Техническая документация Forest Research

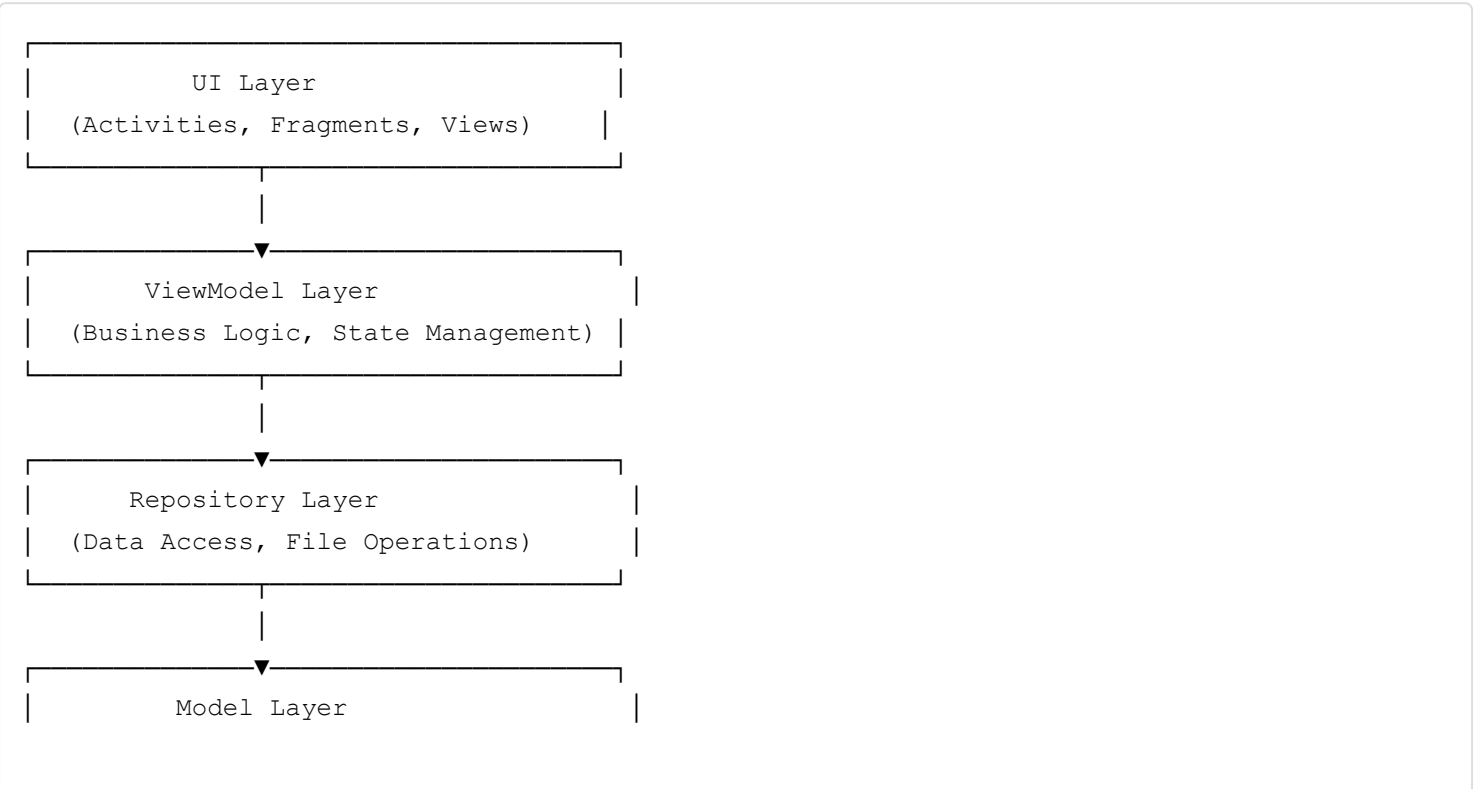
Содержание

Архитектура приложения

Паттерн архитектуры: MVVM (Model-View-ViewModel)

Приложение построено на архитектурном паттерне MVVM, который обеспечивает разделение ответственности и упрощает тестирование.

