Gymnázium, Praha 6, Arabská 14

Programování

Maturitní práce

2025

Tobiáš Brňák

Gymnázium, Praha 6, Arabská 14

Arabská 14, Praha 6, 160 00

Maturitní práce

Předmět: Programování

Téma: SocioMAP

Školní rok: 2024/2025

Autor: Tobiáš Brňák

Třída: 4.E

Vedoucí práce: Mgr. Jan Lána

Třídní učitel: Mgr. Blanka Hniličková

Prohlašujeme, že jsme jedinými autory tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů udělujeme bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne 31. března 2025 _____

Anotace

Tento projekt je mobilní aplikace vytvořená v jazyce Java s využitím frameworku Android Studio. Aplikace umožňuje uživatelům zobrazovat a filtrovat události na mapě podle různých kritérií, jako jsou přátelé, kteří se přihlásili k události, její vlastník nebo oblíbenost. Data jsou ukládána a spravována pomocí Firebase Firestore, což zajišťuje efektivní synchronizaci a správu uživatelských informací a událostí. Hlavním cílem aplikace je usnadnit uživatelům objevování zajímavých událostí ve svém okolí.

Abstract

This project is a mobile application developed in Java using the Android Studio framework. The application allows users to display and filter events on a map based on various criteria, such as friends who have signed up for an event, its owner, or popularity. Data is stored and managed using Firebase Firestore, ensuring efficient synchronization and management of user information and events. The main goal of the application is to facilitate the discovery and organization of events in the user's surroundings.

Anmerkung

Dieses Projekt ist eine mobile Anwendung, die in Java mit dem Android Studio-Framework entwickelt wurde. Die Anwendung ermöglicht es den Benutzern, Veranstaltungen auf einer Karte anzuzeigen und nach verschiedenen Kriterien zu filtern, z. B. nach Freunden, die sich für eine Veranstaltung angemeldet haben, dem Veranstalter oder der Beliebtheit. Die Daten werden mit Firebase Firestore gespeichert und verwaltet, was eine effiziente Synchronisation und Verwaltung von Benutzerinformationen und Veranstaltungen gewährleistet. Das Hauptziel der Anwendung ist es, die Entdeckung und Organisation von Veranstaltungen in der Umgebung des Benutzers zu erleichtern.

Zadání

Cílem aplikace je vytvořit interaktivní mapu Země, která zobrazuje zajímavá místa a akce, přidávané samotnými uživateli. Aplikace bude vyvíjena v jazyce Java pomocí Android Studio a bude využívat databázi Firebase pro ukládání a správu dat.

Na mapě budou umístěny odznaky, které budou obsahovat informace o kategorii, datu a čase dané události. Například u odznaku může být uvedeno: „Park Ladronka – LadronkaFest, 24.2. 2024“. Uživatelé si budou moci zvolit, jaké typy akcí chtějí vidět.

Další funkcí aplikace bude označení událostí spojených se známými osobnostmi, například koncertů, pomocí speciálních odznaků.

Obsah

1 Úvod **6**

2 Vývojové prostředí **7**

2.1 Android Studio 7

2.1.1 Emulátor Android zařízení 7

2.2 Firebase 8

2.3 Další knihovny 8

3 Analýza požadavků **9**

3.1 Požadavky na uživatele a software 9

3.2 Požadavky ze zadání 9

3.3 Možnosti vylepšení a dosud nesplněné požadavky 11

4 SocioMap **12**

4.1 Architektura aplikace 12

4.2 Logika zobrazování a interakce 17

4.2.1 Role a oprávnění uživatelů 29

5 Program **31**

5.1 Pomocné třídy 31

5.2 Firebase 32

5.2.1 Struktura databáze (Firestore Collections) 32

5.2.2 Pravidla zabezpečení (Security Rules) 34

5.2.3 Autentizace uživatelů 34

5.3 Vlastní design 34

5.4 Markery – události 35

5.4.1 Vývoj struktury událostí 35

5.4.2 Algoritmus doporučené události 35

5.4.3 Vývoj algoritmu a potíže s měřením vzdálenosti 36

5.5 Přátelé 37

6 Instalace **39**

6.1 Android a Git 39

6.2 Firebase (backend) 39

6.3 Postup instalace 40

7 Závěr **41**

Seznam obrázků 42

Seznam ukázek kódu 42

1 Úvod

Předmětem tohoto ročníkového projektu bylo vytvořit aplikaci v Android Studiu v jazyce Java, která slouží jako sociální síť pro vyhledávání všemožných událostí v blízkém okolí.

Nejpřirozenější způsob, jak člověk může objevit události kolem sebe, je pomocí mapy. Proto je aplikace postavena především na interaktivní mapě. Během vývoje jsem se rozhodl přidat i možnost interakce mezi uživateli, přestože to původní zadání nepožadovalo. Uvědomil jsem si totiž, že sociální síť bez vzájemné komunikace není tak úplně sociální sítí.

Aplikace je navržena tak, aby byla přívětivá všem. Zpočátku byla aplikace mířena na dorostence a dospívající, jelikož osoby v tomto věku často tráví svůj volný čas venku společně se svými vrstevníky. Informace o událostech získávají ze sociálních sítí jako je Instagram, ale aplikace není přímo navržena pro vyhledávání akcí, nebo vyhledávají akce na internetu.

Aplikaci lze dále rozšiřovat o mnoho dalších funkcí. Většina z nich je již implementována, ale před zveřejněním v Google Play by bylo vhodné doladit zbývající chybějící prvky a vyladit funkčnost.

2 Vývojové prostředí

Pro vývoj aplikace SocioMap bylo využito intuitivní a programátorsky přívětivé vývojové prostředí, které umožňuje efektivní práci s mobilní platformou Android a zároveň nabízí přímé propojení s cloudovými službami (Firebase propojení). Následující sekce stručně popisují klíčové technologie a nástroje, které byly při vývoji použity.

2.1 Android Studio

Android Studio je oficiální integrované vývojové prostředí (IDE) pro vývoj aplikací na platformě Android. Je založeno na IntelliJ IDEA a poskytuje široké spektrum nástrojů a funkcí pro vývoj, ladění (náhled XML souborů) a testování mobilních aplikací.

Vývoj aplikace SocioMap probíhal ve verzi Android Studio Giraffe (2023.1.1), která nabízí plnou podporu pro programovací jazyk Java a Kotlin, vizuální návrhář uživatelského rozhraní (XML layout editor), emulátory Android zařízení a integrované napojení na Google služby. Výhodou prostředí je i integrovaná správa závislostí přes Gradle a možnost snadného nasazení a ladění aplikace na reálném zařízení.

2.1.1 Emulátor Android zařízení

Při vývoji a testování aplikace SocioMap byl využit vestavěný Android emulátor dostupný v Android Studiu. Emulátor umožňuje simulovat různé verze systému Android, velikosti obrazovky, rozlišení a další vlastnosti zařízení, což je ideální pro testování kompatibility a správného chování aplikace při absenci fyzického zařízení.

V projektu byly testovány zejména emulované telefony s verzemi Android 10 (API 29) a Android 12 (API 31). Konkrétně telefony Pixel 8 Pro, Pixel 7. Emulátor také umožňoval testování lokalizačních funkcí pomocí simulace GPS polohy, což bylo klíčové pro ověřování funkčnosti mapy a filtrování událostí podle blízkosti uživatele.

Díky emulátoru bylo možné otestovat design a vyzkoušet náročnější logiku method aplikace.

2.2 Firebase

Firebase je platforma vyvinutá společností Google, která nabízí rozsáhlé backendové služby pro mobilní a webové aplikace. V projektu SocioMapa byly využity následující moduly Firebase:

- **Firebase Authentication** – slouží k registraci a přihlašování uživatelů pomocí e-mailu a hesla. Zajišťuje bezpečné ověření identity a správu uživatelů.
- **Firebase Firestore** – cloudová NoSQL databáze, která uchovává data o uživatelích, událostech a jejich vzájemných vztazích. Nabízí jednoduché čtení, zápis a dotazování nad kolekcemi a dokumenty v reálném čase.
- **Firebase Storage** – připraveno pro případné budoucí využití, například pro ukládání profilových fotografií či obrázků k událostem.

Výhodou Firebase je snadná integrace do Android aplikace a dostupnost ve free-tier verzi, která postačuje pro začatečnické malé projekty.

