

ZandarIS Kotlin – Architecture

System Overview

ZandarIS Kotlin is an Android application used by municipal police officers in Bratislava for field enforcement operations. It is the mobile client of the ZandarIS information system and communicates exclusively with the NestJS backend at <https://zandar.is.staging.bratislava.sk/api/>.

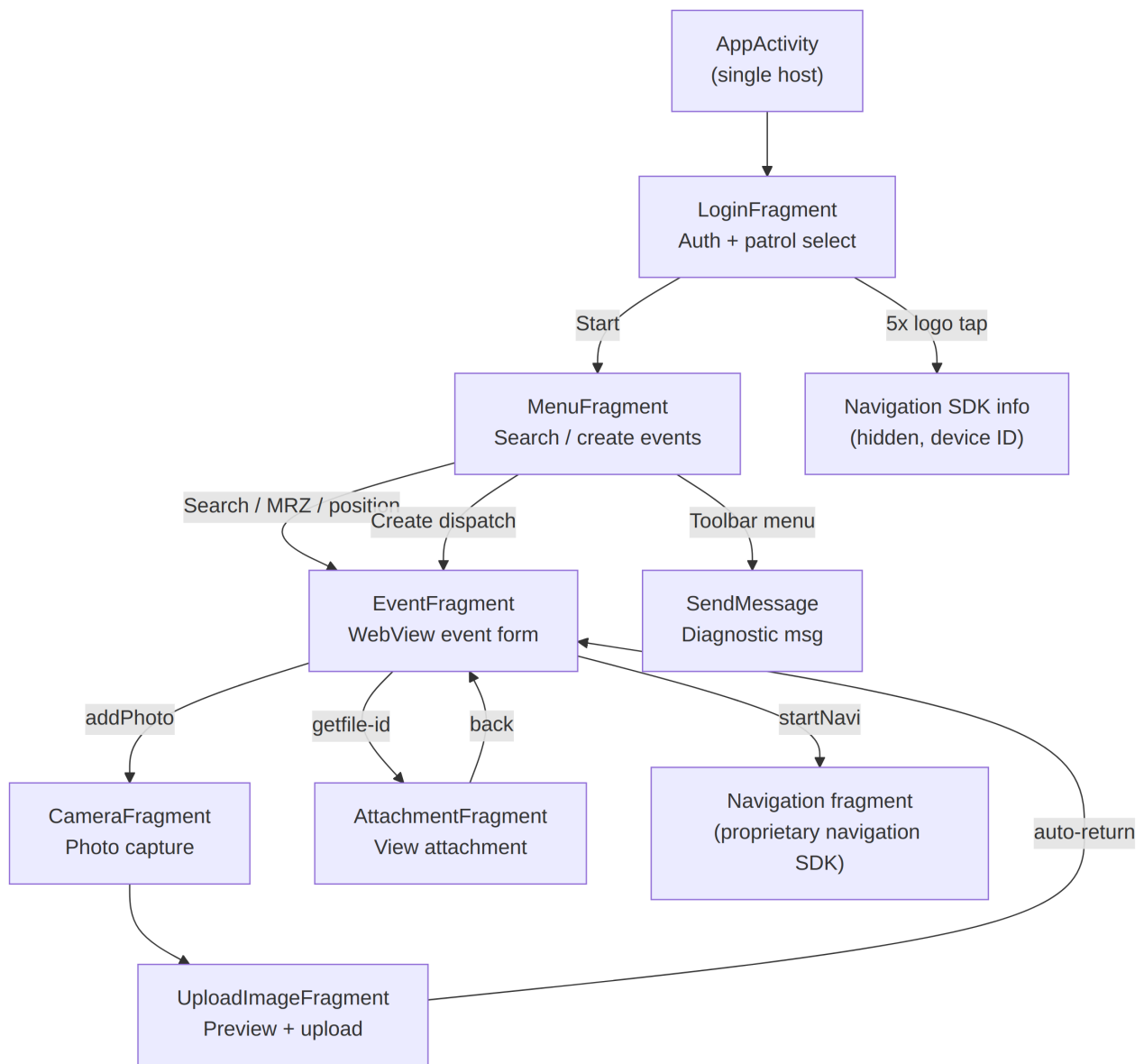
The app enables officers to:

- Authenticate via Azure AD and select their patrol
- Search and create enforcement events
- Capture and upload evidence photos with GPS/timestamp overlay
- View event details and attachments via an embedded WebView form
- Navigate to event locations using a proprietary navigation SDK
- Scan MRZ identity documents (via DESKO USB reader) to look up offenders
- Report their patrol position in real time

Tech Stack

Layer	Technology
Language	Kotlin 1.9
Min SDK	29 (Android 10) / Target SDK 33
Architecture	Single-Activity + Fragments (no ViewModel/MVVM)
HTTP	Volley 1.2.1
Auth	MSAL 5.6.0 (Azure AD)
Navigation	Proprietary navigation SDK
Camera	CameraX 1.3.4
Location	Google Play Services Location 21.3.0
JSON	Gson 2.11.0
MRZ parsing	Proprietary OCR SDK

Navigation Flow



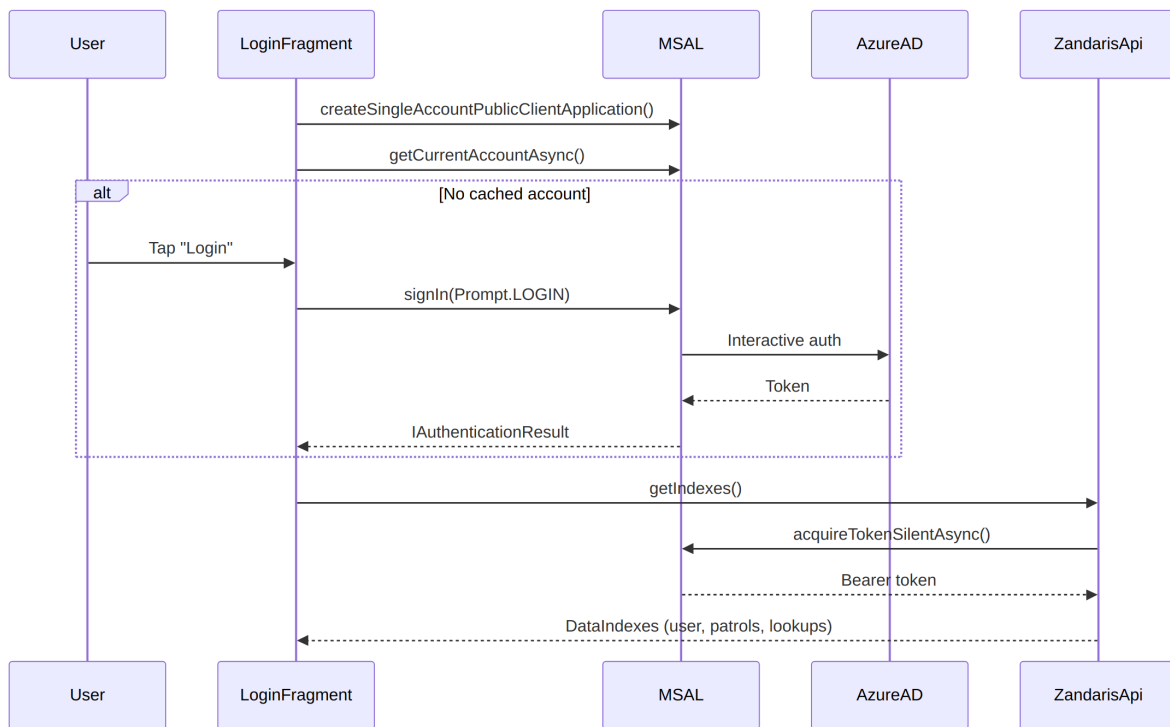
Navigation Flow

Authentication (MSAL / Azure AD)

Authentication uses Microsoft Authentication Library (MSAL) with a single-account public client:

- **Tenant:** f5fe6d23-b843-40d9-b7ec-a40b571fca59
- **Client ID:** 8ab61c7d-d5f4-4dc9-9bec-c15f90639bb8
- **Scope:** api://8ab61c7d-d5f4-4dc9-9bec-c15f90639bb8/access_as_user
- **Config:** auth_config_debug.json (debug) / auth_config_release.json (release) in res/raw/

Auth Flow



Auth Flow

Every API call in ZandarisiApi silently acquires a fresh token via `acquireTokenSilentAsync()` before adding it as a Bearer header.