Слои архитектуры:



```
|   (Data Classes, Entities)            |
└──────────────────────────────────────┘
```

Компоненты архитектуры:

1. Application класс

- **`ForestResearchApplication`** — точка входа приложения
- Инициализация OpenCV
- Управление локализацией
- Централизованное управление контекстом

2. Базовые классы

- **`BaseFragment`** — базовый класс для всех фрагментов
- **`BaseViewModel`** — базовый класс для всех ViewModels
- Обеспечивают общую функциональность для наследников

3. Система навигации

- Использование `Activity` для основных экранов
- `ViewPager2` для вкладок формы описания точки
- Явные Intent'ы для перехода между экранами

Структура проекта

```
app/src/main/java/ru/ymike/forestresearch/
│
├── data/                           # Слой данных
│   ├── model/                      # Модели данных
│   │   ├── ForestPoint.kt          # Основная модель точки
│   │   ├── PointMetadata.kt        # Метаданные точки
│   │   ├── TreeLayer.kt            # Модель 1-го яруса (деревья)
│   │   ├── PlantLayer.kt           # Модель ярусов (2-й, 3-й)
│   │   ├── SoilData.kt             # Модель почвы
│   │   ├── ReliefData.kt           # Модель рельефа
│   │   ├── PhotoMetadata.kt        # Метаданные фотографий
│   │   ├── Geometry.kt             # Геометрия для GeoJSON
│   │   └── GeoJsonPoint.kt         # Модель GeoJSON точки
│   │
│   ├── repository/                 # Репозитории
│   │   └── PointRepository.kt      # CRUD операции с точками, управление файлами
│   │
│   └── Result.kt                   # Результат операций (Success/Error)
│
├── ui/                             # Слой представления
│   ├── base/                       # Базовые классы
```

```
│   │   ├── BaseFragment.kt
│   │   └── BaseViewModel.kt
│   │
│   ├── main/                        # Главный экран
│   │   ├── MainActivity.kt          # Список точек
│   │   └── PointAdapter.kt          # Адаптер списка
│   │
│   ├── form/                        # Форма описания точки
│   │   ├── PointFormActivity.kt     # Активность формы
│   │   ├── PointFormPagerAdapter.kt # Адаптер вкладок
│   │   │
│   │   ├── fragments/               # Фрагменты вкладок
│   │   │   ├── MetadataFragment.kt  # Основное
│   │   │   ├── TreeLayerFragment.kt # 1-й ярус
│   │   │   ├── ShrubLayerFragment.kt # 2-й ярус
│   │   │   ├── HerbLayerFragment.kt # 3-й ярус
│   │   │   ├── SoilFragment.kt      # Почва
│   │   │   ├── ReliefFragment.kt    # Рельеф
│   │   │   └── PhotoFragment.kt     # Фото
│   │   │
│   │   ├── adapters/
│   │   │   └── PhotoAdapter.kt      # Адаптер фотографий
│   │   │
│   │   └── fragments/               # ViewModels
│   │       ├── MetadataViewModel.kt
│   │       ├── PhotoViewModel.kt
│   │       └── PlantLayerViewModel.kt
│   │
│   ├── canopy/                      # Измерение сомкнутости
│   │   └── CanopyMeasurementActivity.kt
│   │
│   ├── map/                         # Карта
│   │   ├── MapActivity.kt           # Активность карты
│   │   ├── MapViewModel.kt          # ViewModel карты
│   │   ├── MapTileProvider.kt       # Провайдер тайлов карты
│   │   ├── BoundaryOverlayManager.kt # Управление границами ООПТ
│   │   ├── GeoJsonParser.kt         # Парсер GeoJSON
│   │   └── PointInfoWindow.kt       # Информационное окно точки
│   │
│   ├── photo/                       # Просмотр фотографий
│   │   └── FullscreenPhotoActivity.kt
│   │
│   └── dialogs/                     # Диалоги
│       └── ExportFilenameDialog.kt # Диалог экспорта
│
├── services/                        # Сервисы
│   ├── location/                    # GPS и локация
│   │   └── LocationManager.kt     # Менеджер локации
│   └── camera/                      # Работа с камерой
│
├── utils/                           # Утилиты
│   ├── Constants.kt                 # Константы (AppConstants)
│   ├── DataFieldIndices.kt          # Индексы массивов ресурсов
```

```
│   ├── ImageUtils.kt              # Утилиты изображений
│   ├── KMZExporter.kt             # Экспорт в KMZ
│   ├── KMZChecker.kt              # Проверка KMZ
│   ├── LocalePrefs.kt             # Управление локалью
│   ├── PermissionHelper.kt        # Помощник разрешений
│   ├── TextDrawableHelper.kt      # Создание текстовых drawable
│   └── Utils.kt                   # Общие утилиты
│
└── ForestResearchApplication.kt   # Application класс
```

