ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORATIKY

Galiba App

DOKUMENTÁCIA

Vypracoval: **Šimon Bartánus**
Študijná skupina: **5ZYS31**
Predmet: **Vývoj aplikácií pre mobilné zariadenia**
Cvičiaci: *doc. Ing. Patrik Hrkút, PhD*.

Obsah

Popis a analýza riešeného problému

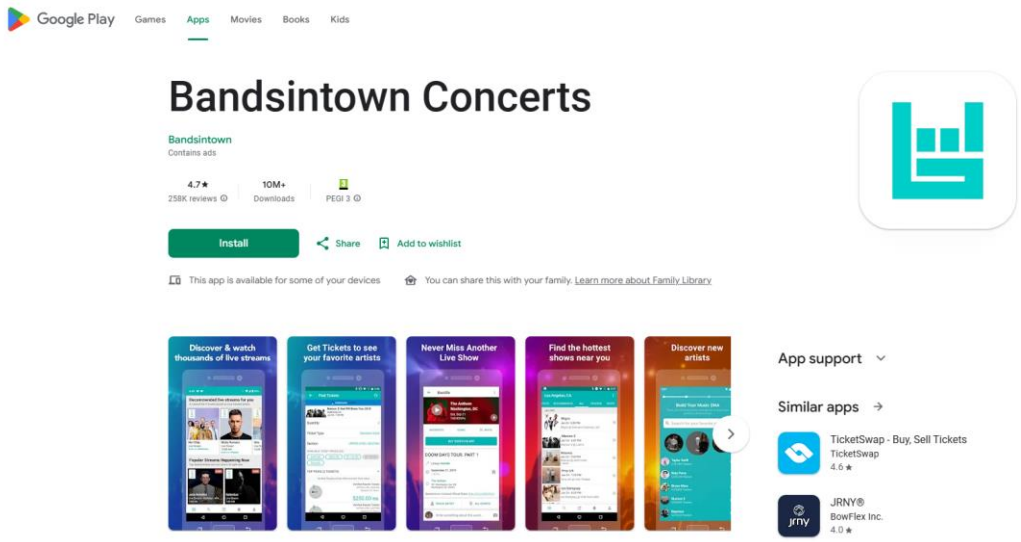
Úvod

Ako semestrálnu prácu sme si zvolili jednoduchú aplikáciu ktorá bude slúžiť komunite menším kapelám/umelcom na rýchle vytváranie akcií/koncertov a poslucháčom na rýchle nájdenie takýchto interpretov/akcií menšieho rozmeru. Aplikácia bude závislá na internetovom pripojení používateľa pre nájdenie podujatia akéhokoľvek typu a jej hlavná myšlienka tak bude spropagovať čo najviac nezávislých, spontánnych koncertov konajúcich sa v blízkom okolí na jednom mieste.

