наличию на диалоговом окне соответствующих кнопок.

					ПГУ1.090501.05.001 ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		

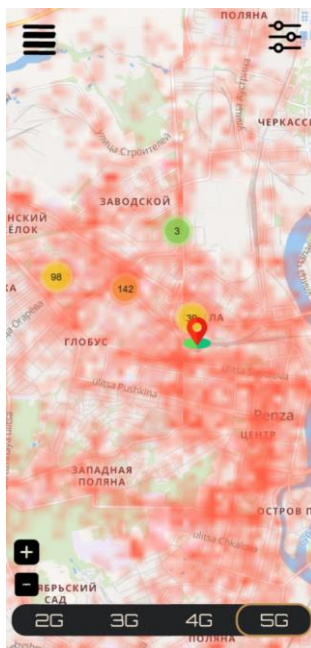


Рисунок 20 – Режим Пользователя. Карта города 5G

Масштабирование позволяет дитализировать карту до уровня улиц и отдельно стоящих зданий. В результате пользователь может анализировать отдельно стоящие вышки сотовой связи (рисунок 21).

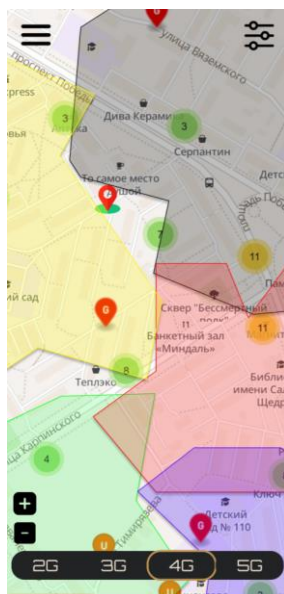


Рисунок 21 – Режим Пользователя. Режим масштабирования карты

Помимо визуального поиска требуемого объекта на карте способом изменения масштаба, пользователь может воспользоваться поиском конкретной улицы населенного пункта (рисунок 22). В результате

пользователь перейдет на карту улицы, где может выбрать вышкусотовой связи и посомтреть информацию о ней (рисунок 23).



Рисунок 22 – Режим Пользователя. Режим поиска

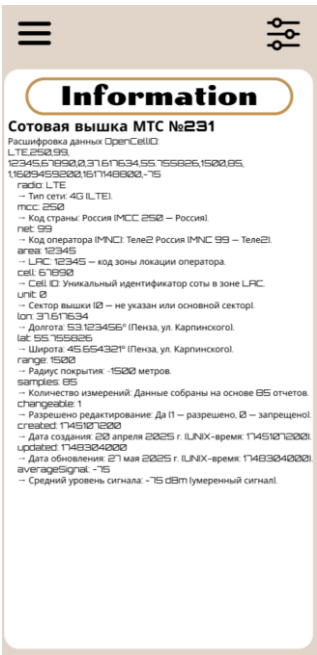


Рисунок 23 – Режим Пользователя. Информация о вышке связи

После получения требуемой информации в режиме «Пользователь» необходимо вернуться в главный диалог. Если есть необходимость в работе с точки зрения профессионала, необходимо выбрать кнопку «Инженер». В

режиме «Инженер» (рисунок 24) пользователь получает возможность создать собственную метку о вышке связи и отправить ее в облако OPENCELLID.

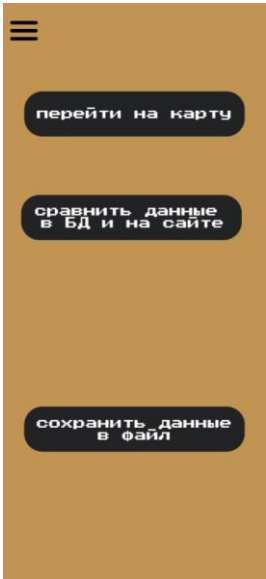


Рисунок 24 – Режим Инженер. Кнопки функций инженера

На карте текущей геолокации показано количество вышек связи в радиусе 5 километров (рисунок 25).



Рисунок 25 – Режим Инженер. Карта местности

Увеличение масштаба позволяет детализировать городской ландшафт и расположение вышек связи (рисунок 26).

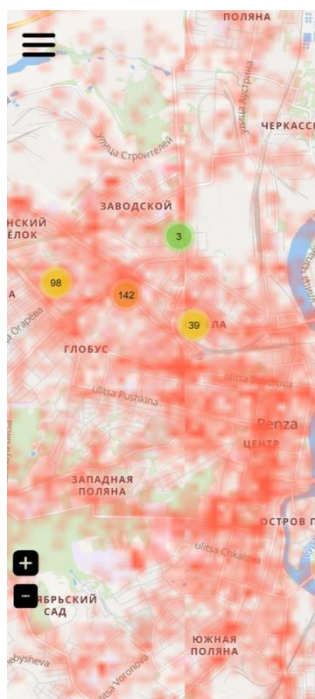


Рисунок 26 – Режим Инженер. Увеличение масштаба карты

В конечном итоге, максимальное масштабирование, показывает текущее положение инженерного работника и типы вышек сотовой связи в радиусе 5 километров (рисунок 27).

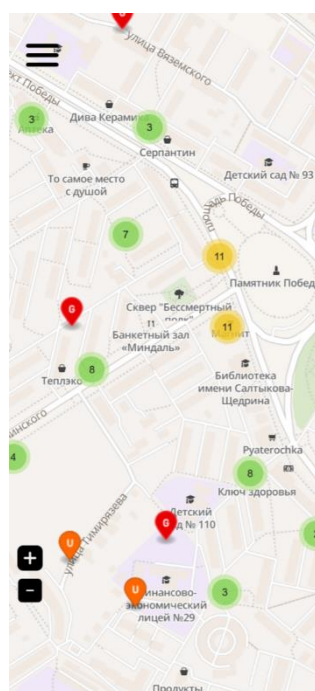


Рисунок 27 – Режим Инженер. Максимальный масштаб карты

Для создания собственной метки на карте, инженер выбирает место, которое маркируется (рисунок 28) и далее происходит переход на диалог

					ПГУ1.090501.05.001 ПЗ	Лист
						66
Изм.	Лист	№ докум.	Подпись	Дата		

заполнения информации о новой метке, что позволяет инженеру ввести данные о новой вышке сотовой связи (рисунок 29).

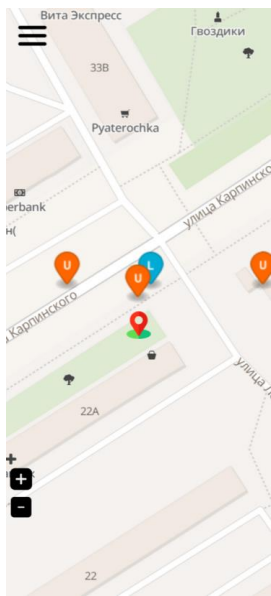


Рисунок 28 – Режим Инженер. Создание метки

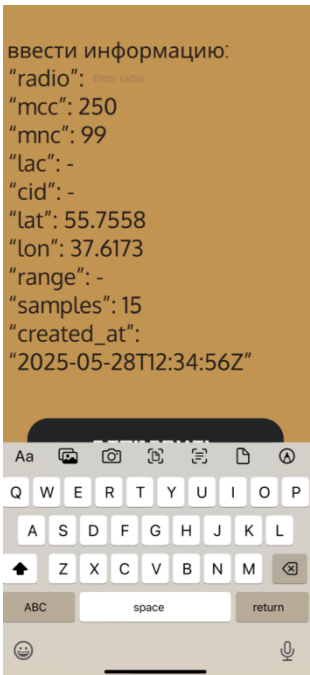
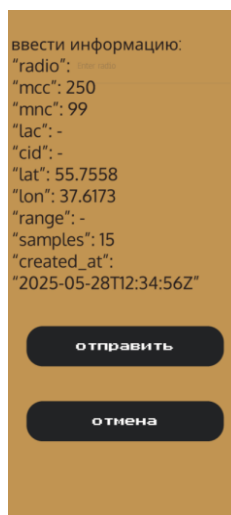


Рисунок 29 – Режим Инженер. Диалог информации о новой метке

Перед отправкой информации в облачный сервис для проверки на достоверность, у инженера имеется возможность отменить или исправить информацию (рисунок 30).



```

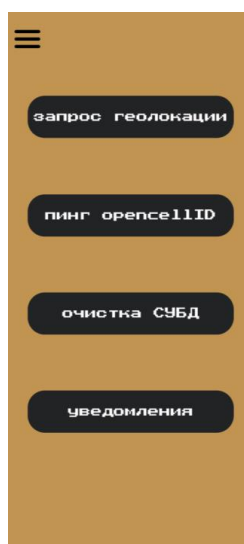
ввести информацию:
"radio": 
"mcc": 250
"mnc": 99
"lac": -
"cid": -
"lat": 55.7558
"lon": 37.6173
"range": -
"samples": 15
"created_at":
"2025-05-28T12:34:56Z"
  
```

отправить

отмена

Рисунок 30 – Режим Инженер. Диалог информации о новой метке

После получения и формирования необходимой информации в режиме «Инженер» необходимо вернуться в главный диалог. Если есть необходимость в работе с точки зрения разработчика приложения или изменения настроек, необходимо выбрать кнопку «Разработчик» главного диалога приложения. В режиме «Разработчик» (рисунок 31) пользователь получает возможность контролировать БД и иных настроек.



☰

запрос геолокации

пинг opencellID

очистка СУБД

уведомления

Рисунок 31 – Режим Разработчик. Диалог выбора функций

Переход по кнопке «Запрос геолокации» показывает текущее положение мобильного устройства (рисунок 32).

