МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»

Кафедра телекоммуникационных систем и вычислительных средств (ТС и ВС)

РГЗ
по дисциплине
«Визуальное программирование»

по теме:
ПОЛУЧЕНИЕ И СОХРАНЕНИЕ ЛОКАЦИИ В ПРИЛОЖЕНИИ ДЛЯ СМАРТФОНА

Студент:
Группа ИА-331 *Д.В. Шкляев*

Предподаватель:
Старший преподаватель *Р.В. Ахпашев*

Новосибирск 2025 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы исследования обусловлена широким распространением мобильных приложений для смартфонов и возрастающей ролью геолокационных сервисов в различных областях: навигации, сервиса «умного» дома, фитнес-трекеров и аналитики пользовательского поведения. Современные платформы Android предоставляют развитые API для получения координат устройства [1; 2](например, FusedLocationProviderClient из Google Play Services), однако эффективное и корректное сохранение этих данных требует учета аспектов безопасности, работы с файловой системой и пользовательского интерфейса.

Цель работы — разработать компонент мобильного приложения на базе Jetpack Compose, обеспечивающий получение последней известной геопозиции устройства и её сохранение в локальный JSON-файл в публичной директории «Документы». Для достижения поставленной цели были решены следующие задачи:

1. Анализ существующих средств Android API для получения геолокации (ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION).
2. Реализация запроса и обработки пользовательских разрешений на доступ к геоданным.
3. Интеграция FusedLocationProviderClient в Jetpack Compose UI для асинхронного получения локации.
4. Организация хранения собранных координат с временными метками в формате JSON в файле `location_data.json`.
5. Построение экрана на Compose, отображающего историю сохранённых точек в виде списка карточек.
6. Тестирование корректности работы при различных состояниях разрешений и отсутствии данных.

Методологической основой исследования послужили принципы реактивного программирования в Compose, рекомендации по работе с разрешениями Android и требования ГОСТ 7.32-2017[3] к оформлению отчёта о научно-исследовательской работе.

ТЕОРИЯ

При разработке мобильных приложений для платформы Android одной из ключевых задач является получение и обработка геопозиционных данных устройства. Геолокация на Android базируется на двух основных подходах:

- **Использование системных провайдеров GPS и сети** через стандартный класс `android.location.LocationManager`. Данный подход обеспечивает детальное управление источниками (GPS, сотовая сеть, Wi-Fi), но требует сложной логики выбора оптимального провайдера, управления жизненным циклом запроса и значительных затрат ресурсов устройства.
- **Fused Location Provider API** из библиотеки Google Play Services (класс `FusedLocationProviderClient`). Абстрагирует детали разных провайдеров, автоматически выбирая наиболее точный и энергоэффективный источник в зависимости от условий и заданных параметров запроса.

Модель разрешений Android

Начиная с версии Android 6.0 (API 23), политика безопасности платформы перешла на модель «рантайм-разрешений». Для доступа к геоданным необходимо запросить у пользователя одно из разрешений:

- `ACCESS_FINE_LOCATION` — высокоточная локация (GPS, сеть).
- `ACCESS_COARSE_LOCATION` — приблизительная локация (сотовые вышки, Wi-Fi).

Приложение должно:

1. Проверить наличие разрешений через `ActivityCompat.checkSelfPermission()`.
2. При отсутствии — инициировать запрос с помощью `ActivityResultContracts.RequestMultiplePermissions`.
3. Обрабатывать результат и корректно реагировать на отказ пользователя.

Jetpack Compose и асинхронные операции

Jetpack Compose предлагает декларативный подход к построению UI на Kotlin[4; 5]. Для интеграции с API (такими как FusedLocationProviderClient[2]) в Compose используются:

- `rememberLauncherForActivityResult` — для запуска запросов разрешений и получения их результата внутри композиции.
- `LaunchedEffect` — для выполнения побочных эффектов при старте или изменении состояний.
- `mutableStateOf` и `remember` — для хранения и отслеживания состояния UI (например, списка сохранённых точек или статуса операции).

Формат и организация хранения данных

Для долговременного хранения точек геолокации используется файл в публичной директории `Environment.DIRECTORY_DOCUMENTS`. Выбор **JSON** обоснован:

- Читаемость и простота отладки.
- Широкая поддержка в стандартных библиотеках (`org.json.JSONArray`, `org.json.JSONObject`).
- Возможность форматирования с отступами для удобства ручного просмотра.

Структура файла:

```
[
  {
    "latitude": 55.7558,
    "longitude": 37.6173,
    "timestamp": "2025-05-19 14:23:10"
  },
  …
]
```

Методы работы с файлом:

1. `file.readText()` для чтения всего содержимого.
2. Создание или парсинг `JSONArray` для добавления новых записей.
3. Запись обратно с помощью `FileWriter` и форматированного вывода `toString(2)`.

Отображение истории точек в UI

Для визуализации сохранённых данных используется `LazyColumn` с элементами `Card`. Каждая карточка содержит:

– Временную метку (`timestamp`) в формате `yyyy-MM-dd HH:mm:ss`.
– Координаты: широта (`latitude`) и долгота (`longitude`).

Преимущества такого решения:

– Отложенная отрисовка элементов при большом числе записей.
– Возможность легко стилизовать и расширять карточку (иконки, кнопки, цветовые маркеры).
– Реактивное обновление списка при изменении состояния `entries`.

Безопасность и ограничения

При работе с внешней памятью важно учитывать:

– Разрешения на запись/чтение: `WRITE_EXTERNAL_STORAGE` (для старых версий) либо использование `Storage Access Framework`.
– Возможность отсутствия файловой системы (например, при отсутствии карты памяти).
– Обработка исключений для надёжного восстановления в случае ошибок I/O.