2.3 Další knihovny

Při vývoji aplikace byly použity i další podpůrné knihovny a API:

- **Google Maps API** – slouží pro zobrazení mapy a práci s geografickými prvky (markery, pozice, události).
- **Material Components** – knihovna pro použití moderních prvků UI v souladu s pravidly Material Designu (TextInputLayout, Buttons, Dialogy apod.).
- **AndroidX** – moderní knihovny pro zpětnou kompatibilitu a správu fragmentů, přechodů mezi aktivitami a systémových oprávnění.

Použití těchto nástrojů a knihoven umožnilo vytvořit stabilní, přehlednou a uživatelsky přívětivou aplikaci s napojením na moderní cloudové technologie.

3 Analýza požadavků

V této kapitole rozeberu, co požaduje a využívá má aplikace, aby byla splněna očekávání od samotného uživatele. K Android zařízení není třeba žádného hardwaru navíc. Postačí vám jen daný chytrý telefon (specifikuji později v této kapitole).

3.1 Požadavky na uživatele a software

Má aplikace požaduje jako každá jiná sociální síť přihlášení přes email a heslo. Jste-li nový uživatel v mé aplikaci, je nutno se registrovat. Registrace požaduje email a heslo, které jsou využívány k autentizaci uživatele. Nadále je třeba vyplnit jméno (Name), příjmení (Surname), přezdívku (Username) a posledně datum narození (Birthday). Pokud vám něco chybí, či máte něco vyplněného špatně, jste na vše upozorněni. SocioMap má též nároky na těžší heslo. Po úspěšném zadání všech registračních polí, vám Firebase mail zašle ???verifikovat??? email. Jakmile ověříte svůj email tím, že ve své příchozí poště naleznete email od mé aplikace a kliknete na odkaz verifikovat, tak se následně můžete přihlásit do aplikace.

Jste přesměrováni dále do aplikace, kde jste žádáni o povolení k přístupu k aktuální poloze telefonu. Není nutné polohu povolit, ale jestli tak neučiníte, omezíte tím chod funkce a přicházíte o rozšířené funkce (převážně o algoritmus nejvhodnější akce).

Poslední požadavek je na systém, týkající se přístupu k internetu. Uživatel nesmí blokovat přístup k internetu aplikaci, jinak mu aplikace nebude fungovat. Zároveň je třeba, aby byl připojen k internetu, aby se navazoval kontakt mezi aplikací a databází Firebase (backend).

3.2 Požadavky ze zadání

Z předchozí kapitoly je patrné, že aplikace SocioMapa je založena na interakci mezi uživateli. Z tohoto důvodu byla implementována funkcionalita přihlášení a registrace uživatelů, včetně

validace vstupních dat a ověření e-mailu.

I přesto, že zadání výslovně nevyžadovalo možnost sociální interakce, bylo zřejmé, že je to stěžejní součást celkové funkcionality. Každý uživatel si může ostatní rozdělit do dvou kategorií:

- uživatelé, které sleduje podle toho, na jaké události se přihlašují (tzv. přátelé),
- tvůrci událostí, jejichž akce chce mít pod dohledem (oblíbení tvůrci).

Aplikace se aktuálně soustředí na události, a proto zatím neobsahuje chatovací funkci ani napojení na externí komunikační nástroje.

Hlavním zobrazením aplikace je mapa, která byla vytvořena pomocí Google Maps API. Mapa obsahuje odznaky (markery) různých barev podle tématu události. Nad mapou jsou dostupné filtry:

- výběr tématu události,
- filtrování podle přátel,
- filtrování podle oblíbených tvůrců,
- filtrování pouze „známých“ událostí (označené jako „famous“, obsahuje nadřazené zbarvení markeru).

Uživatel má možnost přepínat mezi režimem úpravy (přidávání událostí) a režimem náhledu (zobrazení událostí).

Každá událost obsahuje detaily jako popis, čas konání, místo, věkové omezení, tematiku a tvůrce. Markery, které již časově expirovaly, jsou automaticky archivovány a přesunuty do jiných kolekcí ve Firestore.

Dále má uživatel k dispozici profilovou sekci, kde může upravit své osobní údaje, změnit preferovaná témata a odhlásit se. V poslední sekci pak najde přehled všech akcí, které vytvořil, i těch, na které se přihlásil.

Administrátor má přístup ke speciálním funkcím: může zakazovat účty ostatním uživatelům a též uživatelům přidat status „famous“

Splněné požadavky:

- Přihlášení a registrace včetně validací.
- Možnost sledovat přátele a oblíbené tvůrce.
- Zobrazení událostí na mapě s různými filtry.
- Využití lokace uživatele na algoritmus nejvhodnější události.
- Vytváření, filtrování, archivace a správa událostí.
- Uživatelé mohou nastavit svá oblíbená témata.
- Odlišení známých uživatelů pomocí vizuálních prvků.
- Interaktivní design
- Admin rozhraní pro správu aplikace.

Požadavky na software:

- Android Studio (verze 2023.1+ doporučena)
- Firebase Firestore, Authentication
- Google Maps API klíč
- Minimální verze Androidu: 7.0 (API 24)

3.3 Možnosti vylepšení a dosud nesplněné požadavky

Aplikaci se nepodařilo integrovat s přihlášením přes Google Sign-in, přičemž tento kód zůstává zakomentován. Zatím neobsahuje chatovací funkci pro komunikaci mezi uživateli ani není integrována s externími kalendáři či systémem notifikací. Optimalizace pro všechny verze Android zařízení není kompletní, a zejména starší nebo méně výkonné modely mohou mít problémy s výkonem. Pokud by měla být aplikace v budoucnu výdělečná, bylo by vhodné zavést jednorázový poplatek za registraci nebo přihlášení uživatele.

4 SocioMap

Aplikace **SocioMap** je navržena jako interaktivní sociální síť zaměřená na sdílení a objevování událostí prostřednictvím mapového rozhraní. Celý koncept se opírá o rychlost a intuitivní ovládání, které umožňuje uživatelům snadno interagovat s obsahem a ostatními uživateli. Klíčovým prvkem aplikace je právě Google mapa, kde se veškeré události zobrazují jako barevně odlišné markery – každý reprezentuje jednu událost, která se právě na tom místě bude v budoucnu odehrávat. Aplikace poskytuje několik způsobů filtrování, přepínání režimů a vyhledávání.

Navigace mezi fragmenty

Navigace v aplikaci probíhá primárně prostřednictvím spodního navigačního panelu označeného jako 'btnNavMenu', který je přístupný z hlavních částí aplikace. Těmi hlavními částmi aplikace jsou tyto:

- **Mapa (MapFragment)** základní obrazovka aplikace.
- **Profil (ProfileFragment)** sekce s osobními údaji uživatele.
- **Přehled (OverviewFragment)** seznam všech událostí spojené s uživatelem.

Přechody mezi fragmenty jsou okamžité, bez zbytečného zdržování, a uživatel tak může aplikaci ovládat velmi intuitivně. Buďto člověk využívá již zmíněné navigační menu mezi hlavními částmi a dále se dostává pomocí tlačítek. [7] [13]