WebView Token Bridge

The EventFragment WebView loads the backend's web event form. The native app injects ZandarisiApi as a `@JavascriptInterface` named `zandardroid`. The web form calls:

- `zandardroid.getIndexes()` – returns cached index JSON
- `zandardroid.getTokenSilent()` – synchronously acquires a bearer token so the web form can make authenticated API calls

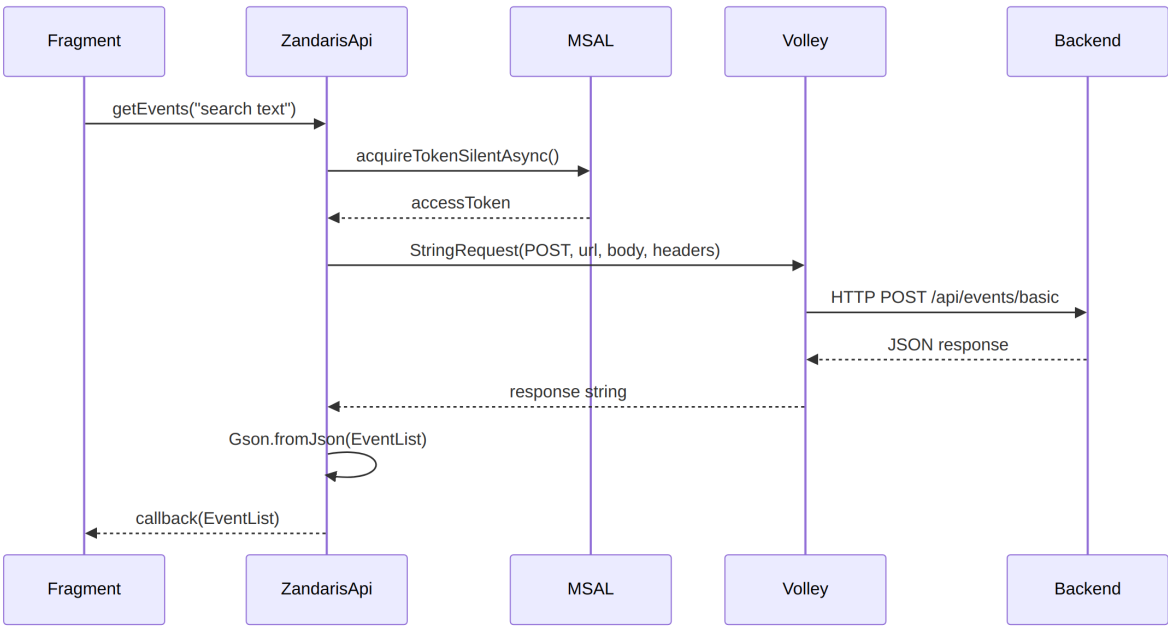
API Consumption

All HTTP communication goes through ZandarisiApi, which uses Volley with a 30-second timeout and MSAL bearer tokens.

Endpoint Catalog

Method	Endpoint	Purpose
POST	home/senddiag	Send diagnostic/crash report to Discord
GET	home/poscheck?posx=&posy=	Reverse-geocode GPS to nearby streets
POST	home/patrol-position	Report patrol GPS position
GET	events/{eventId}	Fetch event details
GET	events/getfile/{fileId}	Download attachment binary
POST	events/basic	Search events by free text
POST	events/advanced	Search events by person identity
GET	indexes?validOnly=true	Fetch all reference/lookup data
POST	events/attach	Upload photo (multipart)
GET	events/getEventAddress/{eventId}	Get event address for navigation
POST	events	Create a new blank event
PATCH	events/fields	Bulk-update event fields

Request Lifecycle



Request Lifecycle

WebView Bridge Protocol

EventFragment loads the event form at {basicUrlEvent}/{eventId} in a MyWebView. The web form communicates back to native code via zandaris-app:// custom URLs, intercepted by MyWebViewClient:

URL	Action
zandaris-app://getfile-{fileId}	Open AttachmentFragment to view/download the file
zandaris-app://finishEvent	Pop the back stack (return to menu)
zandaris-app://startNavi	Fetch event address, start navigation (proprietary navigation SDK)
zandaris-app://addPhoto	Open CameraFragment to capture a photo

Additionally, MyWebView intercepts DESKO MRZ scanner keyboard input, parses it with the proprietary OCR SDK and dispatches the identity data to the web page as a barcodescan JavaScript CustomEvent.