Внутренние API и контракты

Модели данных

ForestPoint

```
data class ForestPoint(
    val id: String,                    // Уникальный ID точки
    val name: String,                  // Название точки
    val description: String,           // Описание
    val metadata: PointMetadata,        // Метаданные (GPS, дата и т.д.)
    val relief: ReliefData?,           // Данные рельефа
    val photos: List<PhotoMetadata>,   // Список фотографий
    val soil: SoilData?,               // Данные почвы
    val herbLayer: PlantLayer?,        // 3-й ярус (травы)
    val shrubLayer: PlantLayer?,       // 2-й ярус (подлесок)
    val treeLayer: TreeLayer?          // 1-й ярус (деревья)
)
```

Методы:

- `toFlatGeoJson(): Map<String, Any>` — создание GeoJSON с плоской структурой (новый формат)
- `toFlatGeoJsonProperties(): Map<String, Any?>` — создание плоской структуры properties
- `fromFlatGeoJson(properties: Map<String, Any?>, geometry: Geometry): ForestPoint?` — парсинг плоской структуры (companion object)
- `longitude: Double` — получение долготы
- `latitude: Double` — получение широты

Примечание: Старый метод `toGeoJson(): String` помечен как `@Deprecated` и сохраняется только для обратной совместимости.

PointMetadata

```
data class PointMetadata(
    val coordinates: Pair<Double, Double>, // (широта, долгота)
    val elevation: Double?,                // Высота над уровнем моря
```

```
    val gpsAccuracy: Float?,                // Точность GPS (метры)
    val creationDate: String,               // Дата создания (ISO формат)
    val project: String?,                   // Проект
    val operator: String?,                  // Оператор
    val notes: String?                      // Заметки
)
```

TreeLayer (1-й ярус)

```
data class TreeLayer(
    val composition: String?,           // Формула древостоя
    val canopyClosure: Int?,            // Сомкнутость крон (%)
    val avgHeight: Int?,                // Средняя высота (см)
    val avgDiameter: Int?,              // Средний диаметр (см)
    val deadTrees: Int?,                // Процент сухостоя
    val crownConditions: CrownCondition // Состояние крон (enum)
)
```

PlantLayer (2-й и 3-й ярусы)

```
data class PlantLayer(
    val totalCoverage: Int?,            // Общее покрытие (%)
    val avgHeight: Int?,                // Средняя высота (см)
    val species: String?                // Видовой состав
)
```

Репозитории

PointRepository

Конструктор:

```
PointRepository(context: Context)
```

Основные методы:

```
// Получение всех точек (поддерживает плоскую и вложенную структуру)
fun getAllPoints(): List<ForestPoint>

// Сохранение точки (в плоском формате GeoJSON)
fun savePoint(point: ForestPoint)

// Загрузка точки по ID
fun getPointById(pointId: String): ForestPoint?

// Удаление точки
fun deletePoint(pointId: String): Boolean
```

```
// Получение файла точки
fun getPointFile(pointId: String): File

// Получение файла фотографии
fun getPhotoFile(pointId: String, angle: PhotoAngle): File
```

Контракты:

- Возвращает пустой список при ошибках чтения
- Создает директории автоматически при первом обращении
- Использует Gson для сериализации/десериализации

Примечание: Актуальная реализация работы с файлами находится в `PointRepository`.

ViewModels

PhotoViewModel

Методы:

```
fun initialize(context: Context, pointId: String?)
fun loadPhotos()
fun handlePhotoTaken(file: File, angle: PhotoAngle)
fun deletePhoto(angle: PhotoAngle)
fun getPhotoFileForCapture(angle: PhotoAngle): File?
fun getPhotosForSave(): List<PhotoMetadata>
fun hasPhotoForAngle(angle: PhotoAngle): Boolean
fun getPhotoUri(context: Context, angle: PhotoAngle): Uri?
```

Состояние:

```
val photosState: StateFlow<PhotosState>
val isLoading: StateFlow<Boolean>
val error: StateFlow<String?>
```

MapViewModel

Методы:

```
fun loadPoints()
fun getPoints(): LiveData<List<ForestPoint>>
```

Форматы данных

GeoJSON формат точек

Каждая точка сохраняется в отдельный файл `.geojson` в формате GeoJSON Feature с **плоской структурой** полей для оптимальной работы в QGIS.

