23. Juni 2025

SRH HOCHSCHULE HEIDELBERG

Random User App

PROJEKTBERICHT

PRAKTISCHE ARBEIT VON

SADE HOSKINS

MATRIKELNUMMER: 11037064

STUDIENGANG: VIRTUAL REALITY & GAME DEVELOPMENT (BA)

SCHWERPUNKT: PROGRAMMING

FACHSEMESTER: 4

MODUL: AR, VR & MOBILE DEVELOPMENT

ART DER ABGABE: PRAKTISCHE ARBEIT

ABGABE-DATUM: 23.06.2025

PRÜFER: PROF. DR. ANDREAS JÄGER

FELIX DÖRSCHNER

Inhaltsverzeichnis

1 Einleitung

Dieser Bericht dokumentiert die Entwicklung einer mobilen Applikation, die die Verwaltung und Darstellung zufälliger Benutzerdaten ermöglicht. Ausgangspunkt des Projekts war die Problemstellung, eine benutzerfreundliche und performante Anwendung zu entwickeln, die sowohl online als auch offline funktioniert und moderne Android-Entwicklungsmethoden nutzt. Ziel war es, eine intuitive Lösung zu schaffen, die Daten aus einer Web-API abruft, lokal speichert und zusätzliche Funktionen wie QR-Code-Generierung und Scannen bietet.

Der Projektbericht umfasst eine Zusammenfassung der Kernanforderungen, eine technische Analyse der Code-Umsetzung und Designentscheidungen. Um die Lesbarkeit des Codes zu gewährleisten, wurden detaillierte Kommentare im Code ergänzt.

Alternativer Repository Link:

Die Anwendung, einschließlich der vollständigen Dokumentation, der Readme-Datei und des Quellcodes, ist unter folgendem Repository-Link verfügbar: (https://github.com/sadehoskins/MobileApplicationSH). Er kann direkt über GitHub heruntergeladen werden.

Zusammenfassung der Implementierung

Kernanforderungen

Die Anwendung nutzt eine SQLite-Datenbank, die mit der Room-Persistenzbibliothek realisiert wurde. Die Integration der Web-API von *randomuser.me* wurde erfolgreich umgesetzt. Vier voll funktionsfähige Bildschirme – Übersicht, Detailansicht, AR/Kamera und Einstellungen – wurden entwickelt, einschließlich einer intuitiven Navigation. Die Generierung und das Echtzeit-Scannen von QR-Codes sind vollständig implementiert. Zudem unterstützt die App eine unbegrenzte Anzahl an Benutzer*innen, wobei alle Daten in der Datenbank gespeichert werden.

Erweiterte Funktionen

Die App bietet Offline-Datenpersistenz mit automatischer Synchronisation. Eine Echtzeitsuche mit Filtermöglichkeiten wurde integriert, sodass Benutzer*innen Daten effizient nach Namen, Standort oder Erstellungsdatum sortieren können. Umfassende Fehlerbehandlung sorgt für eine stabile Nutzung, während Werkzeuge für die Datenbankverwaltung und Debugging-Features die Wartung erleichtern.

Technische Umsetzung

Architektur

Die Anwendung basiert auf einer sauberen Architektur (*Clean Architecture*) und nutzt das *MVVM-Muster* (Model-View-ViewModel) in Kombination mit einer Persistenz-Ebene, um eine klare Trennung der Verantwortlichkeiten zu gewährleisten. Diese Struktur erhöht die Wartung und sorgt für eine übersichtliche Codebasis. Reaktive Benutzeroberflächen werden durch *StateFlow* und *Flow* ermöglicht, die dynamisch auf Zustandsänderungen reagieren.

Für die UI-Entwicklung wurde *Jetpack Compose* gewählt, ein modernes Framework, das durch seinen deklarativen Ansatz sowie die nahtlose Integration mit *ViewModels* und Statusmanagement überzeugt. Das Design orientiert sich an den Richtlinien von *Material Design 3* und wird durch eine individuelle Farbpalette ergänzt. Die Lebenszyklusverwaltung wurde optimiert, um Speicher effizient zu nutzen und Lecks zu vermeiden.

