Slides Progetto LAM 2023/2024

Matteo Canghiari

Informatica per il Management
Università di Bologna

Gennaio 27, 2025

Contents

Scelte di base

Nel corso dello sviluppo dell'applicazione sono state effettuate diverse scelte progettuali, quali:

- Uso di Kotlin
- Design pattern architetturale MVVM
- Jetpack compose per animazioni durante runtime
- Supabase service come SSOT, di cui memorizzata una copia in locale
- Bucket di AWS per condivisione dei dati tra utenti iscritti all'applicazione

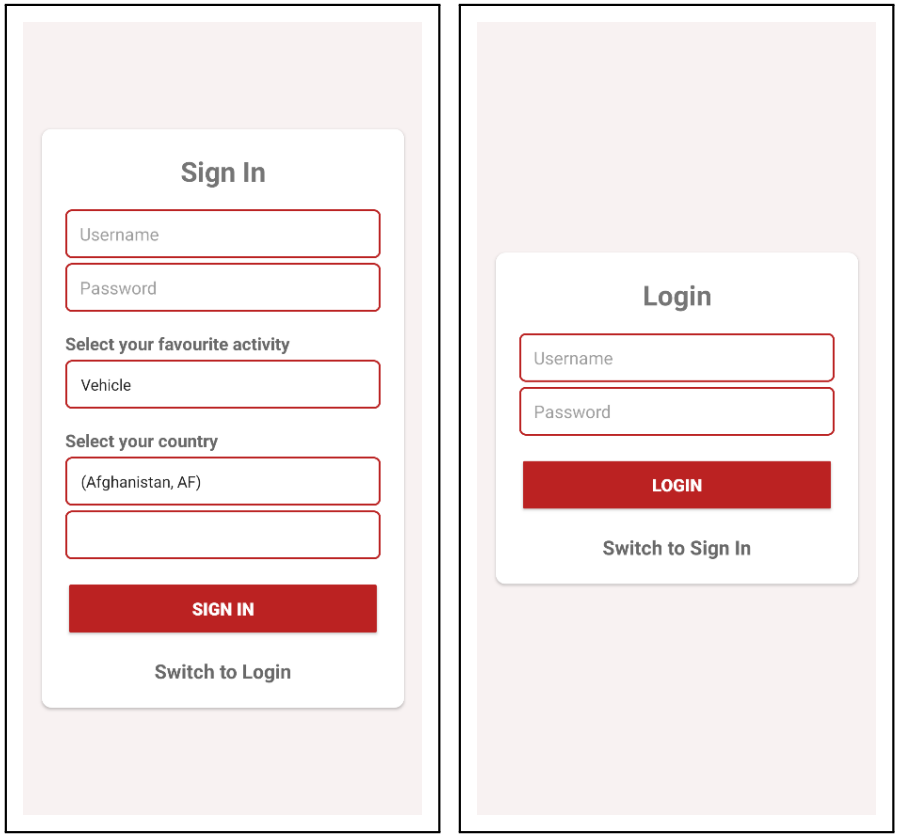
Divisione e struttura del codice

La suddivisione del progetto avviene in differenti componenti, ognuna delle quali è destinata nella propria sub-directory, suddivise in:

- activity
- fragment
- page
- receiver
- room
- service
- singleton
- viewModel
- worker