Структура файла:

```
{
  "geometry": {
    "coordinates": [долгота, широта],
    "type": "Point"
  },
  "properties": {
    "point_id": "20251102_205052",
    "point_name": "Название точки",
    "project": "Проект",
    "operator": "Оператор",
    "description": "Описание",
    "notes": "Примечания",
    "latitude": 56.65970575606069,
    "longitude": 29.346620327711555,
    "elevation_m": 219.55,
    "gps_accuracy_m": 4.9,
    "creation_date": "2025-11-02 20:50:52",
    "stand_composition": "5Е3Ос1С1Ол",
    "canopy_closure_percent": 63,
    "mean_tree_height_m": 25,
    "mean_tree_diameter_cm": 45,
    "snag_percentage": 5,
    "crown_conditions": "GOOD,OPPRESSED",
    "understory_total_cover_percent": 30,
    "understory_mean_height_m": 2,
    "understory_species": "Corylus:15,Salix:10",
    "ground_total_cover_percent": 60,
    "ground_mean_height_cm": 30,
    "ground_species": "Oxalis:25,Equisetum:20",
    "soil_composition": "LOAM,CLAY",
    "soil_moisture": "FRESH",
    "litter_type": "LEAF,MIXED",
    "bare_soil_cover_percent": 10,
    "deadwood_cover_percent": 5,
    "deadwood_decomposition": "FRESH",
    "terrain_form": "FLAT",
    "slope_aspect": "",
    "slope_steepness": "",
    "photo_canopies": "Photos/20251102_205052_UP.jpg",
    "photo_general_view": "Photos/20251102_205052_STRAIGHT.jpg",
    "photo_litter_and_soil": "Photos/20251102_205052_DOWN.jpg",
    "photo_additional": "Photos/20251102_205052_EXTRA.jpg"
  },
  "type": "Feature"
}
```

Ключевые особенности структуры:

1. **Плоская структура properties:** Все 35 полей находятся на верхнем уровне (без вложенных объектов)
    - Преимущества для QGIS: все поля видны в атрибутивной таблице, удобная фильтрация и группировка
    - Порядок полей соответствует порядку вкладок формы для удобства пользователя
2. **Английские названия полей:** Универсальные названия для совместимости с международными стандартами
3. **Порядок полей в GeoJSON:** `geometry` → `properties` → `type` (для совместимости с некоторыми версиями QGIS)
4. **Координаты:**
    - В `geometry.coordinates`: `[долгота, широта]` (стандарт GeoJSON)
    - В `properties`: также сохраняются как `latitude` и `longitude` для удобства работы в атрибутивной таблице
    - Координаты НЕ дублируются в `properties.metadata.coordinates` (в отличие от старой версии)
5. **Фотографии:**
    - Хранятся с относительными путями (`Photos/...`) для работы в QGIS при копировании папки с проектом
    - Четыре поля: `photo_canopies`, `photo_general_view`, `photo_litter_and_soil`, `photo_additional`
    - Каждое поле содержит путь к фото соответствующего ракурса или пустую строку
6. **Обработка null значений:**
    - Пустые строки `""` вместо `null` для текстовых полей
    - `-1` вместо `null` для `canopy_closure_percent` (маркер "не измерено")
    - `0` для числовых полей без значения

Полный список полей (35):

- **Метаданные (11):** `point_id`, `point_name`, `project`, `operator`, `description`, `notes`, `latitude`, `longitude`, `elevation_m`, `gps_accuracy_m`, `creation_date`
- **Overstory / Древостой (6):** `stand_composition`, `canopy_closure_percent`, `mean_tree_height_m`, `mean_tree_diameter_cm`, `snag_percentage`, `crown_conditions`
- **Understory / Подрост (3):** `understory_total_cover_percent`, `understory_mean_height_m`, `understory_species`
- **Ground / Травяной покров (3):** `ground_total_cover_percent`, `ground_mean_height_cm`, `ground_species`
- **Soil / Почва (6):** `soil_composition`, `soil_moisture`, `litter_type`, `bare_soil_cover_percent`, `deadwood_cover_percent`, `deadwood_decomposition`

- **Relief / Рельеф (3):** `terrain_form`, `slope_aspect`, `slope_steepness`
- **Photos / Фотографии (4):** `photo_canopies`, `photo_general_view`, `photo_litter_and_soil`, `photo_additional`

Формат множественных значений:

- Виды растений: `"Corylus:15,Salix:10"` (формат: `название:покрытие%`)
- Перечисления (например, `crown_conditions`, `soil_composition`): `"GOOD,OPPRESSED"` (через запятую)

Совместимость:

- Приложение поддерживает чтение как новой (плоской), так и старой (вложенной) структуры для обратной совместимости
- Метод `getAllPoints()` автоматически определяет формат и парсит соответствующим образом

Важные детали для разработчиков:

- Сериализация: `ForestPoint.toFlatGeoJson()` создает плоскую структуру
- Десериализация: `ForestPoint.fromFlatGeoJson()` парсит плоскую структуру обратно в объект
- Используется `LinkedHashMap` для сохранения порядка полей при сериализации

KMZ формат экспорта

KMZ — это ZIP-архив, содержащий KML файл и связанные ресурсы (фотографии).

Структура KMZ:

```
point_name.kmz
├── doc.kml                  # Основной KML файл
└── files/
    ├── photo_up.jpg
    ├── photo_straight.jpg
    ├── photo_down.jpg
    └── photo_extra.jpg
```

KML структура:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <name>Forest Research Export</name>
    <Folder>
      <name>Точка 1</name>
      <Placemark>
        <name>Название точки</name>
        <description>
```

```
          <!-- HTML описание со всеми данными -->
        </description>
        <Point>
          <coordinates>долгота,широта,0</coordinates>
        </Point>
      </Placemark>
      <!-- Фотографии как дополнительные Placemark'и -->
    </Folder>
  </Document>
</kml>
```

Совместимость:

- Google Earth (полная поддержка)
- QGIS (импорт KMZ)
- ArcGIS (импорт KMZ)

Формат фотографий

Расположение:

```
/Documents/Forest Research/Photos/
  {point_id}_UP.jpg
  {point_id}_STRAIGHT.jpg
  {point_id}_DOWN.jpg
  {point_id}_EXTRA.jpg
```

Свойства:

- Формат: JPEG (`.jpg`)
- Качество сжатия: 85%
- Максимальный размер: 5 MB
- Автоматическая обработка: сжатие и поворот при необходимости

Метаданные фотографий:

```
data class PhotoMetadata(
    val file: File,
    val angle: PhotoAngle,        // UP, STRAIGHT, DOWN, EXTRA
    val absolutePath: String? = null,
    val relativePath: String? = null
)
```

Алгоритм измерения сомкнутости крон

Техническое описание

Алгоритм использует компьютерное зрение (OpenCV) для автоматического анализа изображения неба через кроны деревьев.

Этапы обработки кадра:

1. **Преобразование YUV → HSV**
    - Камера предоставляет изображение в формате YUV_NV21
    - Конвертация: YUV → RGB → BGR correction → HSV
    - Используется для анализа цветовых характеристик
2. **Выделение V-канала для алгоритма Оцу**
    - Извлекается V (Value/Brightness) канал из HSV
    - Применяется алгоритм Оцу для бинарной сегментации
    - Результат: темные области (тени/ветки) и светлые (небо)
3. **Классификация пикселей**
    - **Темные (черные):** `binaryValue == 0` (по алгоритму Оцу)
    - **Зеленые:** `!isDark && (hue in hueMin..hueMax && saturation >= saturationMin)`
    - **Светлые (белые):** остальные пиксели
4. **Расчет сомкнутости**

```
Сомкнутость = ((темные_пиксели + зеленые_пиксели) / общее_количество_пикселей) * 100%
```

5. **Визуализация**
    - Создается цветовая маска:
        - Черный — стволы/ветки
        - Зеленый — кроны
        - Белый — просветы неба
    - Поворот на 270° + отражение для корректного отображения фронтальной камеры

Параметры настройки:

- **Hue Min / Hue Max:** Диапазон оттенка для распознавания зеленых крон
    - OpenCV использует диапазон 0-179 (вместо 0-360)
    - Рекомендуемые значения: 40-70
- **Saturation Min:** Минимальная насыщенность для признания зеленым
    - Диапазон: 0-255
    - Рекомендуемое значение: 30
- **Pixel Step:** Шаг выборки пикселей для оптимизации
    - Значения: 1-4 (1 = максимальная точность, 4 = максимальная скорость)
    - Влияет на время обработки кадра

Оптимизации производительности:

- **Пропуск кадров:** Обработка каждого N-го кадра (skipFrames = 2)
- **Блочная обработка:** Обработка изображения блоками pixelStep x pixelStep
- **Прямое вычисление координат:** Избежание lookup-таблиц
- **Управление памятью:** Автоматическое освобождение Mat и Bitmap

Математические детали:

Преобразование HSV:

```
OpenCV Hue: 0-179 → Android Hue: 0-360
androidHue = opencvHue * 360 / 179
```

Алгоритм Оцу:

- Автоматически определяет оптимальный порог для бинарной сегментации
- Основан на максимизации межклассовой дисперсии

Расчет координат при повороте:

```
Поворот на 270° + отражение:
(x, y) → (height-1-y, width-1-x)
```

Используемые библиотеки

Основные библиотеки

OpenCV 4.11.0

- **Назначение:** Обработка изображений для измерения сомкнутости крон
- **Использование:**
  - Преобразование цветовых пространств (YUV → RGB → HSV)
  - Алгоритм Оцу для бинарной сегментации
  - Работа с матрицами изображений (Mat)
- **Инициализация:** `OpenCVLoader.initDebug()` в `ForestResearchApplication`
- **Лицензия:** Apache License 2.0

osmdroid 6.1.18

- **Назначение:** Оффлайн карты
- **Использование:**
  - Отображение MBTiles карт
  - Навигация по карте

    - Отрисовка маркеров и оверлеев
- **Лицензия:** Apache License 2.0

CameraX 1.3.0

- **Назначение:** Работа с камерой
- **Компоненты:**
    - `camera-core` — базовый функционал
    - `camera-camera2` — Camera2 API
    - `camera-lifecycle` — управление жизненным циклом
- **Использование:** Только ImageAnalysis (без Preview) для оптимизации
- **Лицензия:** Apache License 2.0

Gson 2.10.1

- **Назначение:** Сериализация/десериализация JSON
- **Использование:**
    - Сохранение точек в GeoJSON
    - Загрузка точек из файлов
- **Лицензия:** Apache License 2.0

Glide 4.16.0

- **Назначение:** Загрузка и кэширование изображений
- **Использование:**
    - Отображение фотографий в списке
    - Просмотр фотографий в полноэкранном режиме
- **Лицензия:** Apache License 2.0

ThreeTenABP 1.4.6

- **Назначение:** Работа с датами и временем
- **Использование:** Форматирование дат для метаданных
- **Лицензия:** Apache License 2.0

Google Play Services Location 21.0.1

- **Назначение:** GPS и определение местоположения
- **Использование:** Получение координат точек исследования
- **Лицензия:** Proprietary (Google)

AndroidX библиотеки

- **androidx.core:core-ktx:1.12.0** — Kotlin расширения
- **androidx.appcompat:appcompat:1.6.1** — Совместимость

- **androidx.recyclerview:recyclerview:1.4.0** — Списки
- **androidx.viewpager2:viewpager2:1.0.0** — Вкладки формы
- **androidx.fragment:fragment-ktx:1.6.1** — Фрагменты

Material Design

- **com.google.android.material:material:1.9.0** — Material Design компоненты

Системы хранения данных

Файловая структура

Внутреннее хранилище приложения (управляется автоматически)

```
/Android/data/ru.ymike.forestresearch/files/
└── osmdroid/                    # Рабочая папка для карт (автоматически синхронизируется)
    ├── Russia_0-7.mbtiles       # Базовая карта (стандартная или пользовательская)
    ├── {user_basemap}.mbtiles   # Пользовательская базовая карта (если есть)
    ├── {detailed_map1}.mbtiles  # Детальные карты (скопированы из /maps/)
    └── ...
```

Важно: Эта папка синхронизируется автоматически при каждом запуске карты. Не рекомендуется изменять вручную.

Внешнее хранилище (Documents) - Пользовательская папка

```
/Documents/Forest Research/
├── Points/                      # Точки исследования (GeoJSON)
│   ├── {point_id}.geojson
│   ├── {point_id}.geojson
│   └── ...
│
├── Photos/                      # Фотографии
│   ├── {point_id}_UP.jpg
│   ├── {point_id}_STRAIGHT.jpg
│   ├── {point_id}_DOWN.jpg
│   ├── {point_id}_EXTRA.jpg
│   └── ...
│
├── basemap/                     # Пользовательская базовая карта (НЕОБЯЗАТЕЛЬНО)
│   └── {custom_base}.mbtiles  # Если пуста, используется стандартная Russia_0-7.mbtiles
│
├── maps/                        # Детальные карты (MBTiles)
│   ├── Shorsky_map.mbtiles
│   ├── Remdovsky_map.mbtiles
│   └── ...
│
```