Текущая геолокация:
страна: РФ
область: Пенза
НП: Пенза
координаты: 53.2119, 44.9760

вернуться

Рисунок 32 – Режим Разработчик. Диалог «Запрос геолокации»

Переход по кнопке «Пинг OPENCELLID» проверяет связь с интернет ресурсом (рисунок 33), в случае успешного соединения диалог выдаст уведомление (рисунок 34).

Ожидайте...
Идёт проверка соединения
с сайтом: <https://opencellid.org/>

вернуться

Рисунок 33 –Диалог «Пинг OPENCELLID». Ожидание ответа сервера

					ПГУ1.090501.05.001 ПЗ	Лист
						69
Изм.	Лист	№ докум.	Подпись	Дата		

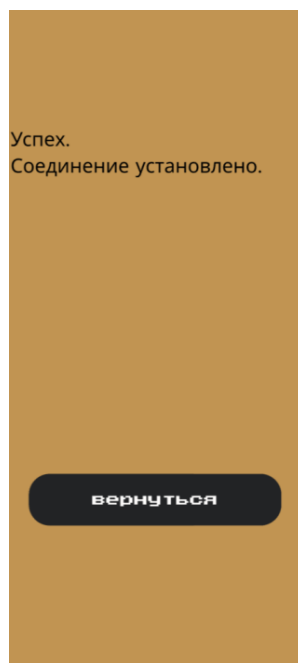


Рисунок 34 –Диалог «Пинг OPENCELLID». Соединение установлено

Переход по кнопке «Очистка СУБД» (рисунок 35) удаляет все записи из базы данных.

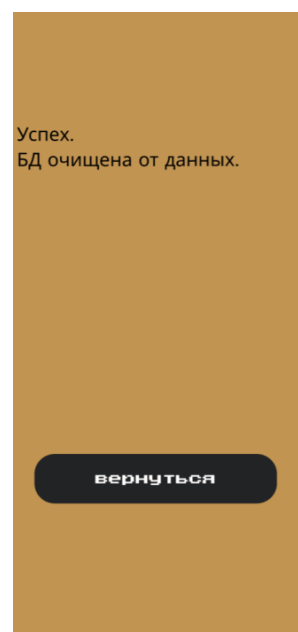


Рисунок 35 –Диалог «Очистка СУБД»

Кнопка «Сохранить данные в файл» на главном диалоговом окне разработчика позволяет сохранить текущее состояние БД в файловой системе устройства.

7 Экономическое обоснование эффективности проекта

Разработка программного обеспечения осуществлена на базе собственной площади 70 кв.м. Содержание помещения в месяц составляет 8 000 рублей.

Для разработки программного обеспечения приобретена вычислительная техника, лицензионное программное обеспечение и расходные материалы (обучающие материалы).

Данные по потребности в оборудовании и лицензионном программном обеспечении представлены соответственно в таблице 11 и 12.

Таблица 11 – Потребность в оборудовании

Оборудование	Организация	Кол-во	Ориентировочная цена за единицу, руб
Компьютер самостоятельная сборка	ООО «Технолинк»	5	60000
МФУ лазерный Conica 1100	ООО «Ситилинк»	1	11500
Клавиатура и мышь	ООО «ДНС»	5	1000
Монитор LG 27 дюймов	ООО «ДНС»	5	27500

Таблица 12 – Потребность в лицензионном программном обеспечении

Лицензионное программное обеспечение	Источник	Цена, рублей	Срок действия лицензии, дней
Microsoft Windows 10 Pro	интернет-магазин ООО «Технолинк»	31000	бессрочная
Антивирус Kaspersky 2023/2024	интернет-магазин ООО «Технолинк»	2100	365

Разработка мобильного программного обеспечения осуществлялась командой разработчиков, состав которой представлен в таблице 13. Состав коллектива разработчиков и информация по заработной плате разработчиков программного обеспечения и их количественному составу приведены в таблице 13.

Таблица 13 – Занятость и заработная плата разработчиков

Специалист	Количество, человек	Сумма заработной платы, рублей
Руководитель проекта	1	80000
Тимлидер	1	60000
Программист	1	60000
Тестировщик	1	50000

Себестоимость разработки программного продукта включает следующие статьи затрат:

- затраты на приобретение оборудования;
- затраты на приобретение лицензионного программного обеспечения;
- затраты на дополнительные материалы;

- затраты на заработную плату;
- затраты на обслуживание.

Затраты на оборудование в период разработки составят 454000 рублей на основании данных в таблице 12.

Затраты на программное обеспечение в период разработки составят 165500 рублей и представлены в таблице 14.

Затраты на дополнительные материалы в период разработки составят 0 рублей в соответствии с условием что потребности в закупке нет.

Затраты на заработную плату включают затраты на основную и дополнительную заработную плату разработчиков программного обеспечения и отчисления на страховые взносы.

Затраты на основную заработную плату разработчиков определяются исходя из установленной заработной платы, количества специалистов и их занятости в процессе разработки программного продукта (продолжительность работы в днях).

Дополнительная заработная плата составляет 15% от затрат на основную заработную плату разработчиков.

С учетом выходных дней при 5-дневной рабочей неделе количество рабочих дней в месяце в среднем составляет 21 день.

Затраты на основную и дополнительную заработную плату разработчиков программного обеспечения представлены в таблице 14.

					ПГУ1.090501.05.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 14 – Затраты на основную и дополнительную заработную плату разработчиков

Исполнитель	Количество, человек	Сумма заработной платы, рублей	Период занятости, дней	Основная заработная плата, рублей	Дополнительная заработная плата, рублей	Всего рублей
Руководитель проекта	1	80000	42	240000	36000	276000
Тимлидер	1	60000	42	180000	27000	207000
Программист	1	60000	42	180000	27000	207000
Тестировщик	1	50000	42	150000	22500	172500
Итого	5	250000	168	750000	112500	862500

Таким образом, затраты на заработную плату разработчиков составят 862500 рублей.

Отчисления на страховые взносы по нормам, установленным законодательством в ПРФ, ФОМС, ФСС РФ вычисляются исходя из данных, представленных ниже.

Отчисления на страховые взносы производится согласно «Налоговому кодексу Российской Федерации (часть вторая)» от 05.08.2000 № 117-ФЗ (ред. от 26.03.2022).

Действующие тарифы на страховые взносы работодателей установлены статьей 34 НК РФ и составляют: ОПС (обязательное пенсионное страхование) – 22%, ВНиМ (страхование временной нетрудоспособности и в связи с материнством) – 2,9%, ОМС (обязательное медицинское страхование) – 5,1%.

Общая сумма отчислений на страховые взносы составит 258750 рублей.

					ПГУ1.090501.05.001 ПЗ	Лист
						74
Изм.	Лист	№ докум.	Подпись	Дата		

Общая сумма затрат является суммой: затрат на оплату труда включая налоги и сборы, затраты на закупку оборудования и лицензионного ПО и составляет 1740750 рублей.

В процессе эксплуатации пользователи получают свободно распространяемое ПО и техническую поддержку от команды разработки.

					ПГУ1.090501.05.001 ПЗ	Лист
						75
Изм.	Лист	№ докум.	Подпись	Дата		

8 Безопасность жизнедеятельности при работе инженера-программиста

Рабочая деятельность инженера-программиста сопряжена с воздействием специфических факторов производственной среды, способных оказать негативное влияние на здоровье. Соблюдение требований безопасности жизнедеятельности (БЖД), регламентированных государственными нормативными документами, является обязательным условием для минимизации профессиональных рисков и обеспечения сохранности здоровья сотрудников. К ключевым вредным факторам для данной профессии относятся вынужденная статичная поза (сидячая работа) и длительное воздействие компьютерной техники. Основными регулирующими документами в этой сфере являются Санитарно-эпидемиологические правила и нормативы СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы», Трудовой кодекс Российской Федерации и Межгосударственный стандарт ГОСТ 12.0.003–2015 «Система стандартов безопасности труда. Опасные и вредные производственные факторы. Классификация».

8.1 Эргономические требования к рабочему месту

Организация рабочего места инженера-разработчика должна строго соответствовать эргономическим принципам для предотвращения развития профессиональных заболеваний опорно-двигательного аппарата, органов зрения и системы кровообращения, возникающих из-за длительной статической нагрузки и визуального напряжения.

Основные требования к оснащению и организации рабочего места подразумевают:

- расстояние от глаз пользователя до экрана должно составлять 50-70 см;

					ПГУ1.090501.05.001 ПЗ	Лист
						76
Изм.	Лист	№ докум.	Подпись	Дата		

- верхний край экрана должен располагаться на уровне глаз или чуть ниже (на 5-10 см);
- использование современных мониторов с качественной матрицей, обеспечивающих стабильное изображение и минимальное мерцание;
- экран должен быть ориентирован таким образом, чтобы исключить попадание прямых солнечных лучей и бликов (допускается использование антибликовых фильтров или жалюзи);
- спинка кресла должна обеспечивать поддержку поясницы и способствовать сохранению физиологического изгиба позвоночника (лордоза), исключая напряжение в спине и пояснице;
- плечи пользователя должны быть расслаблены;
- регулировка высоты сиденья должна позволять поставить стопы всей поверхностью на пол или специальную подставку (при необходимости), обеспечивая угол в коленных и тазобедренных суставах близкий к 90 градусам;
- подлокотники (при наличии) должны располагаться на уровне, позволяющем сгибать руки в локтях под углом 70-90 градусов, не вызывая подъема плеч;
- стол должен иметь пространство для ног: высота не менее 600 мм, глубина на уровне колен не менее 450 мм, на уровне стоп – не менее 600 мм, ширина – не менее 500 мм (СанПиН 2.2.3670–20);
- рабочая поверхность должна быть свободной, достаточной для размещения необходимого оборудования и документов;
- ничто не должно ограничивать движения ног под столом;
- локти и запястья должны располагаться на одной высоте при работе с клавиатурой и мышью, предплечья – параллельно полу;
- обязательны регламентированные перерывы для смены деятельности: кратковременная разминка (2-3 минуты) для глаз, шеи, спины, рук и ног каждые 30-40 минут работы за ПК;

					ПГУ1.090501.05.001 ПЗ	Лист
						77
Изм.	Лист	№ докум.	Подпись	Дата		

– рекомендовано выполнение комплекса производственной гимнастики.