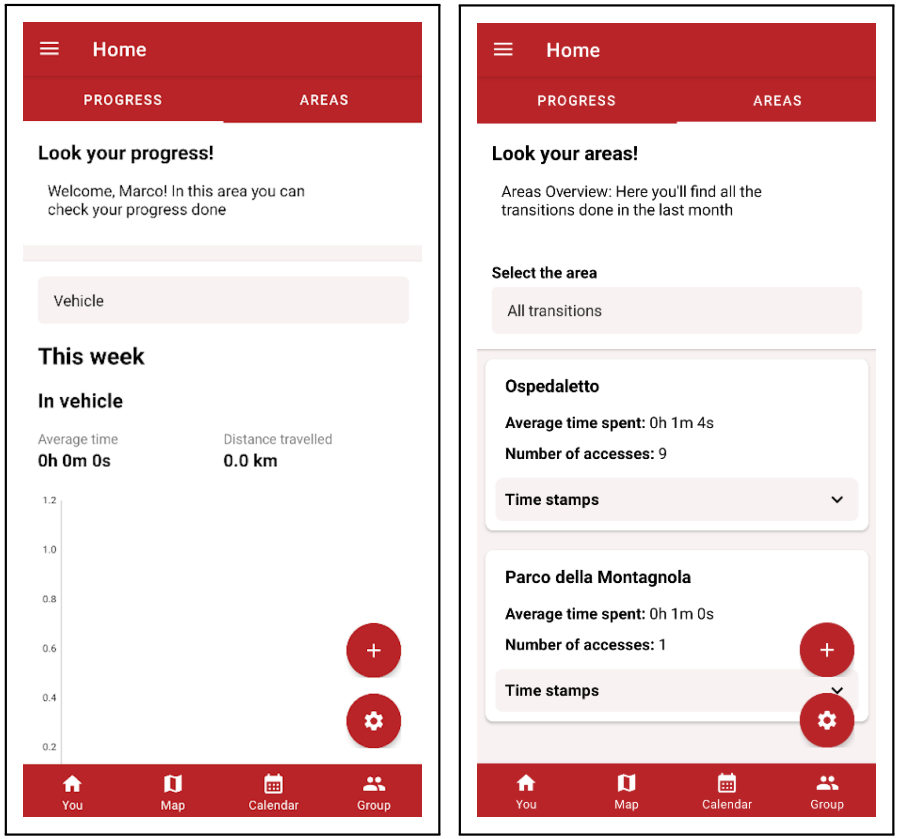
Registrazione & Login

- **Prime schermate** visualizzabili all'avvio dell'applicazione
- Controllo di **connessione** persistente del dispositivo
- Check di **unicità** delle credenziali di accesso
- **Reindirizzamento** autonomo alla Home dell'applicazione qualora l'accesso sia stato già effettuato



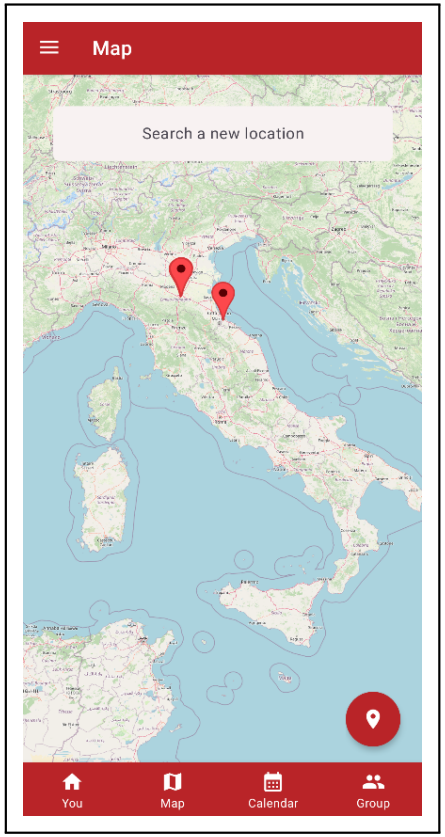
Home

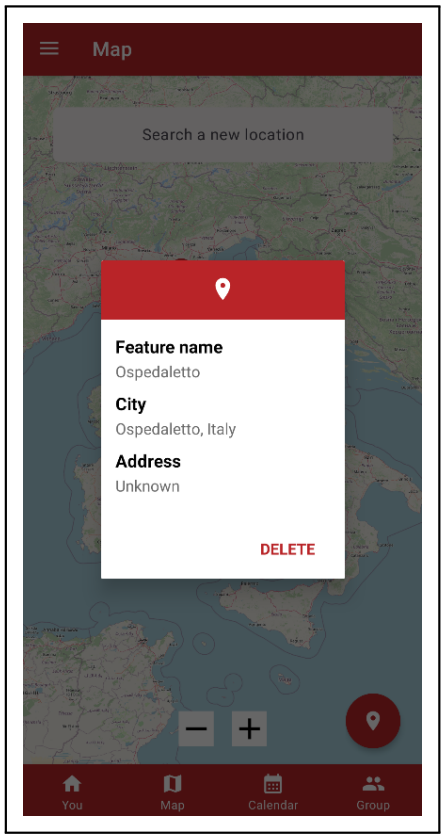
- Visualizzazione della sezione personale tramite **TabLayout**
- **Progress page** ideata per evidenziare i progressi ottenuti durante la registrazione, manuale o autonoma, dell'attività motorie
- **Areas page** visualizzazione delle transizioni avvenute durante un certo arco temporale