Таким образом, теоретическая база для получения, сохранения и отображения геопозиций в мобильном приложении объединяет в себе принципы работы Android API по локации, модель безопасности рантайм-разрешений, декларативный UI Jetpack Compose и практики организации JSON-хранилища.“‘

ПРАКТИЧЕСКАЯ ЧАСТЬ

В данной главе представлен анализ и описание реализации функционала получения и сохранения геолокации в Android-приложении на базе Jetpack Compose. Основой приложения служит файл `Location.kt`, в котором последовательно выполняются следующие шаги:

1. Запрос разрешений на доступ к геоданным (`ACCESS_FINE_LOCATION`, `ACCESS_COARSE_LOCATION`) с помощью `ActivityResultContracts.RequestMultiplePermissions`.
2. Получение последней известной локации через `FusedLocationProviderClient`.
3. Формирование JSON-объекта с полями `latitude`, `longitude` и `timestamp`[6].
4. Сохранение (дозапись) этого объекта в файл `location_data.json` в директории `Environment.DIRECTORY_DOCUMENTS`.
5. Загрузка всей истории точек из файла и отображение в списке `LazyColumn` с карточками (`Card`).

Основные компоненты и их взаимодействие

- **LocationActivity** (наследник `ComponentActivity`) задаёт тему приложения и запускает компоновку `LocationSaverScreen()`.
- **LocationSaverScreen**:
  - `rememberLauncherForActivityResult` для запроса прав.
  - `LaunchedEffect(Unit)` — начальная инициализация: загрузка существующих записей и запрос разрешений.
  - Кнопка `Get Location` запускает логику проверок прав, асинхронный вызов `fusedLocationClient.lastLocation` и сохранение результата.
- **LocationEntry** — дата-класс для хранения одной записи.
- **LocationCard** — компоновка карточки с выводом временной метки и координат.

Пример кода

Листинг 1 — Запись новой локации в файл

```
fusedLocationClient.lastLocation
    .addOnSuccessListener { location: Location? ->
        if (location != null) {
            val lat = location.latitude
            val lon = location.longitude
            val time = SimpleDateFormat(
                "yyyy-MM-dd HH:mm:ss",
                Locale.getDefault()
            ).format(Date())
            // Creating a JSON object
            val newObj = JSONObject().apply {
                put("latitude", lat)
                put("longitude", lon)
                put("timestamp", time)
            }
            // Reading an existing array or creating a new one
            val arr = try {
                JSONArray(file.readText())
            } catch (_: Exception) {
                JSONArray()
            }
            arr.put(newObj)
            // Writing it to an indented file
            FileWriter(file).use { it.write(arr.toString(2)) }
            statusText = "SAVED: \$time"
            entries = loadLocationsFromFile()
        } else {
            statusText = "Location is null"
        }
    }
    .addOnFailureListener { e ->
        statusText = "Location acquisition error: ${'$'}{e.
           message}"
    }
```

Тестирование и результаты

Для проверки корректности работы выполнены тесты на эмуляторе и реальном устройстве:

– Поведение при отказе в разрешениях: приложение повторно запрашивает права и отображает сообщение об ошибке.
– Сохранение нескольких точек: JSON-файл формируется верно, добавляются новые записи, и они отображаются в списке.
– Обработка состояния, когда `lastLocation == null`: выводится соответствующий текст.

В результате получен стабильный модуль, который можно подключить к любому Android-приложению для сбора и хранения геоданных пользователя.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была разработана и реализована компонента Android-приложения на базе Jetpack Compose для получения последней известной геопозиции устройства и её сохранения в формате JSON в публичной директории «Документы». Основные результаты:

– Успешно интегрирован FusedLocationProviderClient для получения координат.
– Реализована модель рантайм-разрешений Android (`ACCESS_FINE_LOCATION`, `ACCESS_COARSE_LOCATION`) с корректной обработкой отказа пользователя.
– Организовано долговременное хранение точек через JSON-файл (`location_data.json`) с возможностью дозаписи и форматированием.
– Обеспечен наглядный вывод истории локаций в `LazyColumn` с реактивным обновлением списка.

Практическая ценность полученного решения заключается в том, что данный модуль может быть легко встроен в любые приложения, требующие сбора и визуализации геоданных пользователей. При дальнейшем развитии проекта возможны следующие направления:

1. Добавление поддержки фонового отслеживания координат с использованием WorkManager или Foreground Service.
2. Интеграция с внешними сервисами картографирования (Google Maps, OpenStreetMap) для визуализации точек на карте.
3. Шифрование и безопасная передача данных на сервер для удалённого хранения и анализа.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Android Developers*. Sensors and location / Google. — 2025. — URL: https://developer.android.com/develop/sensors-and-location/location/retrieve-current?hl=ru.
2. *Google Play Services*. Fused Location Provider API / Google. — 2025. — URL: https://developers.google.com/location-context/fused-location-provider.
3. ГОСТ 7.32-2017. Отчет о научно-исследовательской работе. — МЕЖГОСУДАРСТВЕННЫЙ СОВЕТ ПО СТАНДАРТИЗАЦИИ, МЕТРОЛОГИИ И СЕРТИФИКАЦИИ (МГС), 2017.
4. *Android Developers*. Jetpack Compose — Declarative UI / Google. — 2025. — URL: https://developer.android.com/jetpack/compose.
5. *JetBrains*. Kotlin Language Documentation. — 2025. — URL: https://kotlinlang.org/docs/kotlin-tour-hello-world.html#practice.
6. *Android Developers*. org.json: JSON in Android / Google. — 2025. — URL: https://developer.android.com/reference/org/json/JSONArray.