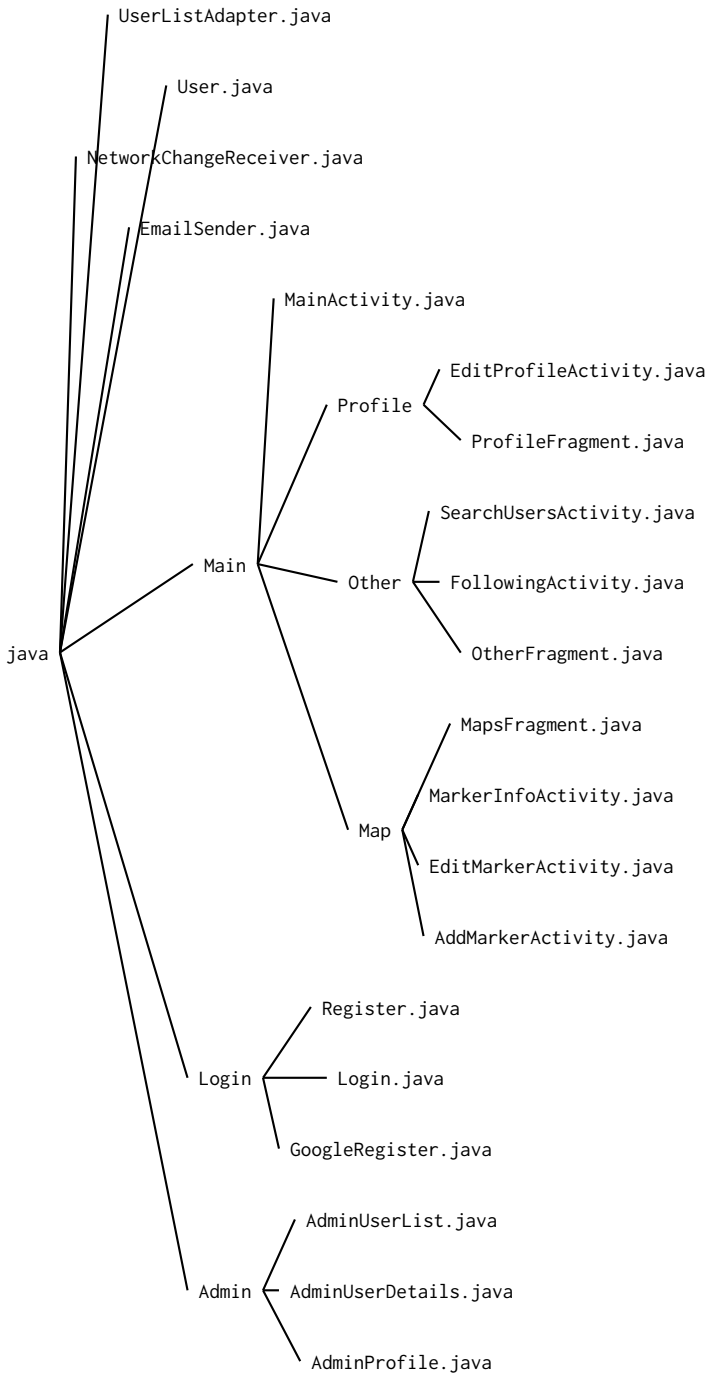
4.1 Architektura aplikace

Architektura aplikace SocioMap je navržena modulárně, aby bylo možné jednotlivé části snadno rozšiřovat. Aplikace je strukturována do logických balíčků (packages) dle odpovědnosti dané komponenty. Tato organizace zajišťuje lepší přehlednost kódu.

Všechny hlavní části aplikace jsou psány v jazyce Java a organizovány podle následující stromové struktury:

- **Admin** – Obsahuje komponenty určené výhradně pro správce systému. Správce má přístup k seznamu uživatelů, může zobrazovat jejich detaily a případně zakazovat přístup nebo je povýšit na status *famous*.
  - `AdminProfile.java` – hlavní rozhraní administrátora.
  - `AdminUserList.java` – komponenta pro zobrazování seznamu uživatelů.
  - `AdminUserDetails.java` – zobrazuje podrobnosti o vybraném uživateli.
- **Login** – Zajišťuje registraci a přihlášení uživatelů.
  - `Register.java` – aktivita pro registraci nového uživatele, včetně validace a propojení s Firestore.
  - `Login.java` – hlavní přihlašovací aktivita, prozatím nepodporuje Google Sign-In.
  - `GoogleRegister.java` – slouží k doplnění údajů po přihlášení přes Google účet. Zatím nevyužíván.
- **Main** – Jádro celé aplikace. Obsahuje tři hlavní podsložky podle jednotlivých fragmentů:
  - `MainActivity.java` – základní třída pro správu spodní navigace a přepínání mezi fragmenty.
  - **Map** – součást odpovědná za mapové rozhraní:
    - `MapsFragment.java` – hlavní logika zobrazování markerů, filtrování a interakce s mapou.
    - `AddMarkerActivity.java` – přidání nové události kliknutím do mapy.
    - `EditMarkerActivity.java` – úprava dříve vytvořené události.
    - `MarkerInfoActivity.java` – detailní informace o konkrétní události. Pokud je uživatel vlastník, zobrazují se mu tlačítka na úpravu
  - **Profile** – zajišťuje správu osobního účtu:
    - `ProfileFragment.java` – hlavní obrazovka s osobními údaji a možností úprav.

    - `EditProfileActivity.java` – umožňuje změnu osobních údajů.
  - **Other** – doplňkové části, které nejsou hlavními obrazovkami, ale slouží k rozšíření funkcionality a obecnému přehledu:
    - `OtherFragment.java` – Přehled všech událostí.
    - `FollowingActivity.java` – zobrazuje seznam přátel, které uživatel sleduje.
    - `SearchUsersActivity.java` – umožňuje vyhledávat další uživatele.
- **Pomocné třídy** – nachází se mimo výše uvedené složky:
  - `User.java` – modelová třída pro uchování údajů o uživateli.
  - `UserListAdapter.java` – adaptér pro zobrazování seznamu uživatelů.
  - `EmailSender.java` – pomocná třída pro odesílání emailů (např. při ověření nebo správě účtů).
  - `NetworkChangeReceiver.java` – komponenta pro zjišťování a kontrolu dostupnosti připojení k internetu.



[7] [13]

Struktura a propojení komponent

Každá část aplikace je odpovědná za svůj úkol a komunikuje s ostatními komponentami přes zřetelně definovaná rozhraní. Fragmenty jsou spravovány pomocí 'MainActivity', která zajišťuje plynulé přepínání mezi jednotlivými hlavními obrazovkami. Dále se odkazují pomocí tlačítek.

Aplikace komunikuje s Firebase (Firestore a Authentication) napříč všemi moduly pomocí 'endpointů' – od přihlášení, přes ukládání a načítání uživatelů a událostí, až po správu dat v reálném čase. Přístup k databázi Firebase (konkrétně Firestore) probíhá pomocí oficiálního `Firebase Java SDK`, které je do projektu přidáno jako závislost pomocí `build.gradle`. Komunikace s databází je asynchronní – data jsou načítána pomocí*addOnSuccessListener()* nebo *addSnapshotListener()*.

Endpointový dotaz na Firebase:

literate

```
    firestore.collection("markers")
    .whereGreaterThan("latitude", userLocation.latitude - latOffset)
    .whereLessThan("latitude", userLocation.latitude + latOffset)
    .get()
    .addOnSuccessListener(queryDocumentSnapshots -> {
        if (!queryDocumentSnapshots.isEmpty()) {
            for (DocumentSnapshot document : queryDocumentSnapshots) {
                Double latitude = document.getDouble("latitude");
                Double longitude = document.getDouble("longitude");
                String title = document.getString("title");

                if (latitude != null && longitude != null && title != null) {
                    LatLng markerPosition = new LatLng(latitude, longitude);
                    googleMap.addMarker(new MarkerOptions()
                        .position(markerPosition)
                        .title(title));
                }
            }
        }
    })
    .addOnFailureListener(e -> {
        Log.e("Firestore", "Error - loading markers", e);
    });
```

Listing 4.1: Endpoint - dotázání se na databázi

Data se z Firestore vrací ve formě *DocumentSnapshot*, odkud je lze získat pomocí metod jako *getString()*, *getDouble()* apod.

Mapové rozhraní je realizováno pomocí knihovny **Google Maps SDK for Android**, která je také přidána do projektu přes `build.gradle`. Fragment s mapou je definován pomocí `SupportMapFragment`, který se načítá v rámci 'MapsFragment':

literate

```
    SupportMapFragment mapFragment =
    (SupportMapFragment) getChildFragmentManager()
        .findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
```

Listing 4.2: Načítání GoogleMaps Api - `SupportMapFragment`

V metodě `onMapReady()` pak dochází k práci s objektem `GoogleMap`, na který se přidávají markery, nastavuje se zoom, lokalizace a další prvky mapy.

Použití pomocných tříd, jako je `NetworkChangeReceiver`, zajišťuje kontrolu dostupnosti připojení a celkovou robustnost aplikace. [7] [13] [4] [11] [3]

4.2 Logika zobrazování a interakce

MainActivity.java

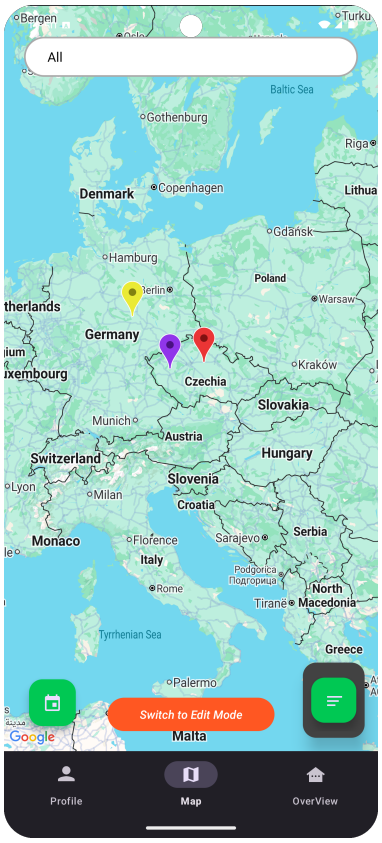
Tato aktivita představuje hlavní vstupní bod aplikace po přihlášení uživatele. Zajišťuje zobrazení spodní navigační lišty a řízení přepínání mezi jednotlivými fragmenty:

- **MapFragment** – zobrazení událostí na mapě,
- **OtherFragment** – přehled přihlášených a vytvořených událostí,
- **ProfileFragment** – uživatelský profil a nastavení.