Die Wahl von *Room* als Datenbanklösung bietet typensicheren Zugriff, Unterstützung für *Coroutines* und *Flow* sowie automatische Migrationen. Zudem wurde eine QR-Code-Lösung implementiert, die *ZXing* für die Generierung und *ML Kit* für das Echtzeit-Scanning kombiniert, ergänzt durch eine Parsing-Logik zur Verarbeitung von QR-Inhalten.

Code-Struktur

- **Datenebene:** Konfiguriert mit Retrofit (ApiClient), Room (UserDao) und einem zentralen Repository (UserRepository) zur Datenverwaltung. Modelle werden über UserMapper konvertiert.
- **UI-Ebene:** Enthält klar strukturierte Bildschirmkomponenten, wiederverwendbare Elemente, ein individuelles Design-System und einen Navigationsgraphen.
- **Logik:** UserViewModel koordiniert die Statusverwaltung, während QR-Code-Operationen und Datenparsing von spezialisierten Klassen unterstützt werden. Fehlerbehebungsstrategien stellen benutzerfreundliches Feedback sicher.

Leistungsoptimierungen

- **Datenbank:** Effiziente Abfragen mit *Indizes, Lazy Loading* großer Datensätze und asynchrone Verarbeitung über *Coroutines*.
- **UI:** Optimiertes Bildladen mit *Coil*, effizientes Statusmanagement und performantes Listen-Rendering mit *LazyColumn*.
- **Speicher:** Lebenszyklusbewusste *ViewModels*, automatische Freigabe von Kameraressourcen und effiziente Bitmap-Verarbeitung.

Teststrategie

Die Anwendung wurde mit Blick auf umfassende Testbarkeit entwickelt. Das Repository-Muster ermöglicht einfaches *Mocking* von Datenquellen, *ViewModels* sind unabhängig von der UI testbar und Datenbankoperationen können mit In-Memory-Datenbanken geprüft werden. Simulierte Antworten erleichtern die Überprüfung von API-Interaktionen.

Zukünftige Verbesserungen

Geplante Erweiterungen

Die Unterstützung für XML-API-Formate konnte aus Zeitgründen nicht umgesetzt werden. Umfassende Unit- und Integrationstests sowie eine für Tablets optimierte Detail-Ansicht sind in Planung.

Architektonische Überlegungen

Die Integration von *Hilt* soll die Testbarkeit verbessern. Zudem wird der Code modularisiert, um größere Entwicklungsprojekte zu erleichtern. Langfristig wird eine *Offline-First-Architektur* mit zuverlässigen Synchronisationsmechanismen angestrebt.

Schlussfolgerung

Die Random User App veranschaulicht moderne Ansätze der Android-Entwicklung und erfüllt sämtliche Kernanforderungen. Die Umsetzung überzeugt durch eine klare Architektur, effiziente Datenverwaltung und eine intuitive Benutzeroberfläche. Dank ihrer modularen Struktur ist die App bestens für zukünftige Erweiterungen gerüstet und garantiert eine hohe Codequalität.

Quellen

Literaturverzeichnis

Android Developer Webseite. https://developer.android.com/?hl=de (letzter Zugriff: 23.06.2025)

Github Repository Random User App von Sade Hoskins sadehoskins/MobileApplicationSH (letzter Zugriff: 23.06.2025)

Abhängigkeiten

Core Framework

- Jetpack Compose BOM: 2023.10.01
- Activity Compose: 1.8.1
- Navigation Compose: 2.7.6
- Lifecycle ViewModel Compose: 2.7.0

Database

- Room Runtime: 2.6.1
- Room KTX: 2.6.1
- Room Compiler: 2.6.1 (annotation processor)

Networking

- Retrofit: 2.9.0
- Gson Converter: 2.9.0
- OkHttp Logging Interceptor: 4.12.0

Camera and QR Code Processing

- CameraX Bundle: 1.3.1
- ML Kit Barcode Scanning: 17.2.0
- ZXing Core: 3.5.2
- ZXing Android Embedded: 4.3.0

Additional Libraries

- Coil Compose: 2.5.0
- Accompanist Permissions: 0.32.0
- Coroutines: 1.7.3