Dostupné aplikácie podobného zamerania

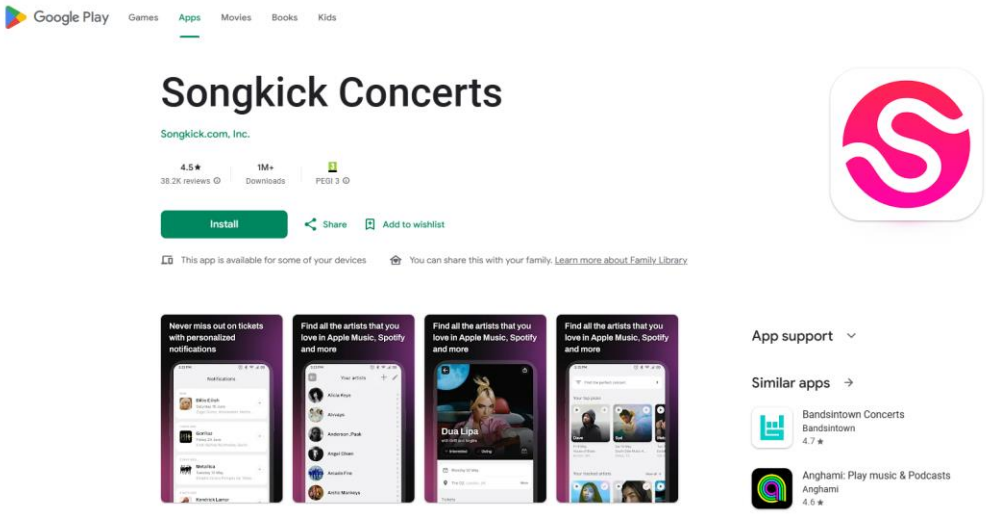
Bandsintown

Bandsintown je poprednou aplikáciou na objavovanie koncertov pre Android, ktorá má najvyššie hodnotenie a je najviac sťahovaná. Umožňuje užívateľom nikdy nezmeškať živé vystúpenia, pretože v jednom mieste poskytuje informácie o všetkých živých vystúpeniach. Pomáha udržiavať kontakt s obľúbenými umelcami prostredníctvom virtuálnych koncertov prenášaných priamo do zariadenia užívateľa alebo nájdením vstupeniek na nadchádzajúce predstavenia a turné.



Soundkick

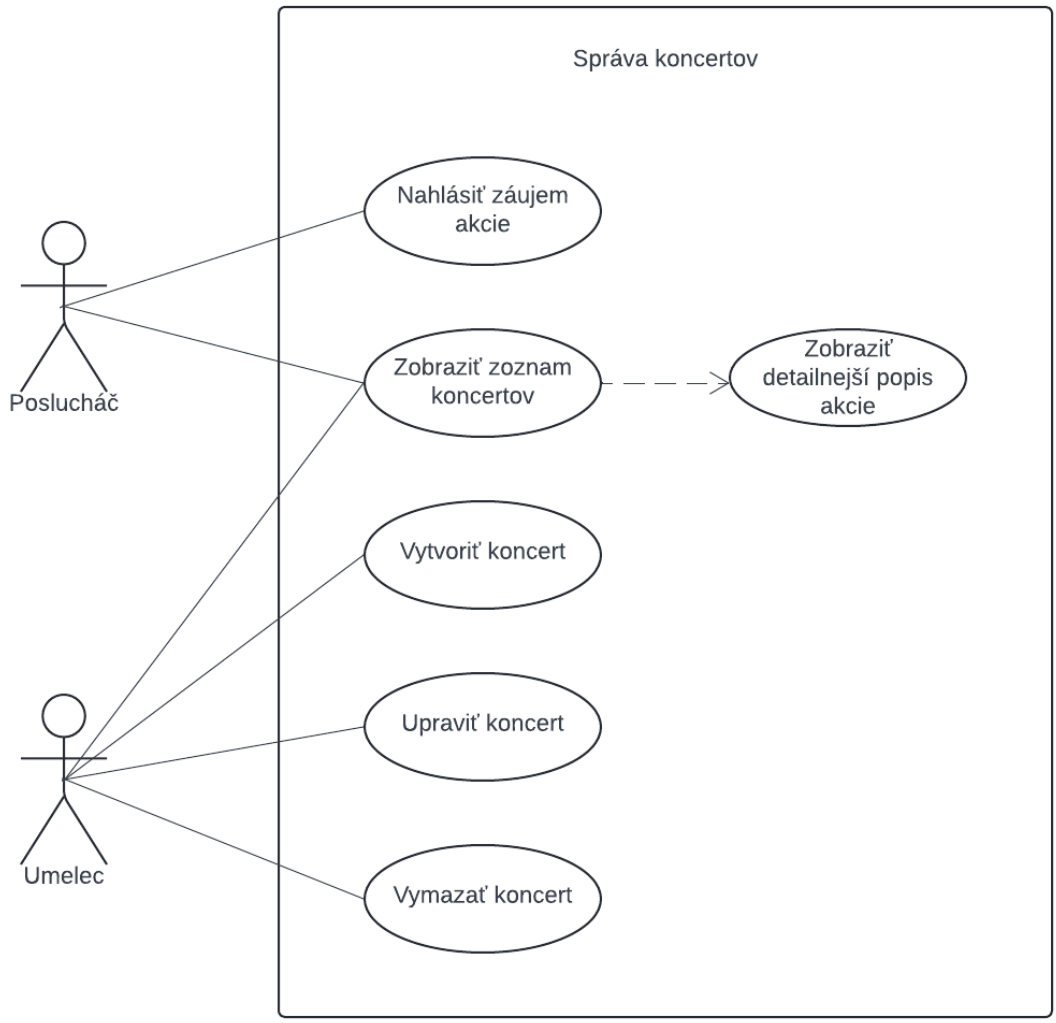
Aplikácia Songkick je komplexný nástroj pre nadšencov živej hudby. Umožňuje používateľom sledovať ich obľúbených umelcov a dostávať notifikácie o dátumoch koncertov a predaji vstupeniek. Integráciou so službami ako Spotify, Google Play Music a Facebook ponúka používateľom objavovať živé hudobné podujatia vo svojom okolí, priamo v spomenutých aplikáciách a tak nikdy nezmeškať príležitosť vidieť ich obľúbených umelcov naživo.