Při kliknutí na jednotlivé položky navigace se načítají odpovídající fragmenty. Aplikace si uchovává aktivní stav, a proto zůstává uživatel v posledním zvoleném okně i při rotaci displeje nebo návratu do aplikace.

Aktivita také načítá data o uživateli a v případě, že se jedná o slavnou osobu, aplikuje speciální design.

MapsFragment



Obr. 4.1: MapsFragment

Zobrazování událostí je centralizováno do MapFragmentu. Uživatel se v mapě pohybuje pomocí tahů prstů zoom-in/out. Markery se načítají z databáze Firebase Firestore, kde jsou události uloženy. Každý marker se zobrazuje v jiné barvě podle tematického zaměření události (např. sport, hudba, festival apod.), které může vyfiltrovat. Uživatelská interakce s markery je možná pomocí jednoduchého kliknutí, které otevře okno s názvem a popisem události, a následně může uživatel přejít na detailní obrazovku události.

Součástí mapového rozhraní je filtrovací panel, který lze rozbalit a sbalit. Pomocí něj může uživatel filtrovat pouze události, kterých se účastní přátelé, filtrovat podle tvůrců, které sleduje, nebo zobrazit pouze události pořádané slavnými osobami.

Aplikace bere ohled na nezletilé, tudíž je-li uživatel příliš mladý na danou akci, automaticky se mu událost neukáže. Věkovou hranici nastavuje tvůrce akce.

Z hlediska režimu ovládání aplikace je možné přepnout do režimu úprav, kdy může uživatel

přidat vlastní událost kliknutím na libovolné místo na mapě. Tento režim skrývá všechny filtry, aby nedocházelo k rušení při přidávání nové události.

Na základě uživatelovy polohy dokáže aplikace také zvýraznit „nejlepší událost v okolí“ – pomocí algoritmu, který zohledňuje vzdálenost, oblíbená témata, historii účasti a aktivitu přátel. Lze využít lokalizování a přiblížení uživatele pomocí tlačítka od Googlu v pravém horním rohu.

Všechny markery, které teprve budou nadcházet, se sesbírají z kolekce ’markers’, kde se vezmou pouze jejich koordinace (longitude, latitude) a tématika, kvůli zbarvení.

literate

```
private void loadMarkers() {
    firestore.collection("users").document(userId).get().addOnSuccessListener
    (userDoc -> {
        int userAge = calculateAge(userDoc.getString("birthyear"));

        firestore.collection("markers").get().addOnSuccessListener(
        queryDocumentSnapshots -> {
            googleMap.clear();
            Date now = new Date();

            for (DocumentSnapshot document : queryDocumentSnapshots) {
                String title = document.getString("title");
                String theme = document.getString("theme");
                String eventDateTime = document.getString("eventDateTime");
                int ageLimit = document.getLong("ageLimit").intValue();

                if (userAge < ageLimit) continue;
                if (!passesAllFilters(document)) continue;

                LatLng position = new LatLng(
                    document.getDouble("latitude"),
                    document.getDouble("longitude")
                );

                googleMap.addMarker(new MarkerOptions()
                    .position(position)
                    .title(title)
                    .icon(BitmapDescriptorFactory.defaultMarker(getMarkerColor(theme)))
                ).setTag(document.getId());
            }
        });
    });
}
```

Listing 4.3: Metoda `loadMarkers()` – načítání a filtrování událostí

Tato metoda slouží k načtení všech událostí z databáze a jejich vykreslení na mapě. Před samotným zobrazením se kontroluje věk uživatele vůči minimálnímu věkovému limitu každé události, dále se aplikují filtry (např. podle data, tématu, známých osob nebo přátel), a pokud událost projde těmito podmínkami, je zobrazena na mapě.

literate

```
private float getMarkerColor(String theme) {
    if (theme == null) return BitmapDescriptorFactory.HUE_RED;

    switch (theme.toLowerCase()) {
        case "sports":
            return BitmapDescriptorFactory.HUE_BLUE;
        case "music":
            return BitmapDescriptorFactory.HUE_VIOLET;
        case "festival":
            return BitmapDescriptorFactory.HUE_YELLOW;
        case "workshop":
            return BitmapDescriptorFactory.HUE_GREEN;
        case "custom":
            return BitmapDescriptorFactory.HUE_ORANGE;
        default:
            return BitmapDescriptorFactory.HUE_RED;
    }
}
```

Listing 4.4: Změna barvy markeru podle tématu události

Tato metoda vrací barvu, kterou bude mít marker na mapě podle zadaného tématu události. Pomocí konstrukce *switch* se porovnává textový řetězec tématu a přiřazuje se odpovídající hodnota barvy z `BitmapDescriptorFactory`. Pokud je téma neznámé nebo chybí, použije se výchozí červená barva.

AddMarkerActivity

Tato aktivita slouží k přidání nové události na mapu. Uživatel se na tuto obrazovku dostane kliknutím na libovolné místo v mapě v režimu „Úpravy“. V úvodu obrazovky se zobrazí formulář, ve kterém uživatel vyplňuje název události, stručný popis, zvolenou tematiku (např. sport, hudba, festival), věkové omezení a přesný čas konání.

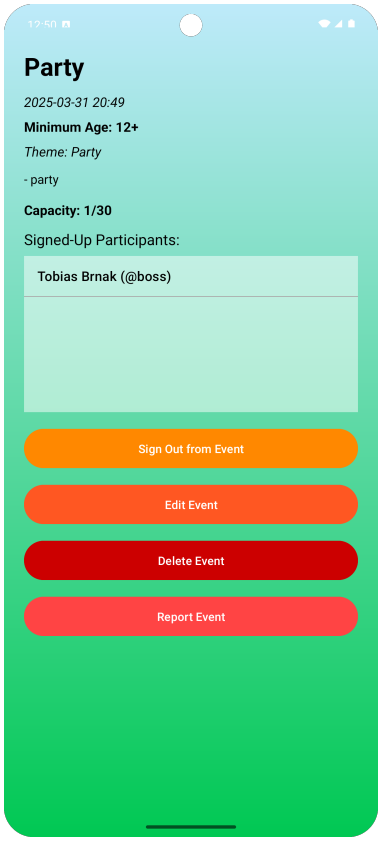
Formulář rovněž obsahuje datumový a časový výběr pomocí příslušných dialogových oken, které zajišťují správné formátování vstupu. Veškeré údaje jsou validovány — např. datum musí být v budoucnosti, některá pole jsou povinná.

Po úspěšném přidání je uživatel automaticky přesměrován zpět do mapového zobrazení, kde se nová událost objeví po synchronizaci jako nový marker. Logika v této aktivitě je

tedy zcela propojena s databází a aktuální polohou uživatele.

Nový marker se uloží do kolekce ’markers’, kde vznikne pod novým ID.

MarkerInfoActivity



Obr. 4.2: MarkerInfoActivity

Tato aktivita slouží jako detailní přehled konkrétní události, která byla vybrána z mapového zobrazení. Obsahuje všechny dostupné informace o události – název, popis, téma, datum a čas, věkové omezení, pozici na mapě i informace o tvůrci události.

Zobrazí se zde i možnost přihlášení se na danou událost (tlačítko „Zúčastnit se“), pokud se uživatel ještě nezúčastnil, případně možnost odhlášení. Uživatel zde také vidí, kolik dalších lidí se události účastní. Též vidí kolik míst zbývá do plné kapacity události.

Pokud je uživatel tvůrcem události, má navíc k dispozici tlačítko editace (přesměrování do ’EditMarkerActivity’) a mazání události.

Získává základní informace z kolekce 'users'. Nadále je vyplní do kolonek a po uložení se použije příkaz aktualizace databáze.

Tato aktivita vyhledá ID markeru v kolekci 'markers' a vypíše všechna informace. Seznam přihlášení se vezme z kolekce 'event_guest_list', kde pod stejným ID jsou napsání uživatelé, kteří se na akci přihlásili.