8.2 Требования к микроклимату и освещению рабочего помещения

Оптимальные параметры микроклимата и освещенности являются критически важными факторами для поддержания высокой работоспособности, концентрации внимания и профилактики утомления инженерно-технического персонала на основании СанПиН:

– эквивалентный уровень шума в помещении не должен превышать 50 дБА (СанПиН 2.2.3670–20) [7]. Превышение допустимых уровней шума приводит к снижению концентрации, повышению утомляемости и общей работоспособности. Пиковый скорректированный уровень звука ограничен 127 дБА;

– для снижения уровня шума рекомендуется применение шумоизолирующих материалов в отделке стен и потолков, установка окон с повышенными шумозащитными характеристиками;

– норма искусственной освещенности на рабочей поверхности для работ с ПК составляет 300-500 люкс;

– освещение должно быть равномерным, без резких теней и бликов на экранах мониторов. Рекомендуется комбинированная система освещения (общее + местное);

– площадь на одно рабочее место должна составлять не менее 5 кв.м.. Для обеспечения равномерного освещения рекомендуется расчетное расположение светильников общего освещения;

– обязательно использование средств защиты от прямого солнечного света (жалюзи, шторы);

– оптимальная температура воздуха в холодный период года: 22-23 °С; в теплый период года: 22-25 °С. Допустимые диапазоны: 19-24 °С (холодный период) и 20-29 °С (теплый период);

					<i>ПГУ1.090501.05.001 ПЗ</i>	Лист
						78
Изм.	Лист	№ докум.	Подпись	Дата		

- относительная влажность воздуха должна поддерживаться в пределах 55-65% при температуре около +20 °С;
- для обеспечения требуемых параметров микроклимата необходимо наличие систем кондиционирования и вентиляции воздуха с возможностью регулировки температуры и направления воздушного потока. В случае необходимости следует использовать увлажнители воздуха.

8.3 Организация безопасной работы с персональным компьютером

Безопасная эксплуатация персональных компьютеров (ПК) регламентируется Типовой инструкцией по охране труда при работе на персональном компьютере ТОО Р-45-084-01 [9]. Основные аспекты включают:

- классификация работ и регламент перерывов: В зависимости от характера деятельности устанавливаются различные режимы труда и отдыха:
 - 1) группа а (чтение информации с экрана): 2 перерыва по 15 минут каждый (без учета обеденного) в течение 8-часовой смены;
 - 2) группа б (ввод информации): перерывы по 10 минут каждый час работы;
 - 3) группа в (творческая работа в режиме диалога с ПК): перерывы по 15 минут каждый час работы;
- обязательные правила безопасности:
 - 1) прохождение вводного и периодического инструктажей по охране труда и пожарной безопасности;
 - 2) строгое соблюдение требований инструкций по эксплуатации ПК и периферийного оборудования;
 - 3) соблюдение правил электробезопасности при обращении с оборудованием;
 - 4) контроль за соблюдением установленного температурного режима в помещении;

					ПГУ1.090501.05.001 ПЗ	Лист
						79
Изм.	Лист	№ докум.	Подпись	Дата		

5) наличие и знание инструкции по пожарной безопасности, расположения первичных средств пожаротушения и путей эвакуации;

6) проверка исправности оборудования (целостность кабелей, отсутствие повреждений корпусов, стабильность работы) перед началом работы;

7) самоконтроль состояния здоровья;

8) немедленное сообщение руководителю о недомогании или обнаружении неисправности оборудования.

Соблюдение изложенных требований и правил безопасности жизнедеятельности на рабочем месте инженера-программиста является неотъемлемой частью производственного процесса. Выполнение этих требований гарантируется соответствующими нормативными актами и является обязанностью как работодателя, так и самого работника.

					ПГУ1.090501.05.001 ПЗ	Лист
						80
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе выполнения ВКР была изучена предметная область реализации приложения сканирования местности на предмет наличия вышек сотовой связи, в результате чего сформулировано техническое задание на разработку программного обеспечения.

Был произведен выбор стека технологий, выполнено проектирование, в результате которого разработаны UML диаграммы и реализовано ПО.

Приложение реализовано в соответствии с техническим заданием. Тестирование программного обеспечения выполнено успешно.

В экономическом разделе был произведен расчет трудоемкости и затрат на разработку программного обеспечения.

В разделе безопасности жизнедеятельности описаны общие требования к рабочему месту пользователя ПЭВМ.

					ПГУ1.090501.05.001 ПЗ	Лист
						81
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. GSMA Intelligence: The Mobile Economy 2024 [Электронный ресурс] - Режим доступа: <https://www.gsmainelligence.com/mobileeconomy/> (дата обращения: 15.05.2025).

2. Официальный отчет о состоянии мобильной индустрии и прогнозах развития Ericsson Mobility Report November 2023 [Электронный ресурс] - Режим доступа: <https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/november-2023> (дата обращения: 15.05.2025)

Анализ глобальных тенденций мобильного трафика и технологий связи

3. WorldGlobalStat [Электронный ресурс] - Режим доступа: <https://www.worldglobalstat.org/reports/mobile-november-2024> (Дата обращения: 15.05.2025)

Анализ глобальных тенденций мобильного трафика и технологий связи

4. Официальная документация Android Developers [Электронный ресурс] - Режим доступа: <https://developer.android.com/> (дата обращения: 10.04.2024)

5. Официальная документация Android Developers [Электронный ресурс] - Режим доступа: <https://developer.android.com/> (Дата обращения: 01.06.2025).

6. Жемаеров Д., Исакова С. Kotlin in Action [Электронный ресурс] - Режим доступа: <https://www.manning.com/books/kotlin-in-action> (Дата обращения: 10.07.2024).

7. Оуэн М. The Definitive Guide to SQLite [Электронный ресурс] - Режим доступа: <https://www.apress.com/gp/book/9781484219650> (Дата обращения: 10.07.2024).

8. Google Maps Platform Documentation [Электронный ресурс] - Режим доступа: <https://developers.google.com/maps/documentation> (дата обращения: 01.06.2025).

9. Developer Documentation: Core Location [Электронный ресурс] - Режим доступа: <https://developer.android.com/documentation/corelocation>(дата обращения: 01.06.2025).

10. OpenCellID Project: Crowdsourced Cell Tower Database [Электронный ресурс] - Режим доступа: <https://opencellid.org/>(дата обращения: 01.06.2025).