Návrh riešenia problému

Diagram prípadov použitia

Diagram ukazuje proces správy a nahliadanie koncertov v aplikácií. Rozlišuje 2 rozdielne typy používateľov aplikácie (Poslucháč a Umelec). Poslucháč je štandardný typ ktorý môže zobraziť zoznam koncertov a po zvolení zobraziť samostatne detailnejšie popisy každej akcie ktoré sa mu zobrazia formou listu v domovskej obrazovke, následne sa môže rozhodnúť či ho akcia zaujala alebo sa definitívne rozhodol ísť pokiaľ poslucháča zaujal Umelec ktorý organizuje danú akciu môže si nahliadnuť jeho profil a začať Kapelu/Umelca sledovať aby mal jednoduchší prehľad o nadchádzajúcich akciách. Na druhej strane Umelec môže okrem podobného zobrazenia koncertov mazať alebo upravovať svoje nadchádzajúce udalosti, taktiež má rýchly prehľad o záujme jeho akcie formou vyčísleného počtu záujemcov a tých ktorý naozaj chcú prísť. Po zvolení jeho vytvorenej udalosti dostáva namiesto záujmu možnosť upraviť meno, dátum, umiestnenie, fotku alebo popis akcie.



Obrázok 1 Use case diagram popisujúci správu koncertov

Návrh architektúry aplikácie

Architektúra aplikácie sa skladá z databázového serveru Firebase, ktorý uchováva informácie o používateľoch a vytvorených koncertoch a aplikáciou, ktorá s ním komunikuje. Všetci používatelia ktorý chcú používať aplikáciu sa musia prihlásiť do svojho účtu aby mohli ďalej pokračovať vo využívaní.