EditMarkerActivity

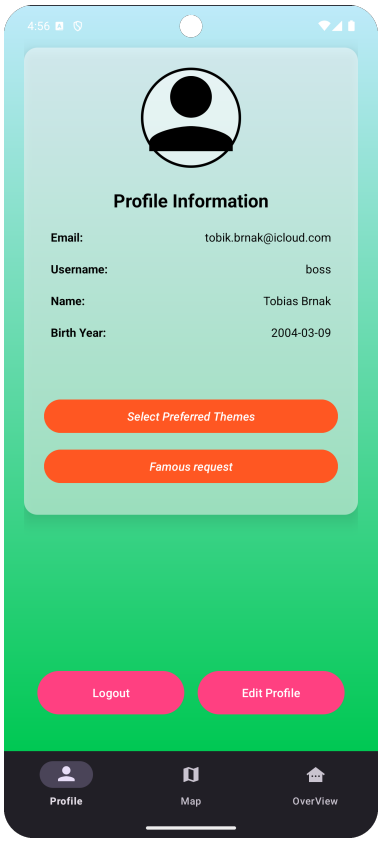
Editační aktivita slouží ke správě již existující události, kterou vytvořil sám uživatel. Přístup k této obrazovce je možný skrze detail události 'MarkerInfoActivity' pouze v případě, že je aktuálně přihlášený uživatel zároveň tvůrcem dané události.

Zobrazuje předvyplněný formulář s údaji o události, jako je název, popis, datum a čas, tematické zaměření a případné věkové omezení. Uživatel může libovolné pole upravit, a následně změny uložit. Náhled události se po editaci aktualizuje i v mapovém zobrazení.

Tato aktivita rovněž umožňuje zrušit událost úplným smazáním. Bezpečnostní kontrola zajišťuje, že danou událost může upravovat a mazat výhradně její vlastník.

Získává základní informace z kolekce 'markers'. Nadále je vyplní do kolonek a po uložení se použije příkaz aktualizace databáze.

ProfileFragment



Obr. 4.3: ProfileFragment

Tento fragment slouží jako uživatelský profil. Po načtení fragmentu se zobrazí informace přihlášeného uživatele - přezdívka (username), jméno, příjmení, e-mail a rok narození. Tyto informace jsou zobrazeny formou přehledného seznamu v kartičce.

Uživatel má možnost upravit své osobní údaje kliknutím na tlačítko Editovat profil, které ho přesměruje do aktivity 'EditProfileActivity', odhlásit se z aplikace pomocí tlačítka Odhlásit se, které spustí návrat na přihlašovací obrazovku, nebo si vybrat oblíbená témata událostí kliknutím na tlačítko Vybrat témata, což otevře výběrové dialogové okno se seznamem témat.

Pokud je uživatel slavný (*Famous*), je jeho profilový obrázek automaticky změněn na ikonku korunky. Tím je vizuálně odlišena jeho identita.

'ProfileFragment' vyhledá ID shodné s přihlášeným uživatelem a vypíše dané informace, které se následně zobrazí.

EditProfileActivity

Tato aktivita umožňuje uživateli upravit své základní osobní údaje. Po načtení se automaticky předvyplní stávající data – přezdívka (username), jméno, příjmení a rok narození.

Uživatelské rozhraní obsahuje vstupní pole pro každou informaci a tlačítko pro potvrzení změn. Při odeslání dat proběhne kontrola platnosti zadaných údajů (např. zda nejsou prázdné) a následně dojde k jejich aktualizaci v dokumentu přihlášeného uživatele.

Získává základní informace z kolekce 'users'. Nadále je vyplní do kolonek a po uložení se použije příkaz aktualizace databáze.

OtherFragment

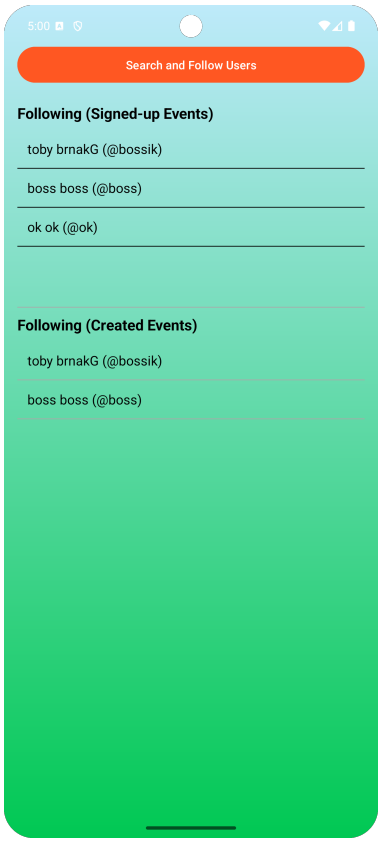
Tento fragment slouží jako přehled účasti uživatele. Uživatel zde najde 4 hlavní sekce seznamů:

- Události, které uživatel vytvořil a budou se konat – bere je z kolekce 'markers', kde zjišťuje, zda se schoduje 'userID' s ID přihlášeného uživatele.
- Události, na které vytvořil a již se konaly – dotázání se na kolekci 'user_owner_arch'
- Události, na které se přihlásil v budoucnu – dotázání se na kolekci 'user_events'
- Události, na které se přihlásil v minulosti – dotázání se na kolekci 'user_arch'

Každá událost je zobrazena jako jednoduchý seznam s názvem a datem. Kliknutím na konkrétní událost se uživatel přesměruje do detailu události (‘MarkerInfoActivity‘), kde si může zobrazit detaily nebo se z události odhlásit.

Tento fragment je vhodný pro rychlou orientaci v historii a plánech uživatele.

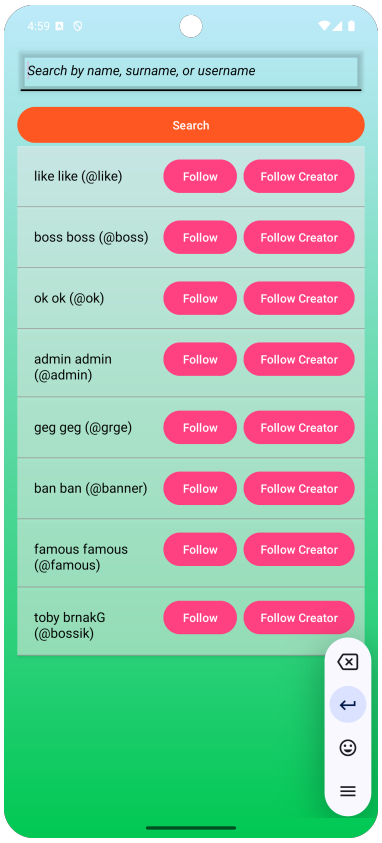
FollowingActivity



Obr. 4.4: FollowingActivity

Aktivita umožňuje zobrazit seznam uživatelů, které daný uživatel sleduje, a to jak v roli tvůrců, tak přihlášených přátel. Aktivita je vybavena dvěma seznamy uživatelů. Sbírá data z kolekce ’user_signup_follow’ a ’user_create_follow’

SearchUsersActivity



Obr. 4.5: SearchUsersActivity

Tato aktivita slouží k vyhledávání ostatních uživatelů v databázi. Po zadání jména nebo přezdívky se vyfiltrují odpovídající výsledky. U každého uživatele má uživatel možnost zobrazit základní informace, začít sledovat daného uživatele z důvodu, kam se přihlašuje (Follow), nebo začít sledovat daného uživatele z důvodu jeho tvorby (Follow creator).

Načítání uživatelů je pomocí adaptéru ve třídě 'UserAdapter', která dále využívá další třídu 'User' shrnující základní informace o uživateli. Filtrování funguje pomocí dotázání se příkazem *.whereEqualTo()*. Po stisknutí jednoho z tlačítek se informace zapíše do kolekce 'user_signup_follow' nebo 'user_owner_follow' pod dokument ID přihlášeného uživatele uživatel, kterého zaklikne, že chce sledovat.

AdminProfile.java

Tato aktivita představuje hlavní rozhraní pro administrátora aplikace. Po přihlášení jako administrátor je uživatel přesměrován právě sem. Od této aktivity se administrátor může dostávat do jiných odvětví.

Zde má administrátor možnost kliknout na tlačítko, kde se zobrazit přehled všech uživatelů aplikace 'AdminUserList'a nebo se může z aplikace odhlásit.

AdminUserList.java

Třída slouží pro získání a zobrazení seznamu všech registrovaných uživatelů aplikace.

Uživatelé jsou zobrazováni v přehledném seznamu. Seznam je interaktivní – kliknutím na položku je administrátor přesměrován na detailní zobrazení konkrétního uživatele.