					ПГУ1.090501.05.001 ПЗ	Лист
						83
Изм.	Лист	№ докум.	Подпись	Дата		

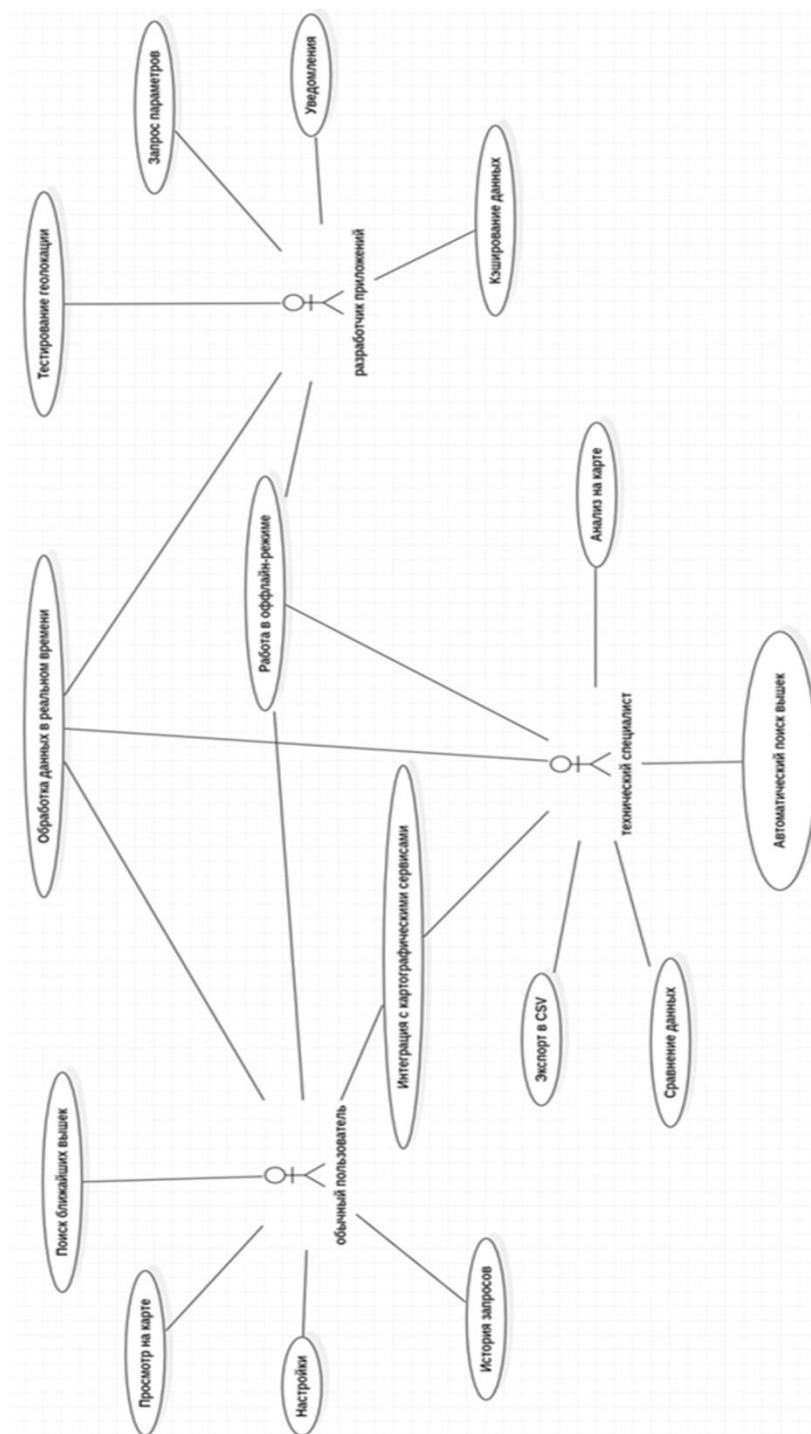
Приложение А

(обязательное)

Приложение обучения специалистов в области информационных
технологий
UML-диаграммы

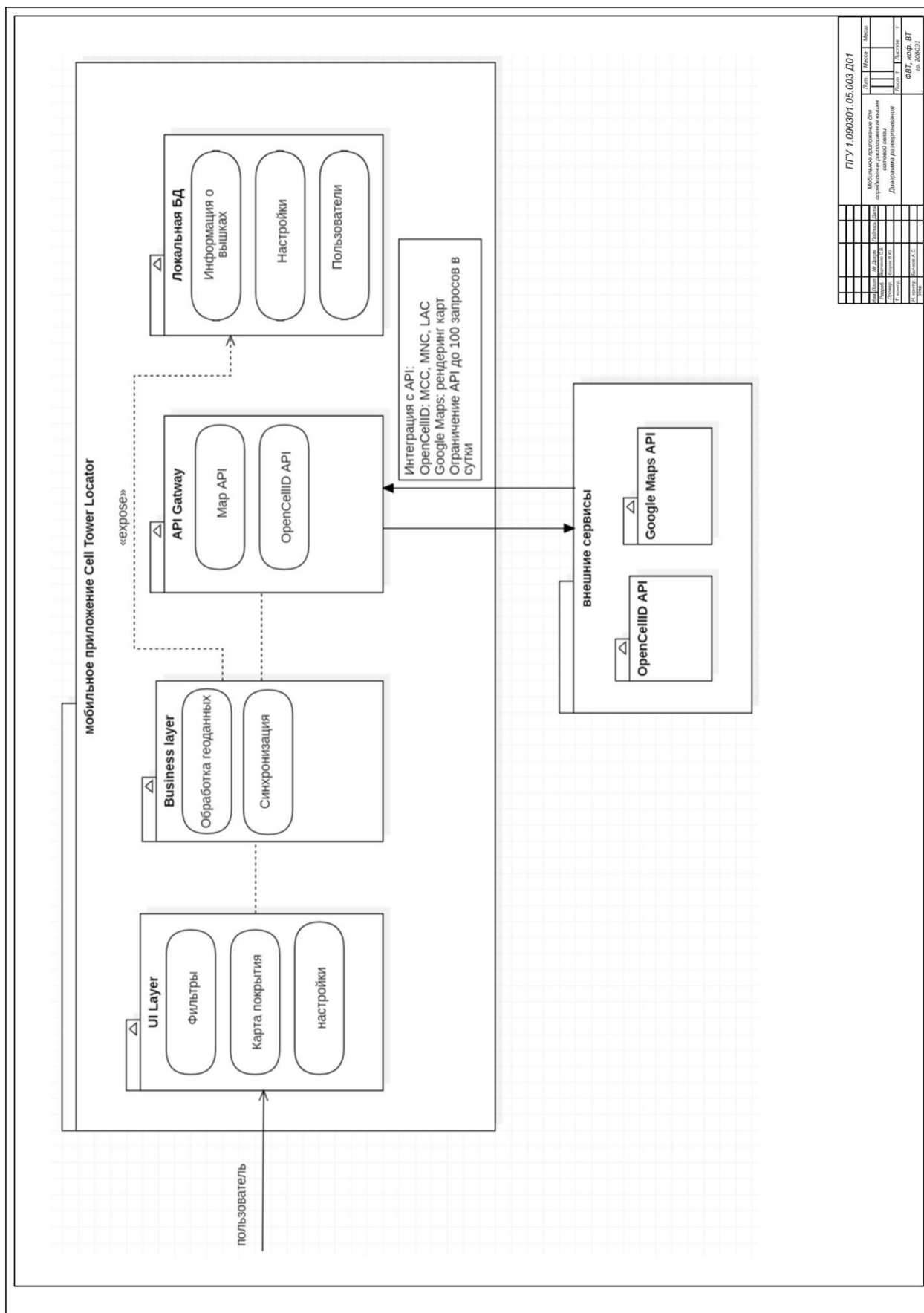
					ПГУ 1.090501.02.001 ПЗ	Лист
						84
Изм.	Лист	№ докум	Подпись	Дата		

Приложение А.1 — Диаграмма вариантов использования

[illegible]