Map

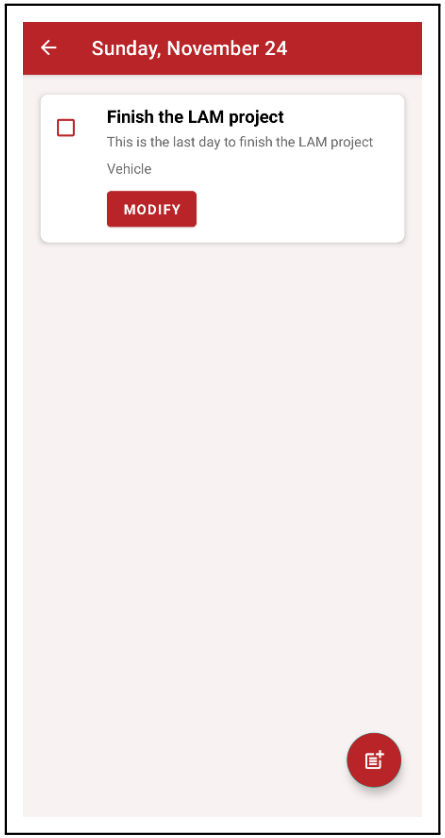
- Mappa implementata tramite **Osmdroid** piuttosto che Google Maps
- Visualizzazione delle **aree geografiche** personali
- Aggiunta e rimozione delle aree geografiche tramite **SearchBar** e **Dialog**
- Floating button per marcare la **posizione corrente**





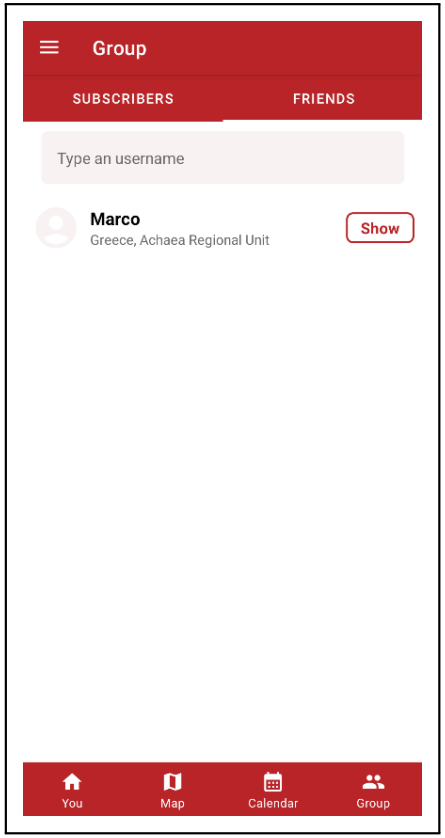
Calendar

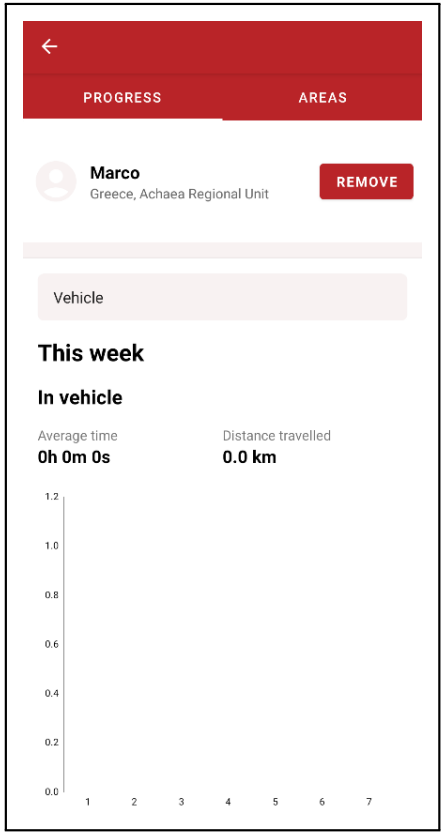
- Visualizzazione schematica per l'organizzazione delle **proprie attività**
- Per ciascuna data selezionata sono mostrati tutti i **Memo** correnti
- Ciascun Memo può essere modificato oppure eliminato
- Tramite il floating button è garantito l'accesso al **form di creazione** di nuovi Memo