Aplikácia sa skladá z niekoľkých procesov, v prvotnom spustení aplikácie sa vykonáva **prihlasovací proces** ktorý používateľovi vykreslí textové polia (emailová adresa a heslo) následne počká na vstup od používateľa. Pokiaľ používateľ ešte nemá vytvorený účet môže použiť možnosť vytvorenia si účtu, kde mu následne pribudnú vstupné polia pre zvolenie si jeho používateľského mena, potvrdenie hesla a vybratie si z možností o aký typ účtu sa bude jednať (Poslucháč alebo Umelec). Po dokončení procesu prihlásenia sa nastáva **proces spravovania obsahu**, tento proces predstavuje zobrazenie všetkých možných dostupných podujatí. Používateľ môže zahájiť **proces vyhľadania mesta záujmu** v ktorom si po kliknutí na vyhľadávač môže s ponuky vybrať ľubovoľné mesto záujmu a prípadne si ho uložiť k svojím obľúbeným mestám. Teda buď po výbere konkrétneho mesta alebo priamo z úvodnej obrazovky po prihlásení sa do aplikácie poslucháč bude môcť po zvolení nahliadnuť na konkrétne podujatie a dozvedieť sa o ňom bližšie informácie a pokiaľ ho zaujmú môže tak prejaviť možnosťami (prídem alebo mám záujem). Umelec rovnako ako poslucháč si môže zvoliť rôzne podujatia avšak pri zvolení koncertu ktorý vytvoril on sám sa možnosti o prejave záujmu menia na úpravu podujatia kde bude môcť zmeniť všetky dostupné informácie alebo podujatie vymazať. Taktiež umelcovi pribúda možnosť vytvorenia vlastnej akcie priamo z domovskej obrazovky pomocou FAB ktorým môže zahájiť **proces vytvorenia udalosti**. Z domovskej obrazovky sa môže používateľ dostať na obrazovku s umelcami ktorých aktuálne sleduje tá obsahuje hlavný **proces správy sledovaných umelcov**. **Podproces zobrazenia profilu** predstavuje možnosť používateľa vidieť nadchádzajúce podujatia daného umelca ktorého zvolil priamo zo zoznamu sledovaných. Každý používateľ si vytvára na začiatku jednoduchý profil a neskôr si ho môže trocha upraviť prostredníctvom **procesu spravovania profilu**, po zvolení bočného navigačného panela sa používateľ po zvolení vlastnej profilovej ikony (alebo kliknutím priamo na svoje používateľské meno na domovskej obrazovke) dostane na jednoduchý náhľad svojho profilu, proces spravovania sa začne zvolením ikony spravovacích prvkov vedľa používateľského mena, používateľ sa následne objaví v ďalšej obrazovke kde si môže rýchlo upraviť svoje osobné informácie a následne uložiť.

Popis implementácie

Aplikácia potrebuje pre funkčnosť internetové pripojenie na komunikáciu s Google Firebase. Všetky používateľské dáta sú ukladané online.

Ukážka nadviazania komunikácie s Google Firebase

```
class FirebaseInit : Application() {

    override fun onCreate() {
        super.onCreate()

        FirebaseApp.initializeApp(this)

    }

}
```

Na obrázku môžeme vidieť počiatočnú inicializáciu Firebase pomocou kontextu aplikácie, ktorá prebieha v triede Application. Túto metódu je potrebné volať ešte predtým ako budeme môcť použiť akékoľvek Firebase služby, napríklad Firestore, Authentication, Storage atď.

```
private val firebaseAuth = FirebaseAuth.getInstance()

private val firebaseFirestore =
FirebaseFirestore.getInstance()

private val firebaseStorage = FirebaseStorage.getInstance()
```

Pre jednoduchosť sme si nastavil v triede *Firebaseviewmodel* inštancie používaných služieb ako globálne atribúty ktoré budeme používať všade kde budeme potrebovať prácu s danou službou.

```
fun getCurrentUserData() {
    firebaseAuth.currentUser?.also {
        it.email?.also { email ->
            emailId.value = email
        }
    }

    firebaseFirestore.collection("users").document(firebaseAuth.currentUser?.uid.toString())
        .get().addOnCompleteListener {
            if (it.isSuccessful) {
                currentUserId.value = firebaseAuth.currentUser?.uid.toString()
                val document = it.result
                if (document != null && document.exists()) {
                    username.value = document.getString("username")
                    bio.value = document.getString("bio")
                    profilePic.value = document.get("profilePic").toString()
                    isArtist.value = document.getBoolean("isArtist")
                    instagramUsername.value = document.getString("instagramUsername")
                    facebookUsername.value = document.getString("facebookUsername")
                    youtubeUsername.value = document.getString("youtubeUsername")
                    tiktokUsername.value = document.getString("tiktokUsername")
                    website.value = document.getString("website")

                }
            }
        }
}
```