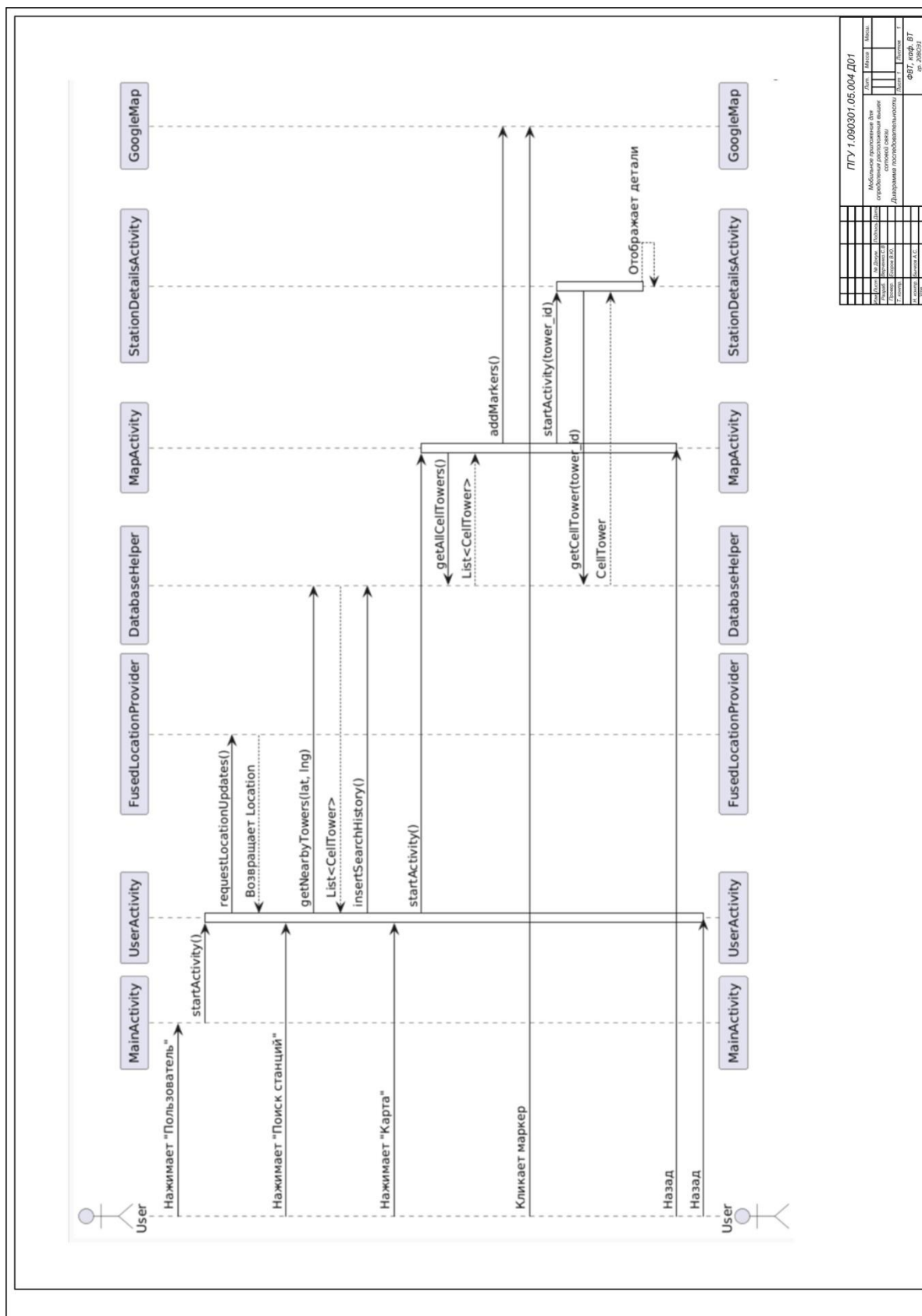
Изм.	Лист	№ докум	Подпись	Дата

Приложение А.2 — Диаграмма развертывания

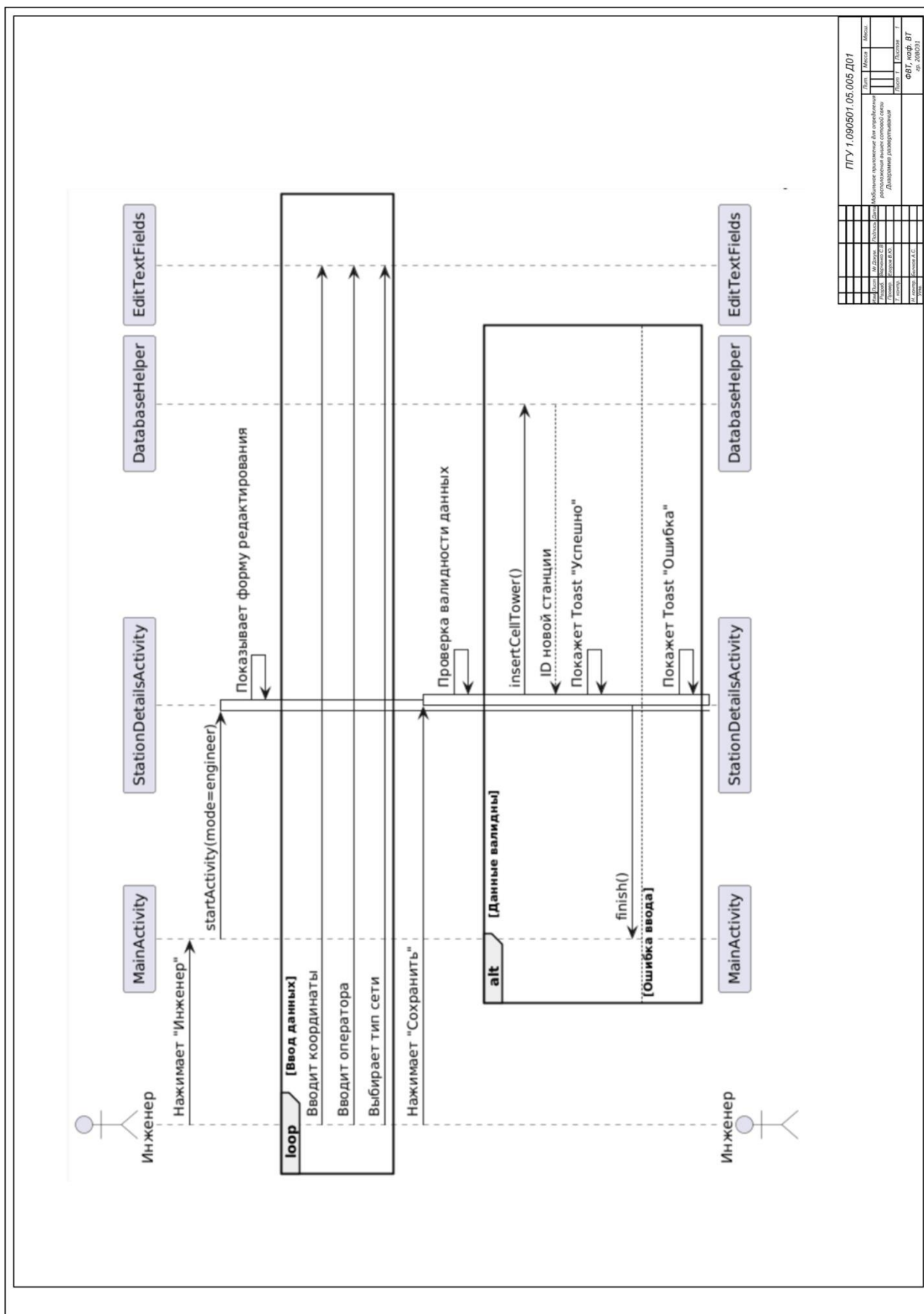
[illegible]

Изм.	Лист	№ докум	Подпись	Дата

Приложение А.3 — Диаграмма последовательности

[illegible]

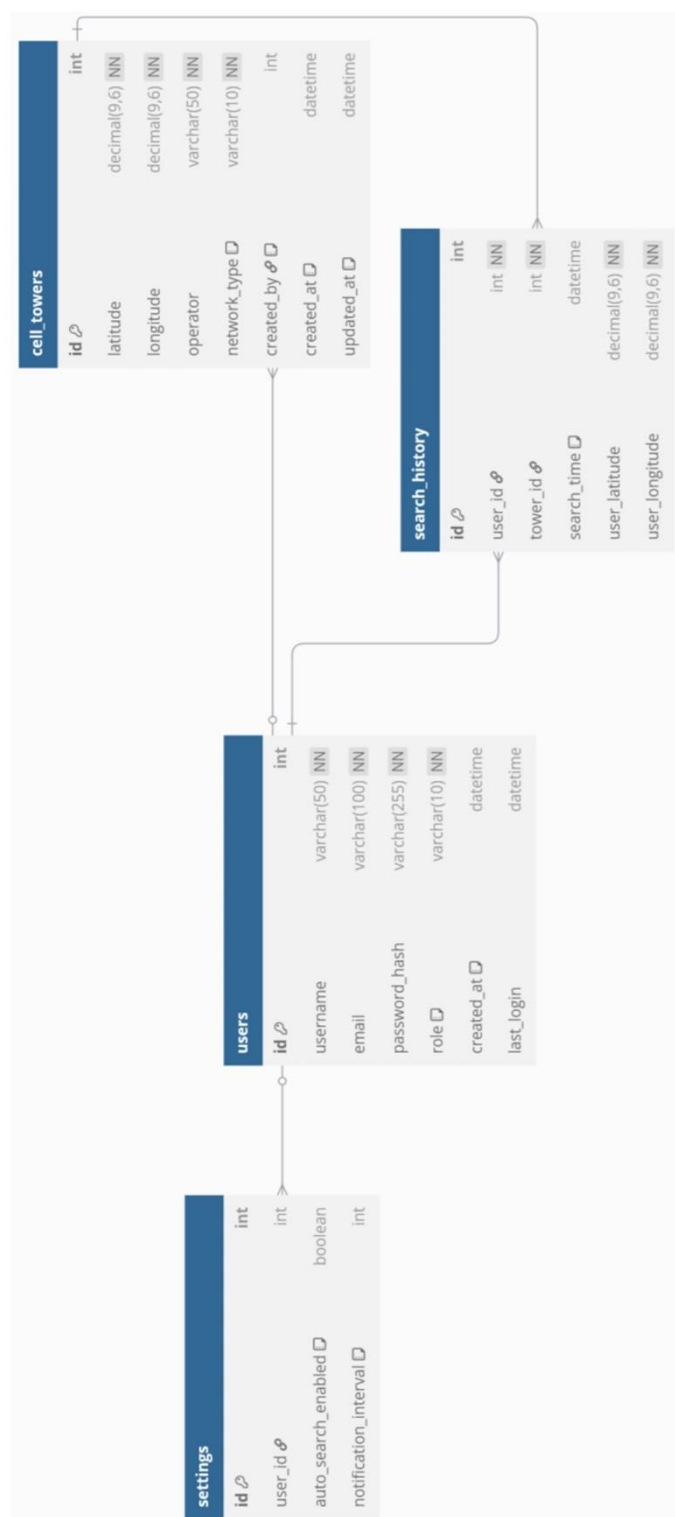
Приложение А.4 — Диаграмма последовательности



ПУ 1.090501.05.005 Д01									
Имя	Фамилия	Отчество	Дата рождения	Пол	Место	Дата	Время	Страна	Город
Иван	Иванов	Иванович	1990-01-01	М	Москва	12.12.2023	14:30	Россия	Москва
Диаграмма разработана									
Имя	Фамилия	Отчество	Дата рождения	Пол	Место	Дата	Время	Страна	Город
Иван	Иванов	Иванович	1990-01-01	М	Москва	12.12.2023	14:30	Россия	Москва
Файл: 001_001.d01									
Страна: Россия									
Город: Москва									

Изм.	Лист	№ докум	Подпись	Дата

Приложение А.5 — Диаграмма данных

[illegible]

Приложение Б

(обязательное)

Приложение обучения специалистов в области информационных
технологий

Листинг программы

Activity_main.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:id="@+id/btnUser"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Пользователь" />

    <Button
        android:id="@+id/btnEngineer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Инженер" />

    <Button
        android:id="@+id/btnDeveloper"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Разработчик" />

    <Button
        android:id="@+id/btnSettings"
        android:layout_width="wrap_content"
```

```

        android:layout_height="wrap_content"
        android:text="Настройки" />

```

```

<Button
    android:id="@+id/btnHistory"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="История" />
</LinearLayout>

```

StationDetailsActivity.kt

```

// StationDetailsActivity.kt
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import kotlinx.android.synthetic.main.activity_station_details.*

class StationDetailsActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_station_details)

        val station = intent.getSerializableExtra("station") as? Station
        station?.let {
            tvStationId.text = "ID: ${it.id}"
            tvCoordinates.text = "Coordinates: ${it.latitude}, ${it.longitude}"
            tvOperator.text = "Operator: ${it.operator}"
            tvNetworkType.text = "Network Type: ${it.networkType}"
        }
    }
}

```

MapActivity.kt

```

import android.os.Bundle

```

					ПГУ 1.090501.02.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		91