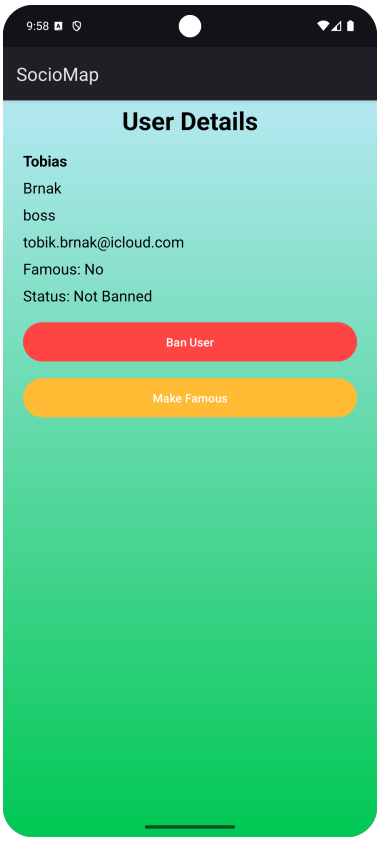
Součástí je již využitá základní logika filtrování a aktualizace dat. Funguje na stejném principu jako aktivita 'SearchUsersActivity'.

AdminUserDetails.java

Tato aktivita zobrazí detailní informace o konkrétním uživateli, vybraném z předchozího seznamu. Zobrazuje všechna informace z databáze.

Administrátor zde může pomocí přepínačů měnit atributy uživatele – například přidělit mu roli admina, slavné osoby nebo ho zabanovat. Po úpravách jsou změny ihned synchronizovány s Firestore databází.

Po kliknutí na kartičku se vyhledá v kolekci 'users' id dokumentu shodný s ID uživatelem, na který bylo kliknuto.

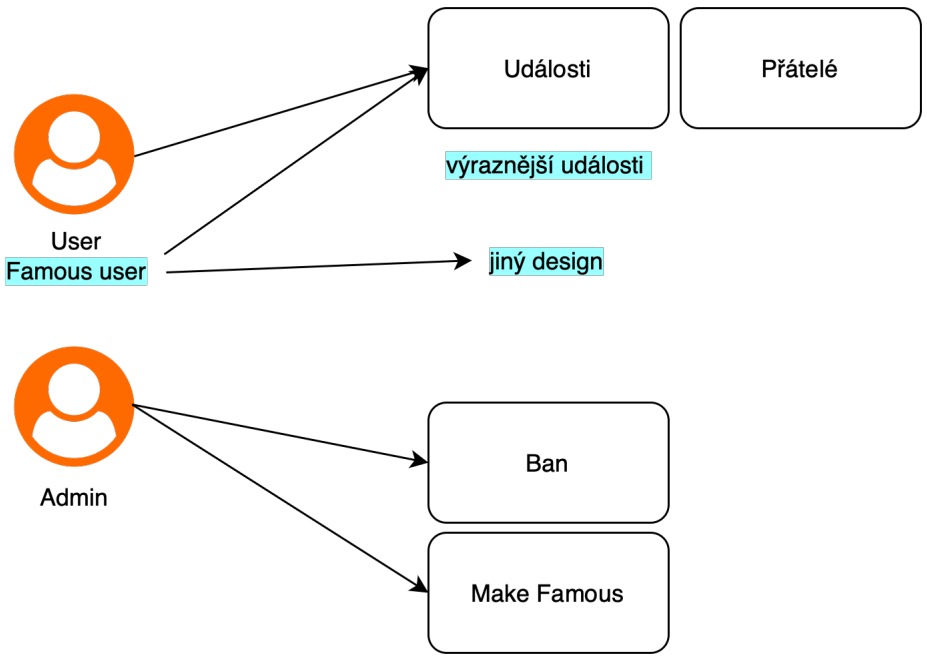


Obr. 4.6: AdminUserDetails

4.2.1 Role a oprávnění uživatelů

Aplikace SocioMapa rozlišuje tři základní typy uživatelů, kteří mají rozdílná oprávnění:

- **Běžný uživatel** – může plně využívat funkcionality aplikace, jako je přidávání přátel, přihlašování se na události, vytváření vlastních událostí a filtrování záznamů v mapě. Má přístup pouze ke svým údajům a událostem.
- **Slavný uživatel (Famous)** – je vizuálně zvýrazněn pomocí speciálního designu. Na mapě se jeho události zobrazují pomocí odlišného stylu (např. korunka, výraznější barva markeru), což zvyšuje jejich viditelnost pro ostatní uživatele.
- **Administrátor (Admin)** – má rozšířená oprávnění pro správu systému. Může:
  - nastavovat status `isFamous` jiným uživatelům,
  - blokovat účty pomocí příznaku `ban`,
  - zobrazovat podrobné informace o všech uživatelích.



Obr. 4.7: Role uživatele a jejich vztahy v systému

Na obrázku je znázorněno, jak běžný uživatel a slavný uživatel interaguje s přáteli (sleduje je a vidí jejich aktivity) a s událostmi (vytváří je nebo se na ně přihlašuje). Admin uživatel smí interagovat s proměnnými jako jsou *ban* a *isFamous*.

[2] [4] [5] [12] [6] [13] [7] [1] [8]

5 Program

5.1 Pomocné třídy

User.java

Třída 'User' slouží jako datový model pro uživatele aplikace. Obsahuje základní atributy, jako je identifikátor uživatele (userId), přezdívka (username), jméno (name) a příjmení (surname). Tento model je využíván při práci s databází Firestore – jak při získávání údajů o uživateli, tak při jejich ukládání. Umožňuje přehlednější manipulaci s daty v rámci kódu a zajišťuje jednotnou strukturu uživatelských informací napříč aplikací.

UserListAdapter.java

Tato třída je 'RecyclerView.Adapter', která slouží k zobrazení seznamu uživatelů v rozhraní administrátora a v aktivitě 'SearchUsersActivity'. Každý uživatel je vykreslen jako samostatná položka se základními informacemi (jméno, příjmení, přezdívka) dle předlohy ze třídy 'User'.

Adapter propojuje data z databáze s vizuální reprezentací v seznamu. Při kliknutí na položku je vyvolán listener, který otevře detail konkrétního uživatele v aktivitě 'AdminUserDetails'.

NetworkChangeReceiver.java

Tato komponenta je zaregistrována jako `BroadcastReceiver`, který sleduje stav připojení k internetu.

Pokud dojde ke ztrátě připojení, je uživatel upozorněn vizuálním prvkem (ikona, hláška) a některé interakce (např. přihlášení, registrace) jsou dočasně zakázány. Tudíž se uživatel nemůže přihlásit.

Třída preventivně zakazuje vstoupit do aplikace bez připojení k internetu, jelikož bez přístupu k databázi, nemá aplikace smysl.

EmailSender.java

Tato pomocná třída umožňuje odesílání e-mailů pomocí SMTP protokolu. Je využívána zejména pro správu systému ze strany administrátora – např. při nahlášení události nebo informování uživatelů. Výjimkou je změna statusu z běžného uživatele na slavného, kdy uživatel obdrží email s informací, že se stává slavným.

Třída obsahuje základní SMTP konfiguraci a metody pro vytvoření zprávy a její odeslání. Připojuje se k e-mailové službě, jako je Gmail, a zabezpečuje přenos přihlašovacích údajů i obsahu zprávy.

Používá JavaMail API a pracuje na pozadí, aby neblokovala hlavní UI vlákno.

[7] [8]

5.2 Firebase

Pro realizaci backendové části jsem zvolil platformu Firebase, konkrétně služby Firestore, Firebase Authentication. Vzhledem k povaze aplikace – tedy zaměření na víceuživatelské prostředí a potřebu rychlé synchronizace dat – nebyla použita lokální SQL databáze. Lokální databáze by neumožnila správu účtů napříč zařízeními, a při přihlášení z jiného zařízení by uživatel neměl přístup ke svým událostem či osobním údajům. Firestore zároveň umožňuje práci v reálném čase, díky čemuž se změny (např. vytvoření nebo úprava události) okamžitě promítnou všem uživatelům.

5.2.1 Struktura databáze (Firestore Collections)

Databáze je tvořena několika kolekcemi, z nichž každá má specifický účel:

- **users** – obsahuje základní údaje o uživatelích (jméno, příjmení, přezdívka, rok narození, email, typ účtu – admin/slavný, priorizované tématiky) [String = (username, name,

surname, email, birthday) Boolean = (isFamous, isAdmin, ban) List = (preferredThemes)]. Dokument vytváří ID uživatele po úspěšné registraci.