Funkcia *getCurrentUserData* slúži na načítanie aktuálnych používateľských dát z Firebase Authentication a Firestore a ich uloženie do zadaných atribútov. Funkcia začína tým, že skontroluje, či je aktuálny používateľ prihlásený. Ak je, získa jeho emailovú adresu a uloží ju do atribúta. Ďalej funkcia prehľadáva v Firestore kolekciu "users" a načíta dokument, ktorý má ako identifikátor ID aktuálneho používateľa. Po dokončení sa skontroluje, či bol úspešný. Ak áno, nastaví sa hodnota *currentUserId* na aktuálne ID používateľa. Následne funkcia skontroluje, či dokument existuje a obsahuje údaje. Ak áno, extrahuje jednotlivé polia (napr. username, bio, profilePic, isArtist, instagramUsername, facebookUsername, youtubeUsername, tiktokUsername, website) a uloží ich do príslušných atribútov.

```
fun saveToUserProfilePictures(
    onSuccess: () -> Unit,
    onFailure: () -> Unit,
    uri: Uri
) {

    firebaseStorage.reference.child("userProfileImages/${firebaseAuth.uid.toString()}.jpg")
        .putFile(uri).addOnSuccessListener {

firebaseStorage.reference.child("userProfileImages/${firebaseAuth.uid.toString()}.jpg").downloadUrl.addOnSuccessListener {
                firebaseFirestore.collection("users").document(firebaseAuth.uid.toString())
                    .update("profilePic", it.toString())
                    .addOnSuccessListener {
                        onSuccess()
                    }
                    .addOnFailureListener {
                        onFailure()
                    }
            }.addOnFailureListener {
                onFailure()

            }
        }
        .addOnFailureListener {
            onFailure()
        }
}
```

Funkcia *saveToUserProfilePictures* slúži na nahrávanie profilovej fotky používateľa do Firebase Storage a aktualizáciu vygenerovanej URL adresy fotky v konkrétnom zázname v Firestore. Na začiatku sa používa referencia na Firebase Storage, ktorá ukazuje na cestu userProfileImages/(ID používateľa).jpg. Obrázok sa nahrá pomocou metódy putFile(cesta k obrázku v zariadení). Po úspešnom nahraní sa získa URL adresa obrázka pomocou downloadUrl a tento URL sa použije na aktualizáciu Firestore stĺpca „profilePic“ v dokumente ktorého ID je zhodné s aktuálne prihláseným používateľom. Funkcia ošetruje zlyhania v každom

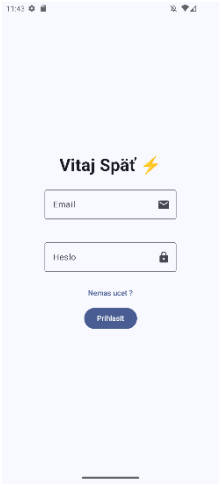
kroku. Ak zlyhá akýkoľvek krok (nahranie obrázka, získanie URL, aktualizácia Firestore), zavolá sa *onFailure()* lambda funkcia.

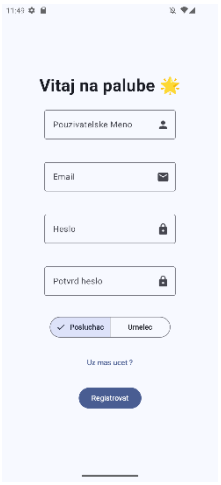
Navigácia

```
NavHost(
    navController = navController,
    route = Screens.ROOT.name,
    startDestination = if(firebaseViewModel.isUserLoggedIn.value == true) {
        Screens.HOME.name
    } else {
        Screens.AUTHROOT.name
    }
) {
    navigation(
        route = Screens.AUTHROOT.name, startDestination = Screens.LOGIN.name
    ) {
        composable(route = Screens.LOGIN.name) {
            LoginScreen(
                onLoginClick = {
                    navController.popBackStack()
                    navController.navigate(Screens.HOME.name)
                },

                onRegisterClick = {
                    navController.navigateToSingleTop(Screens.REGISTER.name)
                },
                firebaseViewModel = firebaseViewModel
            )
        }
        composable(route = Screens.REGISTER.name) {
            RegisterScreen(
                onLoginClick = { navController.navigateToSingleTop(Screens.LOGIN.name) },
                onRegisterClick = {
                    navController.popBackStack()
                    navController.navigate(Screens.HOME.name)
                },
                firebaseViewModel = firebaseViewModel
            )
        }
    }
    composable(route = Screens.HOME.name) {
        HomeScreenNavigation(firebaseViewModel = firebaseViewModel)
    }
}
```