```

import androidx.appcompat.app.AppCompatActivity
import com.google.android.gms.maps.CameraUpdateFactory
import com.google.android.gms.maps.GoogleMap
import com.google.android.gms.maps.OnMapReadyCallback
import com.google.android.gms.maps.SupportMapFragment
import com.google.android.gms.maps.model.LatLng
import com.google.android.gms.maps.model.MarkerOptions

class MapActivity : AppCompatActivity(), OnMapReadyCallback {

    private lateinit var mMap: GoogleMap

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_map)
        val mapFragment = supportFragmentManager.findFragmentById(R.id.map) as
SupportMapFragment
        mapFragment.getMapAsync(this)
    }

    override fun onMapReady(googleMap: GoogleMap) {
        mMap = googleMap

        // Пример добавления маркера
        val stationLocation = LatLng(55.7558, 37.6173) // Пенза
        mMap.addMarker(MarkerOptions().position(stationLocation).title("Base Station"))
        mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(stationLocation, 10f))

        // Обработка нажатия на маркер
        mMap.setOnMarkerClickListener { marker ->
            val intent = Intent(this, StationDetailsActivity::class.java)
            startActivity(intent)
            true
        }
    }
}

```

```

    }
}
}

```

UserActivity.kt

```

import android.location.Location
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import com.google.android.gms.location.FusedLocationProviderClient
import com.google.android.gms.location.LocationServices
import kotlinx.android.synthetic.main.activity_user.*

class UserActivity : AppCompatActivity() {

    private lateinit var fusedLocationClient: FusedLocationProviderClient

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_user)

        fusedLocationClient = LocationServices.getFusedLocationProviderClient(this)

        btnFindStations.setOnClickListener {
            getLastLocation()
        }

        btnMap.setOnClickListener {
            startActivity(Intent(this, MapActivity::class.java))
        }
    }

    private fun getLastLocation() {
        fusedLocationClient.lastLocation.addOnSuccessListener { location: Location? ->

```

					ИПУ 1.090501.02.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		93

```

        location?.let {
            tvLocation.text = "Latitude: ${it.latitude}, Longitude: ${it.longitude}"
        }
    }
}
}

```

MainActivity.kt

```

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        btnUser.setOnClickListener {
            startActivity(Intent(this, UserActivity::class.java))
        }

        btnEngineer.setOnClickListener {
            startActivity(Intent(this, StationDetailsActivity::class.java))
        }

        btnDeveloper.setOnClickListener {
            // Реализовать действия для разработчика
        }

        btnSettings.setOnClickListener {
            startActivity(Intent(this, SettingsActivity::class.java))
        }
    }
}

```

```

        btnHistory.setOnClickListener {
            startActivity(Intent(this, HistoryActivity::class.java))
        }
    }
}

```

DatabaseHelper.kt

```

// DatabaseHelper.kt
import android.content.ContentValues
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class DatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION)
{

    companion object {
        private const val DATABASE_NAME =
            "cell_tower_locator.db"
        private const val DATABASE_VERSION = 1
        private const val TABLE_STATIONS = "stations"
        private const val COLUMN_ID = "id"
        private const val COLUMN_LATITUDE = "latitude"
        private const val COLUMN_LONGITUDE = "longitude"
        private const val COLUMN_OPERATOR = "operator"
        private const val COLUMN_NETWORK_TYPE = "network_type"
    }

    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = ("CREATE TABLE $TABLE_STATIONS ("
            + "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, "
            + "$COLUMN_LATITUDE REAL, "
            + "$COLUMN_LONGITUDE REAL, "

```

```

        + "$COLUMN_OPERATOR TEXT, "
        + "$COLUMN_NETWORK_TYPE TEXT)")
    db?.execSQL(createTable)
}

override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int,
newVersion: Int) {
    db?.execSQL("DROP TABLE IF EXISTS $TABLE_STATIONS")
    onCreate(db)
}

fun addStation(station: Station) {
    val db = this.writableDatabase
    val values = ContentValues().apply {
        put(COLUMN_LATITUDE, station.latitude)
        put(COLUMN_LONGITUDE, station.longitude)
        put(COLUMN_OPERATOR, station.operator)
        put(COLUMN_NETWORK_TYPE, station.networkType)
    }
    db.insert(TABLE_STATIONS, null, values)
    db.close()
}

fun getAllStations(): List<Station> {
    val stations = mutableListOf<Station>()
    val db = this.readableDatabase
    val cursor = db.rawQuery("SELECT * FROM
$TABLE_STATIONS", null)
    while (cursor.moveToNext()) {
        val id =
cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN_ID))
        val latitude =
cursor.getDouble(cursor.getColumnIndexOrThrow(COLUMN_LATITUDE))
        val longitude =
cursor.getDouble(cursor.getColumnIndexOrThrow(COLUMN_LONGITUDE))

```

```

val operator =
    cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_OPERATOR))
        val networkType =
            cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_NETWORK_TYPE))

        stations.add(Station(id, latitude, longitude,
            operator, networkType))
    }
    cursor.close()
    return stations
}