```
├── borders/                    # Границы ООПТ (GeoJSON)
│   ├── boundary1.geojson
│   ├── boundary2.geojson
│   └── ...
│
└── *.kmz                       # Экспортированные KMZ файлы (в корне папки)
    ├── export1.kmz
    ├── export2.kmz
    └── ...
```

Важно:

- Вся папка `/Documents/Forest Research/` предназначена для пользователя. Все файлы в этой папке можно изменять, добавлять и удалять. Приложение обрабатывает все эти сценарии автоматически.
- **Детальные карты из `/maps/` используются напрямую без копирования в `osmdroid`**, что экономит память устройства и избегает дублирования данных.

SharedPreferences

locale_prefs

- **Ключ:** `user_locale`
- **Значение:** `"ru"` или `"en"`
- **Управление:** `LocalePrefs` утилита

canopy_settings

- **Ключи:**
  - `hue_min` — минимальный оттенок
  - `hue_max` — максимальный оттенок
  - `saturation_min` — минимальная насыщенность
  - `pixel_step` — шаг пикселей
- **Использование:** Настройки измерения сомкнутости

Константы и настройки

AppConstants

Расположение: `utils/Constants.kt`

ViewPager настройки

```
VIEWPAGER_OFFSCREEN_LIMIT = 6
TAB_COUNT = 7
```

GPS и локация

```
GPS_MIN_TIME_MS = 1000L          // Минимальное время между обновлениями
GPS_MIN_DISTANCE_M = 1f          // Минимальное расстояние
GPS_GOOD_ACCURACY_M = 10f        // Хорошая точность
GPS_EXCELLENT_ACCURACY_M = 5f    // Отличная точность
```

Карта

```
DEFAULT_LATITUDE = 55.7558       // Москва (по умолчанию)
DEFAULT_LONGITUDE = 37.6176
DEFAULT_ZOOM_LEVEL = 10f
MAP_ZOOM_MIN = 0.0
MAP_ZOOM_MAX = 20.0
BASE_MAP_NAME = "Russia_0-7.mbtiles"
```

Базовая карта:

- Стандартная карта находится в assets приложения: `assets/Russia_0-7.mbtiles`
- При запуске карты метод `syncBasemapToOsmdroid()` сканирует папку `/Documents/Forest Research/basemap/`
- **Логика синхронизации:**
  - Если в `/basemap/` есть `.mbtiles` файл(ы) — копируется в `osmdroid` и становится базовой картой (пользовательская карта в приоритете)
  - Если `/basemap/` пуста — копируется `Russia_0-7.mbtiles` из assets в `osmdroid`
  - Если в `/basemap/` несколько карт — используется самая свежая по дате модификации, показывается предупреждение пользователю
- **Замена базовой карты пользователем:**
  - Пользователь помещает свой `.mbtiles` файл в `/Documents/Forest Research/basemap/` (любое имя)
  - При следующем запуске карты: файл копируется в `osmdroid`, стандартная `Russia_0-7.mbtiles` удаляется
- **Восстановление стандартной карты:**
  - Пользователь удаляет все `.mbtiles` файлы из `/Documents/Forest Research/basemap/`
  - При следующем запуске карты: `/basemap/` пуста → копируется `Russia_0-7.mbtiles` из assets в `osmdroid`

Детальные карты:

- Хранятся в `/Documents/Forest Research/maps/`
- При запуске карты метод `copyUserMapsToOsmdroid()` копирует их в `osmdroid` для работы с картой
- Метод `scanMapsDirectory()` находит все `.mbtiles` файлы в `/maps/`
- Пользователь может свободно добавлять, изменять и удалять карты в `/maps/` — при следующем запуске произойдет синхронизация с `osmdroid`

- Удаленные карты автоматически удаляются из `osmdroid` при следующей синхронизации

Камера и фото

```
CAMERA_SKIP_FRAMES = 3           // Пропуск кадров для оптимизации
PHOTO_QUALITY = 85               // Качество JPEG (0-100)
MAX_PHOTO_SIZE_MB = 5            // Максимальный размер фото
```

Валидация данных

```
MIN_NAME_LENGTH = 3
MAX_NAME_LENGTH = 50
MIN_PROJECT_LENGTH = 2
MAX_PROJECT_LENGTH = 100
```

Сообщения об ошибках