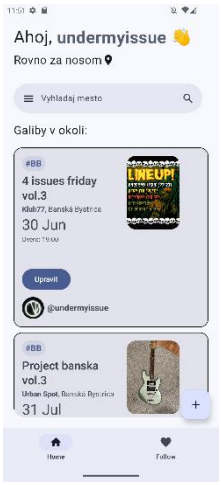
Navigácia medzi obrazovkami sa pri prvotnom spustení aplikácie začína v Login obrazovke (pokiaľ používateľ nie je prihlásený) v ktorej sa používateľ môže prihlásiť do svojho účtu alebo vytvoriť vlastný. Pomocou vnorenej navigácie je možné sa navigovať medzi týmito obrazovkami a po úspešnom prihlásení môže používateľ prejsť na domovskú obrazovku ktorá predstavuje ďalší navigačný graf.

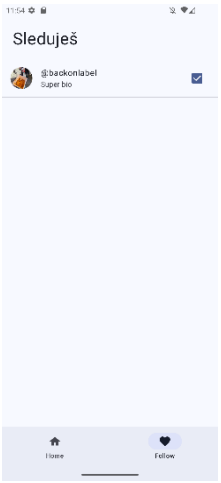




Obrazovka 1 Login

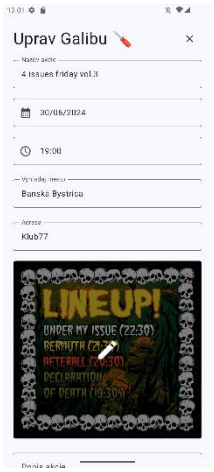
Obrazovka 2 Register

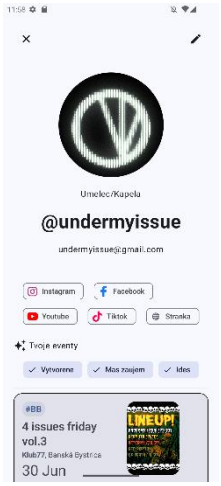


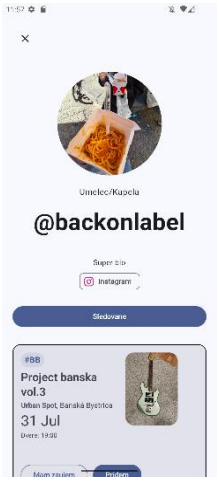


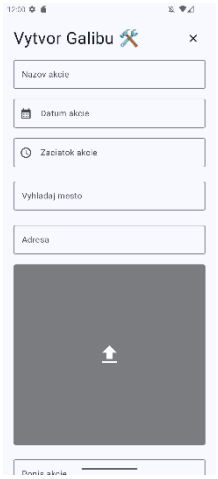
Obrazovka 3 Domov

Obrazovka 4 Sleduješ









Obrazovka 5 Upraviť akciu

Obrazovka 6 Vlastný profil

Obrazovka 7 Upraviť profilové informácie

Obrazovka 8 Profil Umelca

Obrazovka 9 Upraviť akciu

Zoznam použitých zdrojov

Okrem z kurzu predmetu som taktiež čerpal:

- https://m3.material.io/components (Material design komponenty ktoré dotvárajú moderný štýl aplikácie)
- https://github.com/maxkeppeler/sheets-compose-dialogs (Použité pre kalendár a hodiny Composables pretože v súčasnej dobe nie sú voľne dostupné priamo od Google)
- https://firebase.google.com/docs/auth (firebase auth pre jednoduchý user login systém)
- https://firebase.google.com/docs/firestore (firebase firestore pre databázu použitú v app)
- https://firebase.google.com/docs/storage (firebase storage pre ukladanie obrázkov online)
- pomôcky pre prácu s Firebase:
    - https://www.geeksforgeeks.org/android-jetpack-compose-add-data-to-firebase-firestore/?ref=ml_lbp
    - https://firebase.blog/posts/2023/12/adding-complex-queries-to-jetpack-compose-app/
- https://stackoverflow.com/a/6018141 (Špecifický komentár ktorý mi dosť zjednodušil riešenie problému s implementáciou zobrazenia lokácie koncertu)