fun deleteStation(id: Int) {
    val db = this.writableDatabase
    db.delete(TABLE_STATIONS, "$COLUMN_ID = ?",
        arrayOf(id.toString()))
    db.close()
}
}

```

Приложение В

(обязательное)

Приложение обучения специалистов в области информационных
технологий
Презентация

					ПГУ 1.090501.02.001 ПЗ	Лист
						98
Изм.	Лист	№ докум	Подпись	Дата		

Мобильное приложение для определения расположения вышек сотовой связи

Автор работы:

студент гр. 20ВОЭ1 Верченко Сергей Витальевич

Научный руководитель:

к.т.н., доцент Егоров Валерий Юрьевич

Пенза, 2025

Цель работы и объекты

Цель работы – Разработать бесплатное мобильное приложение для определения информации о вышках сотовой связи.

Приложение должно выполнять следующие функции:

- отображение на интерактивной карте информации о вышках сотовой связи, возможность настройки информации о мобильных операторах из открытых источников,
- сохранение информации о мобильных операторах в локальной СУБД.

Приложение должно иметь возможность работы с интерактивной картой.

Программное обеспечение реализуется с помощью среды разработки Android Studio.

В качестве СУБД используется компактная встраиваемая СУБД SQLite.

Задачи приложения

Приложение реализовано в виде мобильного клиента облачных сервисов: OpenCellID и Google Maps.

Приложение обрабатывает и визуализирует собранную с данных платформ информацию на интерактивной карте города в трех режимах: пользователь, инженер, разработчик.

Полученная информация хранится локально на мобильном устройстве в СУБД SQLite.

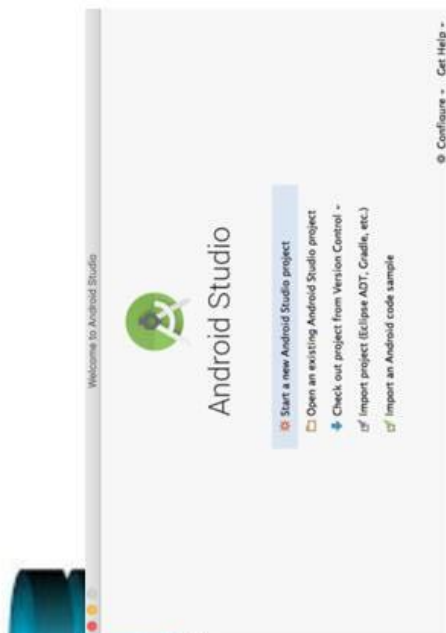
Варианты использования