- **markers** – aktivní události, které se zobrazují na mapě. Každý dokument v kolekci ‘markers‘ obsahuje informace (název, popis, souřadnice, datum a čas, věkové omezení, tematické zaměření a ID autora, kapacitu, maximální kapacita). [Number = (ageLimit, currentAttendees, latitude, longitude, maxCapacity) String = (description, eventDateTime, theme, title, userId)] Dokument nese nově vytvořený ID markeru.
- **markers_arch** – archiv událostí, které již proběhly. Pokud akce již proběhla, bude přesunuta sem.
- **user_events** – seznam událostí, na které se uživatel přihlásil. [List = (events)] Dokument nese ID uživatele.
- **user_owner_arch** – archiv událostí vytvořených konkrétním uživatelem. [List = (events)] Dokument nese ID uživatele.
- **user_sign_up_follow** – seznam uživatelů, jejichž účast na akcích sleduji. [List = (following)] Dokument nese ID uživatele.
- **user_create_follow** – seznam uživatelů, jejichž tvorbu akcí sleduji. [List = (following)] Dokument nese ID uživatele.
- **event_guest_list** – seznam přihlášených uživatelů pro konkrétní událost. [List = (users)] Dokument nese ID markeru.

Aplikace, jak již bylo zmíněno, využívá komunikaci na bázi endpointů pro dotazování do databáze. Pro filtrování dat je použit příkaz *.whereGreaterThan("latitude", userLocation.latitude - latOffset)*, který vrátí dokumenty s hodnotou větší než zadaná hodnota, nebo .whereIn("category", Arrays.asList("sports", "music", "tech")), který vrátí dokumenty odpovídající jedné z hodnot v seznamu. Odkaz na kód je dostupný v kapitole Struktura a propojení komponent.

[4] [12] [10] .

5.2.2 Pravidla zabezpečení (Security Rules)

Pro zajištění bezpečnosti a správné správy dat byla v databázi Firestore nastavena pravidla, která omezují přístup pouze na oprávněné a přihlášené uživatele. Pravidla zajišťují, že číst a zapisovat události v kolekci 'markers' může pouze jejich vlastník. Stejné omezení platí i pro přístup k uživatelským dokumentům v kolekci 'users', kde má každý uživatel přístup výhradně ke svým vlastním datům. Obdobně je chráněna kolekce 'user_events', kde může uživatel pracovat pouze se svými událostmi. Dále jsou nastavena obecná pravidla, která umožňují pouze přihlášeným uživatelům číst a zapisovat, ale neumožňují hromadný výpis celé kolekce, čímž je zabráněno případnému zneužití nebo stahování všech dokumentů najednou.

Tato pravidla zajišťují, že pouze přihlášení uživatelé mají přístup k datům. Uživatel může upravovat nebo číst pouze svůj vlastní dokument. Nikdo nemůže stáhnout celou kolekci dokumentů najednou, což je zajištěno pravidlem list: if false. Každý může číst veřejné informace o událostech, ale upravovat je smí pouze jejich autor.

[4]

5.2.3 Autentizace uživatelů

Firebase Authentication je využita pro přihlašování, registraci a ověření emailu. Prozatím je využívána pouze metoda 'Email a heslo'– uživatel se zaregistruje se svým emailem a heslem. Následně je vyžadována verifikace emailu. Poté je vytvořen nový dokument ve Firebase.

Dále byly nastaveny restrikce na heslo. Heslo požaduje velká a malá písmena, znak, číslici a délku minimálně 8 znaků. Počet přihlášení za hodinu je omezen na 10 pokusů. [4]

5.3 Vlastní design

V celé aplikaci byl použit vlastní design s důrazem na vizuální konzistenci a unikátní uživatelské rozhraní. Místo výchozích systémových prvků byly navrženy vlastní komponenty. Designový prvek filtrační panel na mapě byl graficky upraven tak, aby neomezil zorné pole.

Zároveň byl doplněn o animaci pro plynulé rozbalení a sbalení, což zlepšuje uživatelský dojem.

Seznam uživatelů (tzv. UserList) byl taktéž navržen jako vlastní layout. Každý uživatel je zobrazen v kartičce se zaobleným pozadím, odlišeným tlačítkem pro sledování a celkově přehledným uspořádáním textu. Pro dosažení tohoto vzhledu byly využity vlastní XML soubory s barevným přechodem, úpravou fontů a ohraničením. [1]

5.4 Markery – události

Události jsou reprezentovány jako markery na mapě a jsou uloženy v kolekci `markers` ve Firebase Firestore. Každý marker obsahuje základní informace jako název, popis, lokaci, datum a čas události, téma, věkové omezení a příznak popularity (`isFamous`).

5.4.1 Vývoj struktury událostí

Na začátku jsem zvažoval více způsobů, jak data o událostech ukládat – například jestli mít každého uživatele jako hlavní dokument a jeho události jako podkolekci. Nakonec jsem se rozhodl pro samostatnou kolekci 'markers', protože je přehlednější dotazovat všechny události napříč uživateli. Zároveň je snadnější filtrovat podle témata, datumu nebo věku. Tento přístup umožňuje efektivnější vyhledávání a doporučování událostí bez nutnosti procházení dalších kolekcí a také nabízí možnost využít více úložišť.

Informace o přihlášených uživatelích jsem přesunul do samostatné kolekce 'event_guest_list' a informace o vlastních vytvořených událostech do 'user_owner_arch'. Tento rozklad umožňuje lepší přehlednost, oddělení funkcí a větší kapacitu pro data.

5.4.2 Algoritmus doporučené události

Pro doporučení ideální události v okolí jsem implementoval algoritmus, který zohledňuje polohu uživatele, vzdálenost událostí, oblíbená témata a také počet přihlášených přátel. Cílem je doporučit nejbližší událost, která má co největší šanci uživatele zaujmout. Na

mapě je tato událost zvýrazněna odlišnou barvou a zobrazí se popisek s názvem akce doprovázený popisem o akci.

Algoritmus nejprve získá aktuální polohu uživatele pomocí GPS souřadnic. Poté načte z kolekce 'user_signup_follow' seznam uživatelů, které aktuální uživatel sleduje. Následně stáhne všechny události (markery) v okruhu 7 km podle zeměpisné šířky a délky.

Každé události se vypočítá skóre na základě několika faktorů: vzdálenosti od uživatele, podílu přihlášených přátel (čím více, tím lépe) a shody s oblíbenými tématy uživatele (pokud se téma shoduje, skóre se sníží). Nakonec se událost s nejnižším skóre označí jako "doporučená"a vizuálně se zvýrazní na mapě.

5.4.3 Vývoj algoritmu a potíže s měřením vzdálenosti

Na začátku jsem testoval více způsobů, jak porovnávat vzdálenost mezi událostmi a uživatelem například pomocí přímého odečtu rozdílů souřadnic. Tento způsob ale nebral v úvahu zakřivení Země a rozdíl v měřítku mezi šířkou a délkou, a proto vedl k nepřesnostem.

Zvažoval jsem také použití externí knihovny (např. haversine plugin), ale nakonec jsem se rozhodl implementovat vlastní výpočet pomocí Haversinova vzorce, který bere v úvahu zakřivení Země a je dostatečně přesný pro výpočet vzdáleností v řádu kilometrů. [6]

Haversinův vzorec

Matematický vzorec, který umožňuje vypočítat přibližnou vzdálenost mezi dvěma body na kulovém povrchu (jako je Země). Funguje tak, že nejprve převede souřadnice ze stupňů na radiány, spočítá rozdíly mezi šířkami a délkami, a pak přes trigonometrické funkce určí vzdálenost. Používá se přitom přibližný poloměr Země (6 371 km). Výsledná vzdálenost je pak vrácena v metrech a slouží jako jeden z hlavních faktorů při výběru doporučené události.

Použitá metoda *getDistance()* vrací vzdálenost v metrech mezi dvěma souřadnicemi a je použita jako hlavní faktor při hodnocení jednotlivých markerů.

