Android Stuff

- For L10n Training
  - □□ | Localization Bait Inventory: Training "Cheat Sheet" for "Notes" Android Native App

For L10n Training

For L10n Training

? | Localization Bait Inventory: Training "Cheat Sheet" for "Notes" Android Native App

This document lists all the deliberate localization "traps" implemented in the Notes app. Use this to guide students in identifying why hard-coding is problematic and how to properly implement Android L10n.

Project Github Repo here.

---

1. Hard-coded UI Strings (The "Low Hanging Fruit")

- **Location:** All screens (`AboutScreen.kt`, `NoteListScreen.kt`, `NoteEditorScreen.kt`, `SettingsScreen.kt`)
- **The Bait:** Plain text strings like `"About Notes"`, `"Save"`, `"Search Notes…"`, and `"Delete"`.
- **The Lesson:** Basic translation. These must be moved to `res/values/strings.xml` so the system can swap them based on the user's language.

2. The Plurals Trap (Manual Logic)

- **Location:** `NoteListScreen.kt`
- **The Bait:** `text = if (notesCount == 1) "1 note" else "$notesCount notes"`
- **The Lesson:** English only has two plural forms (one/other), but languages like Arabic have six, and Russian has three. Hard-coded `if/else` logic cannot scale.
- **Solution:** Refactor to use **`res/values/plurals.xml`**.

3. Date & Time Formatting

- **Location:** `NoteListScreen.kt` (inside `NoteItem`)
- **The Bait:** `SimpleDateFormat("MM/dd/yyyy HH:mm", Locale.US)`
- **The Lesson:** Date order (MM/DD vs DD/MM) and clock formats (12h vs 24h) are regional. Forcing `Locale.US` makes the app feel "foreign" in most of the world.
- **Solution:** Use `java.text.DateFormat` formatters that respect the system locale.

4. Text Embedded in Assets (The "Icon Trap")

- **Location:** `res/drawable/ic_app_icon.xml`
- **The Bait:** The vector paths draw the literal English characters **"N-O-T-E-S"** and **"L10N"**.
- **The Lesson:** When an app name is translated, an icon containing English text becomes misleading.
- **Solution:** Either use symbolic icons (no text) or use **Resource Qualifiers** (e.g., `drawable-zh/`) to swap assets.

5. String Concatenation & Templates

- **Location:** `AboutScreen.kt`
- **The Bait:** `text = "Version " + "1.0.0"`
- **The Lesson:** In some languages, the word for "Version" comes *after* the number. Concatenation locks the word order.
- **Solution:** Use string resources with placeholders: `Version %s`.

6. Logic Keys as Display Strings

- **Location:** `NotesViewModel.kt` (Enums and Tags)
- **The Bait:** Tags like `"Work"`, `"Personal"`, and `"Important"` are used as both database keys and UI labels.
- **The Lesson:** Translating a UI label that is also used as a logic key will break the app's functionality.
- **Solution:** Separate **internal IDs** (stable) from **display labels** (localized).

7. Cultural Sorting & Search

- **Location:** `NotesViewModel.kt`
- **The Bait:** `it.title.contains(query, ignoreCase = true)` and `"Title (A-Z)"` sorting.
- **The Lesson:** Alphabetical order varies by script. Simple case-insensitivity fails for accented characters (like `é` vs `E`).
- **Solution:** Use `java.text.Collator` for locale-aware sorting.

8. UI Expansion & Layout Breaking

- **Location:** `NoteListScreen.kt` (Grid Mode)
- **The Bait:** The 2-column grid layout assumes English-length text.
- **The Lesson:** German and French words are often 30% longer than English.
- **Challenge:** Students must ensure the UI doesn't "break" or overlap when words expand.