Группа	Особенности использования
Технические специалисты	Расширенная аналитика, экспорт сырых данных
Разработчики	Доступ к API, <u>кастомизация логики</u> запросов
Обычные пользователи	Упрощенный интерфейс, автоматические настройки

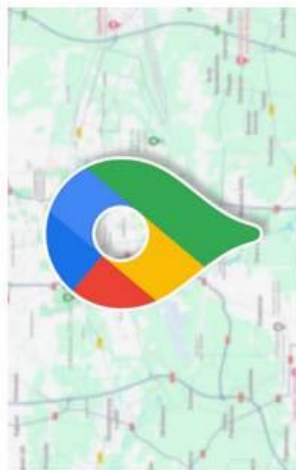
Сравнение аналогов

Параметр	OpenSignal	Network Cell Info Lite	OpenCellID
Тип лицензии	Проприетарная	Проприетарная	Открытая
API для разработки	Ограниченный доступ	Отсутствует	Полный доступ
Стоимость	Платный (от \$20/мес.)	Бесплатно с рекламой	Бесплатно
Данные для экспорта	Только агрегированные	Нет	Координаты вышек
Пример использования	Анализ скорости интернета	Техническая диагностика	Визуализация на карте

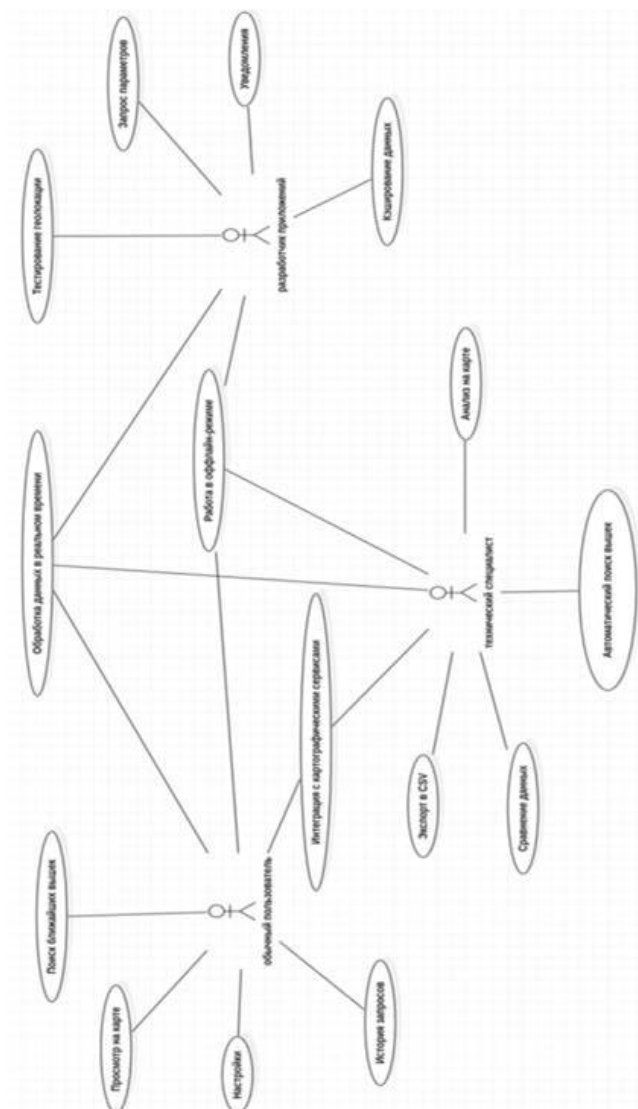
SQLite



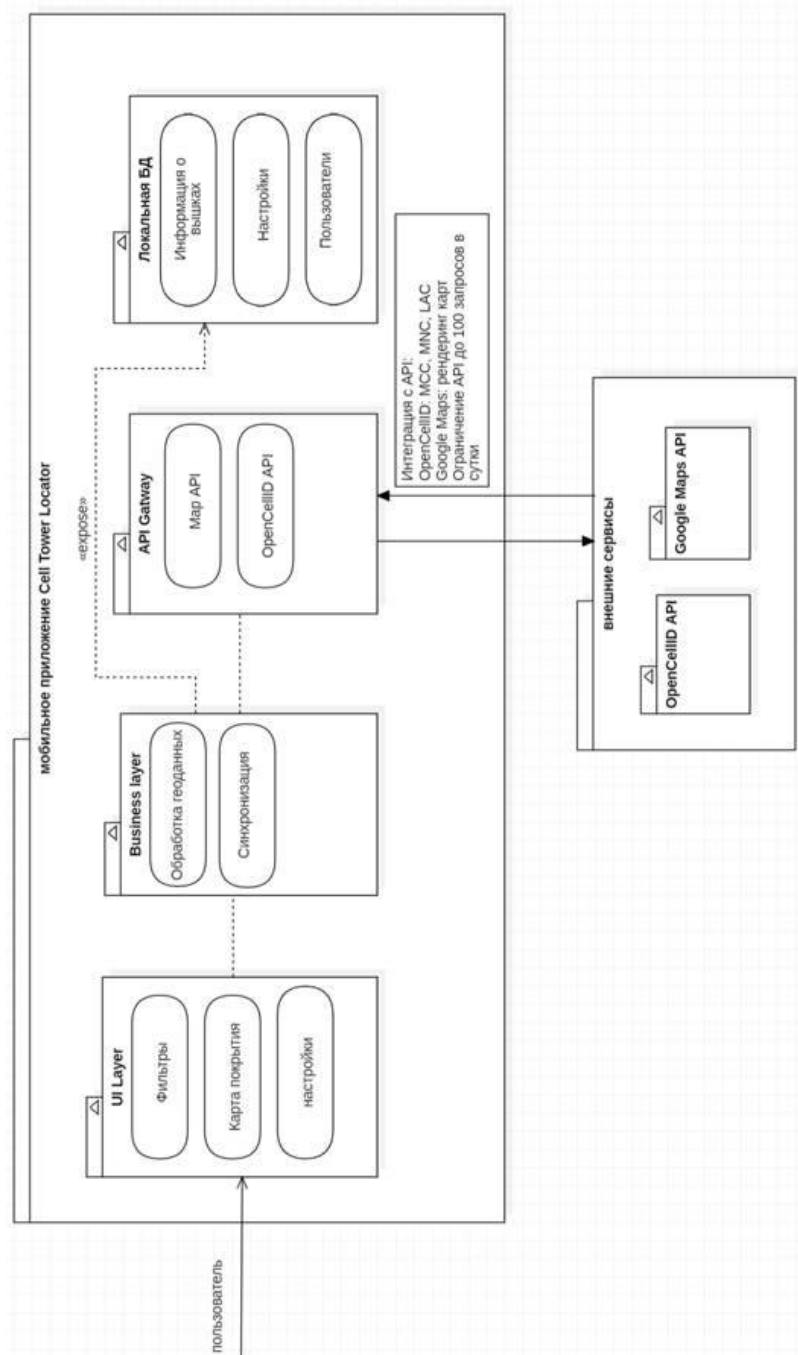
Android Monitor



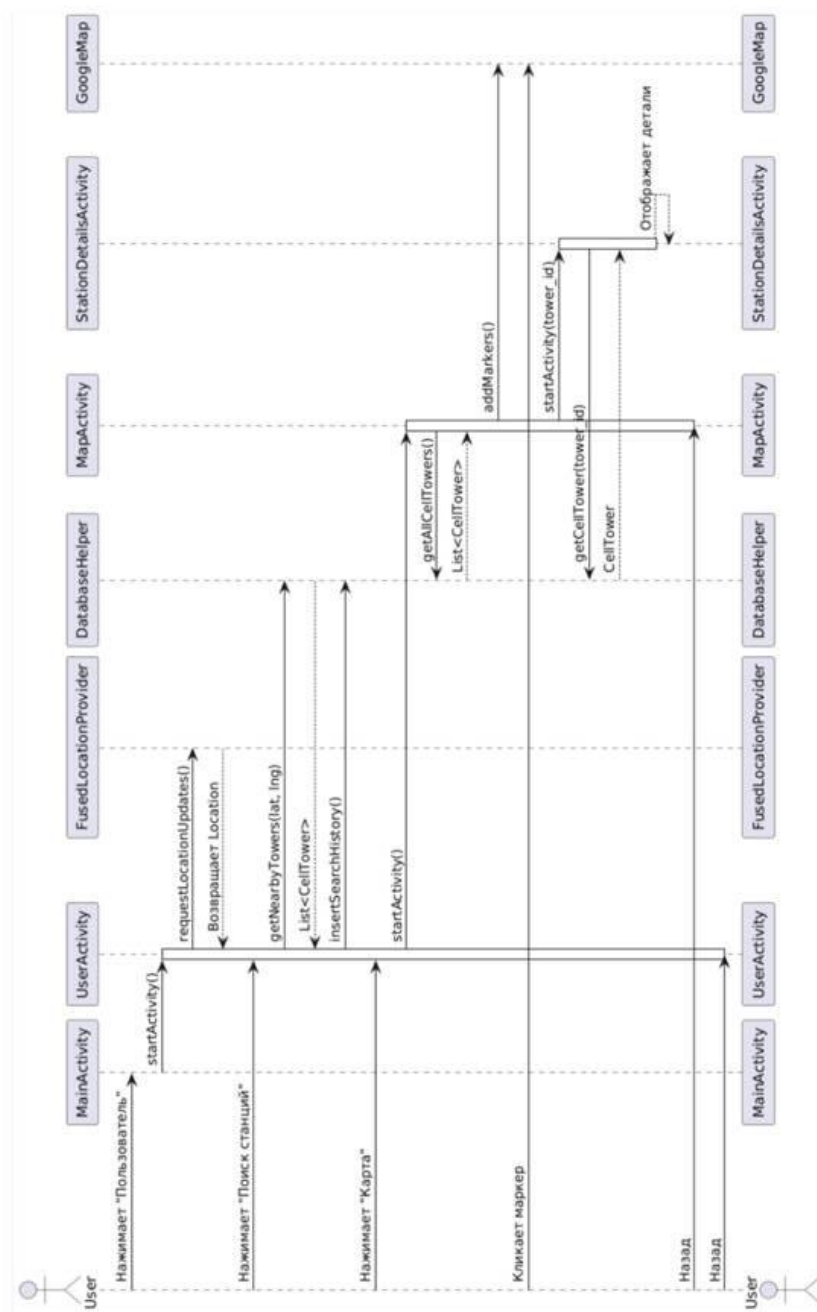
Основные функции приложения



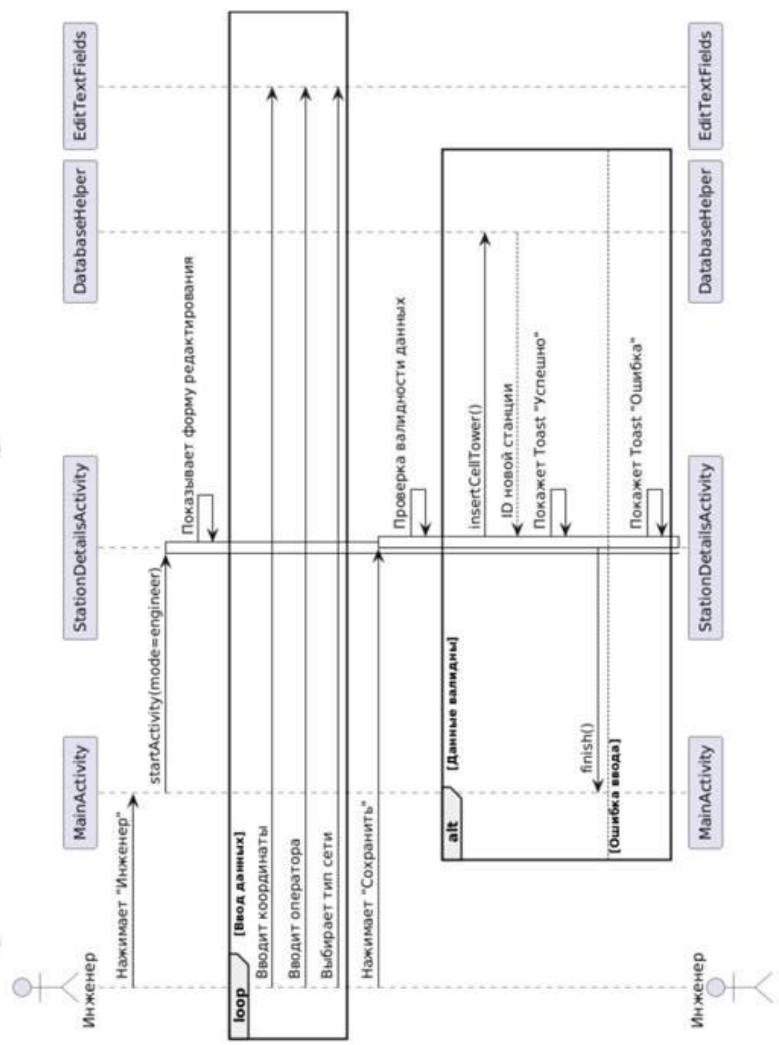
Архитектура приложения



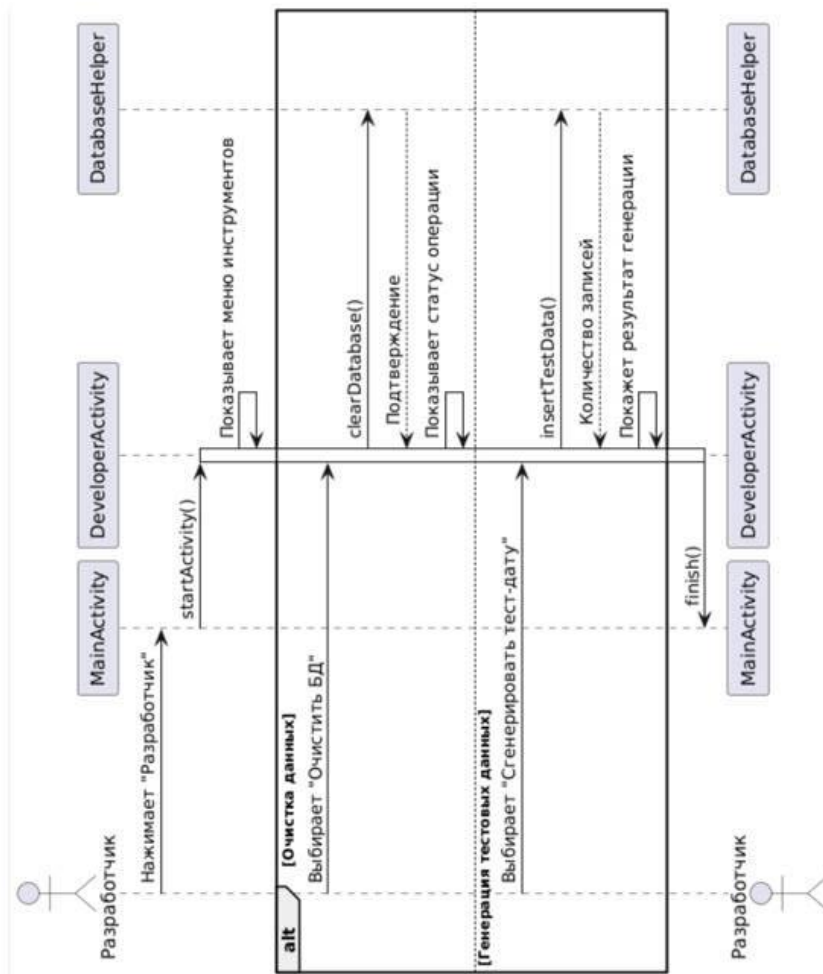
Работа приложения в режиме «Пользователь»



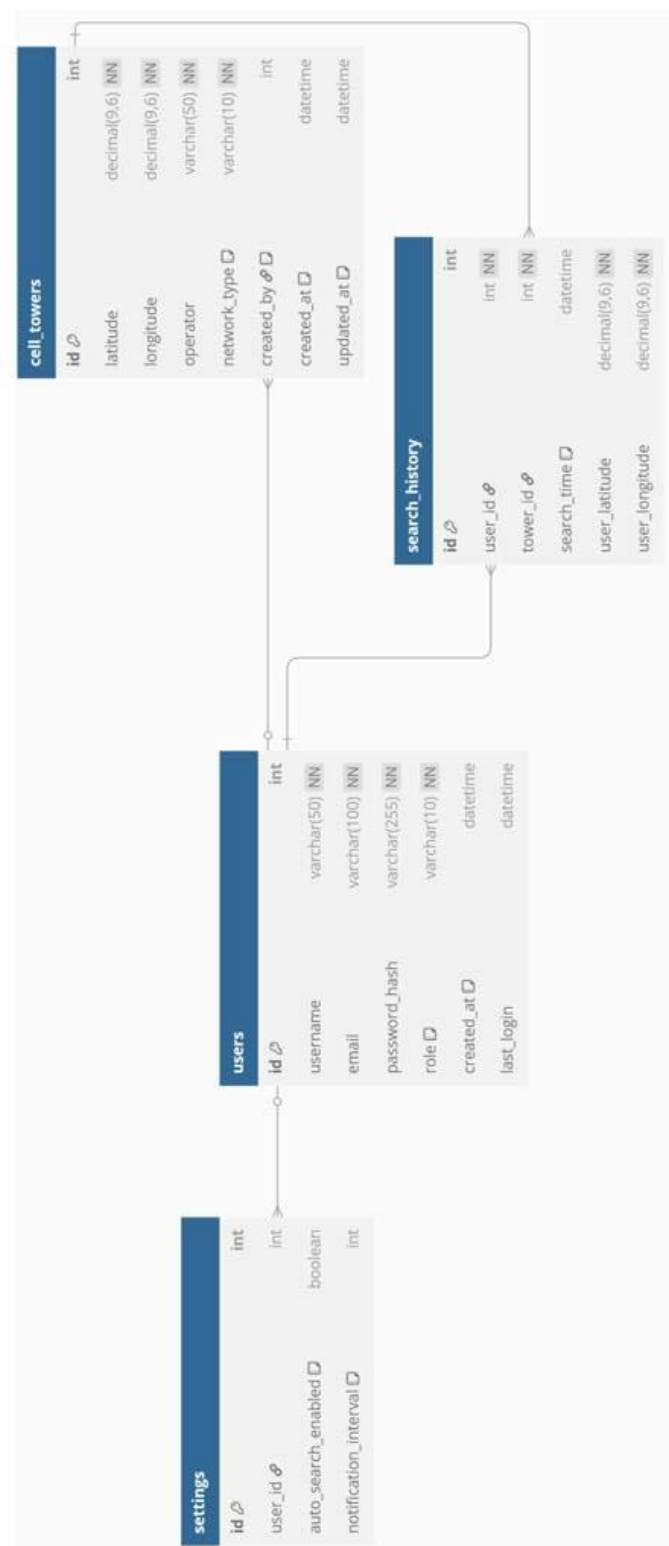
Работа приложения в режиме «Инженер»



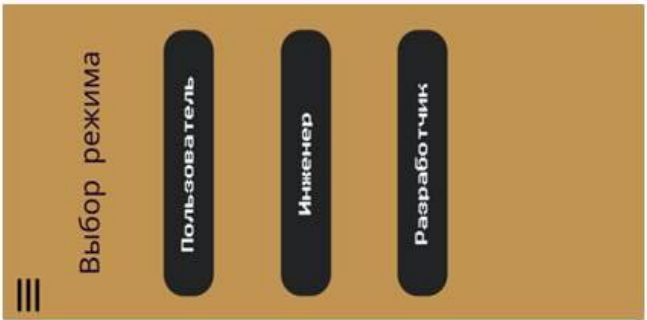
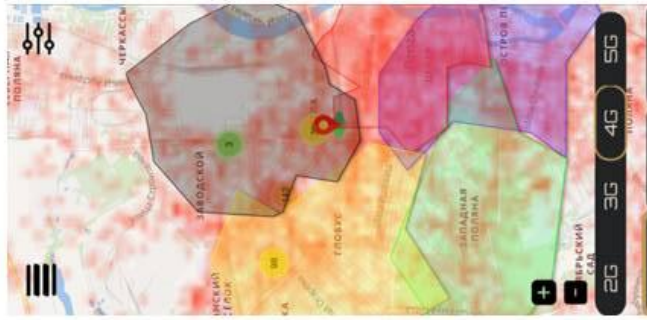
Работа приложения в режиме «Разработчик»



Структура базы данных



Скриншоты и демонстрация



Результаты работы

- Разработано мобильное приложение полностью соответствующее техническому заданию;
- Проведено функциональное тестирование (успешно);
- Намечены задачи по развитию функциональных возможностей программы.

Спасибо за внимание!

					ПГУ 1.090501.02.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		114