Group

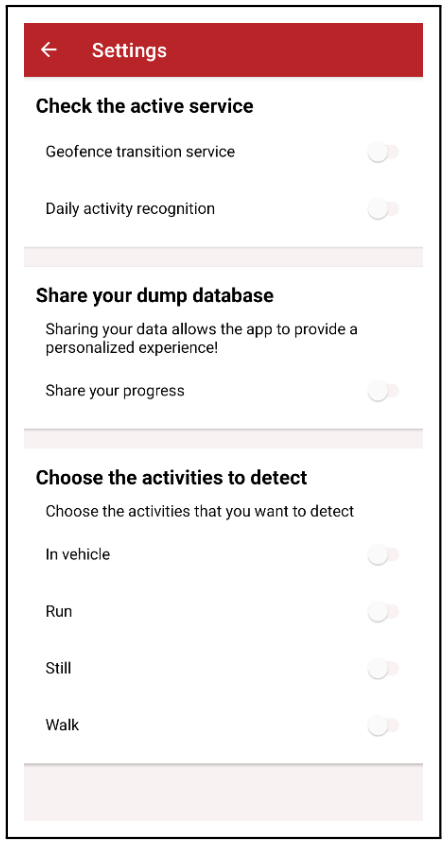
- Accesso ai **dati condivisi** da parte di ulteriori iscritti
- Possibilità di cercare, aggiungere e rimuovere un utente oppure un amico
- Visualizzazione delle informazioni accessibili tramite **TabLayout**





Settings

- Centrallizzata la logica di **acquisizione dei permessi** durante il runtime
- Pannello di controllo in cui l'utente può **abilitare e disabilitare** i servizi offerti
- Definizione da parte dell'utente delle attività da monitorare tramite **Activity Recognition**



Operazioni in background

Le operazioni in background si articolano in tre sezioni differenti, quali:

- **Activity Recognition**, definizione e memorizzazione autonoma delle attività motorie compiute dall'utente
- **Geofencing**, memorizzazione delle transizioni dell'utente qualora dovesse varcare la soglia di un'area di interesse
- **Connectivity**, receiver utilizzato per acquisire tutti i cambiamenti incisivi di connessione del dispositivo

Architettura delle operazioni in background

A livello implementativo, tutte le background operations seguono una stessa architettura. Sono utilizzati tre componenti principali: Broadcast Receiver, Background Service e Worker.

Il Receiver è responsabile di intercettare gli intenti inviati dal SO, per poi delegare la manipolazione delle informazioni acquisite a specifici **Worker**.

I Worker provvederanno ad effettuare alcune operazioni di **pre-processing** prima di risvegliare i **Service**.

Infine, i Service, in base alle informazioni ricevute, stabileranno il comportamento successivo che debba mantenere l'applicazione.

SSOT

Il **Model** definisce l'insieme dei dati ritenuti autorevoli, memorizzati in un contenitore durante le interazioni con l'utente. Avviene una suddivisione del Model in due entità distinte, quali: Room e Supabase.

La decisione di sviluppare un doppio livello di persistenza dei dati nasce principalmente dall'esigenza di garantire che un singolo utente rimanga il punto di **riferimento unico** a livello locale.

Pertanto, il real time database è incaricato di memorizzare tutte le informazioni degli utenti iscritti.

Al momento del login oppure del logout avviene un trasferimento dei dati secondo un formato consono, quale un dump, a seconda della casistica.

Bucket AWS

La condivisione dei propri dati tramite l'applicazione avviene mediante **AWS**.

Abilitato il servizio all'interno dell'activity Settings, l'applicazione programmerà un **Worker periodico** incaricato di effettuare un **dump** del database.

Il dump è successivamente caricato all'interno di un **Bucket** mediante un richiesta **AWS-S3Client**.