Původně jsem se snažil vzdálenost mezi dvěma body (uživatelem a událostí) počítat jednoduše jako rozdíl mezi zeměpisnou šířkou a délkou. Tento přístup se ale ukázal jako velmi nepřesný,

protože nepočítal se zakřivením Země a tím, že vzdálenost jednoho stupně se liší v různých zeměpisných šířkách. Proto jsem se rozhodl použít tzv. haversinovu formuli – matematický vzorec, který umožňuje vypočítat přibližnou vzdálenost mezi dvěma body na kulovém povrchu (jako je Země). Funguje tak, že nejprve převede souřadnice ze stupňů na radiány, spočítá rozdíly mezi šířkami a délkami, a pak přes trigonometrické funkce určí vzdálenost. Používá se přitom přibližný poloměr Země (6 371 km). Výsledná vzdálenost je pak vrácena v metrech a slouží jako jeden z hlavních faktorů při výběru doporučené události.

literate

```
    private double getDistance(LatLng pos1, LatLng pos2) {
        double lat1 = pos1.latitude;
        double lon1 = pos1.longitude;
        double lat2 = pos2.latitude;
        double lon2 = pos2.longitude;

        double R = 6371e3;
        double phi_1  = Math.toRadians(lat1);
        double phi_2 = Math.toRadians(lat2);
        double delta_phi = Math.toRadians(lat2 - lat1);
        double delta_lambda = Math.toRadians(lon2 - lon1);

        double a = Math.sin(delta_phi / 2) *
                Math.sin(delta_phi / 2) +
                Math.cos(phi_1) *
                Math.cos(phi_2) *
                Math.sin(delta_lambda / 2) * Math.sin(delta_lambda / 2);
        double c = 2 * M
        ath.atan2(Math.sqrt(a), Math.sqrt(1 - a));

        return R * c;
    }
```

Listing 5.1: metoda getDistance() -výpočet vzdálenosti

[9]

5.5 Přátelé

Vyhledávání je napojeno na Firestore kolekci 'users'. K vyhledávání se využívají jednoduché dotazy pomocí 'whereGreaterThanOrEqualTo' a 'whereLessThan' filtrů, které odpovídají

textovému vstupu zadanému uživatelem.

Vzhledem k tomu, že jsem se v prvních fázích vývoje věnoval především práci s markery, rozhodl jsem se použít obdobný přístup také pro správu přátel. To znamená, že jsem vytvořil strukturu založenou na kolekcích ve Firebase, které jsou jednoduše dotazovatelné a snadno rozšiřitelné.

Uživatelé si mohou ostatní přidat mezi „sledované“ kliknutím ve výsledku vyhledávání. Tato akce následně vytvoří nebo aktualizuje dokument v příslušné kolekci podle typu sledování. Během konzultace s panem profesorem jsem přišel na to, že by bylo užitečné rozlišovat mezi těmi, které uživatel sleduje na základě účasti na událostech (např. kam chodí), a těmi, které sleduje jako organizátory (tvůrce událostí). Na tomto základě byly vytvořeny dvě samostatné kolekce:

- 'user_signup_follow' – obsahuje ID uživatelů, které sledujeme podle jejich účasti na událostech.
- 'user_owner_follow' – obsahuje ID uživatelů, které sledujeme jako tvůrce událostí.

Při implementaci filtrování v mapě tedy dochází k jednoduchému dotazu na tuto kolekci, načte se seznam ID a následně se porovnává s 'userId' nebo seznamem přihlášených uživatelů v jednotlivých událostech. Ve výsledku je tímto způsobem zajištěna funkcionalita personalizovaného zobrazení mapy podle sociálních vazeb uživatele.

6 Instalace

Tato kapitola popisuje, jakým způsobem lze aplikaci SocioMapa připravit a spustit v prostředí vývojáře. Jsou zde uvedeny všechny potřebné nástroje, nastavení a konfigurace.

6.1 Android a Git

Pro vývoj a správu aplikace SocioMapa je potřeba mít nainstalované následující nástroje, jimiž jsou Android Studio – vývojové prostředí pro Android aplikace, Java Development Kit (JDK) – doporučuje se verze 17 nebo novější, Git – pro klonování repozitáře a správu verzí.

6.2 Firebase (backend)

Aplikace využívá Firebase jako backend pro autentizaci uživatelů, ukládání dat (Firestore) a další služby.

Postup nastavení Firebase:

1. Vytvořte projekt na https://console.firebase.google.com.
2. Přidejte Android aplikaci do projektu pomocí balíčkového jména aplikace.
3. Stáhněte konfigurační soubor google-services.json a vložte ho do složky app/ v Android projektu.
4. Ujistěte se, že v souboru build.gradle (Project) i build.gradle (Module: app) jsou správně přidány závislosti Firebase a plugin google-services.
5. V konzoli Firebase nastavte authentication (Email/Password + Google), Firestore databázi (v režimu testování nebo s upravenými pravidly)

6. Upravte pravidla přístupu do Firestore podle požadavků na zabezpečení.

6.3 Postup instalace

1. Klonujte repozitář aplikace.
2. V Android Studiu otevřete složku s klonovaným projektem.
3. Nechte Android Studio automaticky provést synchronizaci závislostí (Gradle Sync).
4. Nastavte Firebase viz. *Postup nastaveni Firebase:*
5. Vytvořte API pro GoogleMaps a změňte API key v aplikaci

7 Závěr

Podařilo se mi naprogramovat zcela funkční mobilní aplikaci SocioMapa, která umožňuje uživatelům vytvářet a objevovat společenské události na základě geografické polohy, zájmů a interakce s přáteli. Během vývoje jsem se seznámil s řadou technologií — od Firebase služeb přes práci s mapami Google Maps až po návrh uživatelského rozhraní v Android Studiu.

S výsledkem jsem spokojen. Aplikace obsahuje mnoho funkčních interakcí, která jsou použitelná pro běžné využití. Jedinou výraznější překážkou, kterou se zatím nepodařilo zcela vyřešit, je integrace přihlášení přes Google účet, která zůstává nefunkční.

Do budoucna bych rád do aplikace přidal možnost závazné platby při přihlášení na vybrané události, čímž by se systém přiblížil reálnému provozu a zároveň by bylo možné zajistit vyšší spolehlivost přihlášek. Uvažuji také o rozšíření funkcí, jako je systém hodnocení uživatelů, propojení s kalendářem, vylepšené notifikace nebo přímá komunikace mezi účastníky.

Na závěr bych rád poděkoval celému pedagogickému sboru.

Seznam obrázků

4.1 MapsFragment 18
4.2 MarkerInfoActivity 22
4.3 ProfileFragment 24
4.4 FollowingActivity 26
4.5 SearchUsersActivity 27
4.6 AdminUserDetails 29
4.7 Role uživatele a jejich vztahy v systému 30

Seznam ukázek kódu

4.1 Endpoint - dotázání se na databázi 16
4.2 Načítání GoogleMaps Api - `SupportMapFragment` 17
4.3 Metoda `loadMarkers()` – načítání a filtrování událostí 20
4.4 Změna barvy markeru podle tématu události 21
5.1 metoda getDistance() -výpočet vzdálenosti 37

Seznam zdrojů

[1] Material Design. *Material Components for Android*. Dostupné z: https://m3.material.io/develop/android/overview. 2025.

[2] Google Developers. *DatePickerDialog | Android Developers*. Dostupné z: https://developer.android.com/reference/android/app/DatePickerDialog. 2025.

[3] Google Developers. *Maps SDK for Android — Developer Guide*. Dostupné z: https://developers.google.com/maps/documentation/android-sdk/start. 2025.

[4] Google Firebase. *Firebase Authentication for Android - Password Sign-In*. Dostupné z: https://firebase.google.com/docs/auth/android/password-auth#java_2. 2024.

[5] Google Firebase. *Firebase Firestore Filter - Kotlin Reference*. Dostupné z: https://firebase.google.com/docs/reference/kotlin/com/google/firebase/firestore/Filter. 2024.

[6] OpenAI ChatGPT. *Pomoc při vývoji aplikace SocioMap (algoritmus pro nejlepší událost, Google Sign-In)*. Osobní konzultace v rámci vývoje aplikace. 2025.

[7] Oracle. *Java Platform, Standard Edition Documentation*. Dostupné z: https://docs.oracle.com/en/java/javase/. 2025.

[8] Stack Overflow. *Add Marker on Android Google Map via Touch or Tap*. Dostupné z: https://stackoverflow.com/questions/17143129/add-marker-on-android-google-map-via-touch-or-tap. 2013.

[9] Movable Type Scripts. *Calculate distance between two latitude-longitude points using Haversine Formula*. Dostupné z: https://www.movable-type.co.uk/scripts/latlong.html. n.d.

[10] Simplilearn. *SQL for Beginners: Complete Playlist*. Dostupné z: https://www.youtube.com/watch?v=hJPk50p7xwA&list=PLSrm9z4zp4mGK0g_0_jxYGgg3os9tqRUQ. 2021.

[11] Coding With T. *Implement Google Maps API in Android Studio*. Dostupné z: https://www.youtube.com/watch?v=JzxjNNCYt_o. 2021.

[12] Tech With Tim. *Firebase Login and Registration - Quick Setup*. Dostupné z: https://www.youtube.com/watch?v=QAKq8UBv4GI. 2020.

[13] Programmer World. *Firebase + Google Maps Android Tutorial*. Dostupné z: https://www.youtube.com/watch?v=tZvjSl9dswg. 2022.