```
ERROR_PERMISSION_DENIED = "Недостаточно разрешений..."
ERROR_LOCATION_UNAVAILABLE = "Геолокация недоступна"
ERROR_FILE_WRITE_FAILED = "Ошибка записи файла"
ERROR_INVALID_STATE = "Некорректное состояние приложения"
ERROR_UNKNOWN = "Неизвестная ошибка"
```

DataFieldIndices

Расположение: `utils/DataFieldIndices.kt`

Определяет индексы для массивов ресурсов (arrays.xml), используемых в UI.

Группы:

- `Tab` — индексы вкладок
- `LayerName` — индексы названий ярусов
- `PhotoAngle` — индексы ракурсов фотографий
- `Metadata` — индексы полей метаданных
- `TreeLayer` — индексы полей 1-го яруса
- `CrownCondition` — индексы состояний крон
- `Soil`, `SoilComposition`, `SoilMoisture`, `LitterType` — индексы почвы
- `Relief`, `TerrainForm`, `Aspect`, `SlopeSteepness` — индексы рельефа

Разрешения приложения

Объявленные разрешения

В `AndroidManifest.xml`:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERNET" />
```

Использование разрешений

- **ACCESS_FINE_LOCATION:** Определение GPS-координат точек
- **CAMERA:** Съемка фотографий и измерение сомкнутости
- **READ/WRITE_EXTERNAL_STORAGE:** Сохранение данных и доступ к картам (для Android < 11)
- **INTERNET:** Теоретически не используется (приложение полностью оффлайн, но может требоваться для некоторых системных функций)

Запрос разрешений

- **Runtime разрешения:** Запрашиваются при первом использовании функции
- **Управление:** `PermissionHelper` утилита
- **Обработка отказа:** Показывается сообщение пользователю с инструкциями

Локализация

Поддерживаемые языки

- **Русский (ru)** — основной язык
- **Английский (en)** — дополнительный язык

Система управления локалью

Компоненты:

1. **ForestResearchApplication:**
    - Централизованное управление локалью
    - Метод `getAppLocale()` — получение текущей локали
    - Метод `setUserLocale()` — установка языка пользователем
    - LiveData событие `localeChanged` для обновления UI
2. **LocalePrefs:**
    - Сохранение выбранного языка в SharedPreferences

- Ключ: `user_locale`

3. **Переключение языка:**
   - Кнопка языка на главном экране (FAB)
   - Автоматическое пересоздание Activity при смене языка

Приоритет определения языка:

1. Сохранённый пользователем язык (SharedPreferences)
2. FORCE_LOCALE (для отладки, если установлен)
3. Язык устройства (ru → ru, en → en, иначе → en)
4. Дефолт "en"

Ресурсы локализации

- **Строки:** `res/values/strings.xml` (ru), `res/values-en/strings.xml` (en)
- **Массивы:** `res/values/arrays.xml`

Технические детали

Версии SDK

- **Минимальный SDK:** 26 (Android 8.0 Oreo)
- **Target SDK:** 35 (Android 15)
- **Compile SDK:** 35

Язык программирования

- **Kotlin:** Версия 17 (JVM Target)
- **Java Compatibility:** Java 17

Build-инструменты

- **Gradle:** Версия из `gradle-wrapper.properties`
- **Android Gradle Plugin:** 8.13.0 (из `libs.versions.toml`)
- **Kotlin Plugin:** 2.0.21

Features

- **View Binding:** Включен (`buildFeatures { viewBinding = true }`)
- **Data Binding:** Не используется

Производительность

Оптимизации:

- **Измерение сомкнутости:**
  - Пропуск кадров (skipFrames = 2)
  - Блочная обработка (pixelStep)
  - Прямое вычисление координат (без lookup-таблиц)
  - Управление памятью (автоматическое освобождение Mat/Bitmap)
- **Карта:**
  - Оффлайн работа (без интернета)
  - Кэширование тайлов
  - Оптимизированная отрисовка границ
- **Фотографии:**
  - Автоматическое сжатие
  - Glide кэширование
  - Ленивая загрузка

Безопасность

- Данные хранятся локально на устройстве
- Нет передачи данных в интернет
- Разрешения запрашиваются только при необходимости

Заключение

Эта документация описывает техническую архитектуру и внутреннее устройство приложения Forest Research. Для пользовательской документации см. USER_GUIDE.md.

Последнее обновление: Версия 1.0