Third-Party Integrations

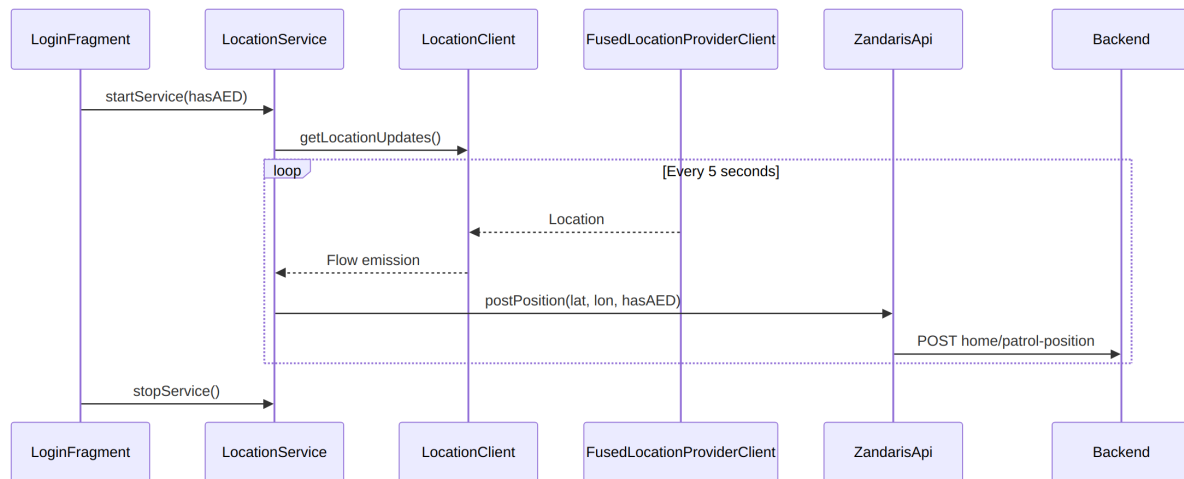
CameraX

- **Version:** 1.3.4
- **Purpose:** Evidence photo capture in CameraFragment
- **Features:** Back camera, flash toggle, slider zoom, pinch-to-zoom, rotation correction
- **Output:** Bitmap passed to UploadImageFragment for preview and upload

Google Play Services Location

- **Version:** 21.3.0
- **Purpose:** GPS position tracking for patrol location reporting
- **Clients:**
 - LocationClient (Flow-based, 5s interval, high accuracy) – used by LocationService
 - Direct FusedLocationProviderClient calls in MenuFragment and UploadImageFragment for one-shot position

Location Tracking



Location Tracking

LocationService runs as a foreground service with a low-priority notification. It is started when the officer taps “Start” on LoginFragment and stopped when they return to the login screen or the app is destroyed.

Build and Signing

- **Build system:** Gradle 8.9 with Kotlin DSL
- **Android Gradle Plugin:** 8.7.3
- **Namespace:** sk.bratislava.zandaris
- **Version:** 1.18 (versionCode 18)
- **Compile SDK:** 34 / **Min SDK:** 29 / **Target SDK:** 33

Signing

Both debug and release builds require signing. Keystore configuration is loaded from keystores/keystore.properties:

```
debug.storeFile=...
debug.storePassword=...
debug.keyAlias=...
debug.keyPassword=...
release.storeFile=...
release.storePassword=...
release.keyAlias=...
release.keyPassword=...
```

Keystore files and properties are not committed to git.

Repositories

The project uses additional Maven repositories for:

- Azure Duo SDK (MSAL transitive dependency)
- Proprietary navigation SDK