ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

Semestrálna práca z VAMZ

Vypracovala: Veronika Sivčáková
Študijná skupina: 5ZYR21
Školský rok: 2023/2024
Vyučujúci: doc. Ing. Patrik Hrkút, PhD.

Popis a analýza riešeného problému

Špecifikácia zadania, definovanie problému
Rozhodla som sa vytvoriť aplikáciu, ktorá bude slúžiť na správu financií určených na cestovanie. Umožní používateľom sledovať, kategorizovať a spravovať ich výdavky, a tým im pomôže udržať prehľad nad svojím rozpočtom.

Bolo potrebné zabezpečiť pridávanie jednotlivých ciest a informácie o nich (názov destinácie, stanovený rozpočet, začiatok a koniec trvania pobytu, možnosť pridať poznámky) a zároveň umožniť pridávať k jednotlivým cestám výdavky a informácie o nich (výška výdavku, kategória, spôsob platby, dátum, poznámky) a následne informovať užívateľa o aktuálnom stave financií pre danú cestu.

Podobné aplikácie a v čom sa moja aplikácia líši

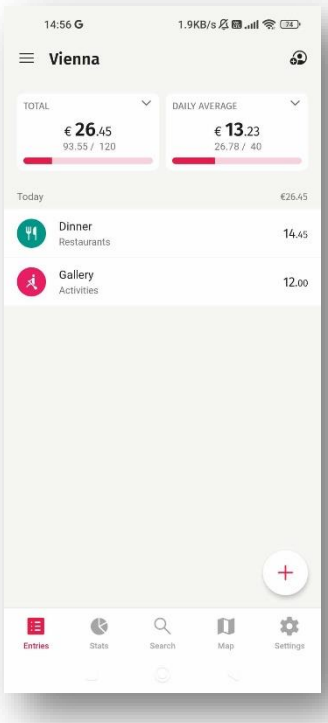
Našla som dve aplikácie podobné mojej – **TravelSpend** a **TrabeePocket**

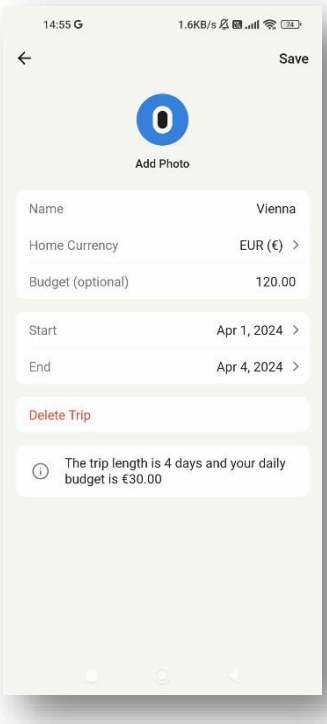
- **TravelSpend**

  TravelSpend je aplikácia na manažovanie výdavkov počas cestovania. Umožňuje kontrolovať výdavky, a tak mať lepší prehľad o svojich aktuálnych financiách určených na cestovanie. Je dobrým pomocníkom pri plánovaní výletov.

  Funkcie:

  - vytvorenie novej cesty (krajina, mena, dĺžka pobytu, nastavenie rozpočtu)
  - zaznamenávanie výdavkov (zaradenie do kategórií, pridanie konkrétneho miesta, poznámok, spôsob platby...)
  - zobrazenie výdavkov číselne alebo graficky podľa zvoleného filtra, prepočítava, koľko je ešte možné minúť do nastaveného limitu
  - prepočítavanie mien podľa kurzov
  - zdieľanie informácií o výdavkoch medzi ľuďmi, ktorí spolu cestujú – rozdelenie platieb



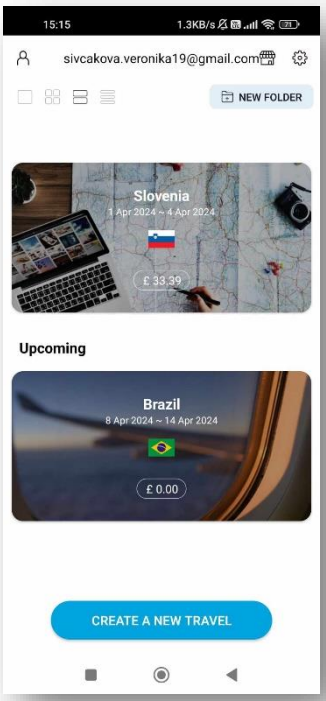


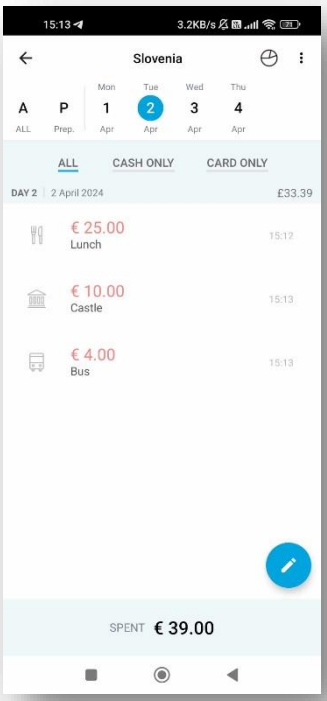
- **TrabeePocket**

TrabeePocket je aplikácia navrhnutá pre cestovateľov, aby im pomohla organizovať a sledovať ich výdavky. Okrem praktickosti, aplikácia ponúka možnosť výzorovo si ju prispôsobiť – zmeniť farby, obrázky ciest, výdavkov... Viaceré funkcie sú platené.

Funkcie:

- vytvorenie novej cesty (krajina, mena sa nastaví automaticky podľa krajiny, dĺžka výletu, možnosť nastavenia fotky reprezentujúcej cestu, pridanie poznámky)
- zaznamenávanie výdavkov (výška výdavku, spôsob platby, zaradenie výdavku do kategórie, poznámka, pridanie fotografie)
- zobrazovanie výdavkov (všetky spolu alebo podľa spôsobu platby), možnosť zobraziť aj graficky, ale len v platenej verzii
- obsahuje zoznam kurzov pre jednotlivé krajiny
- ...

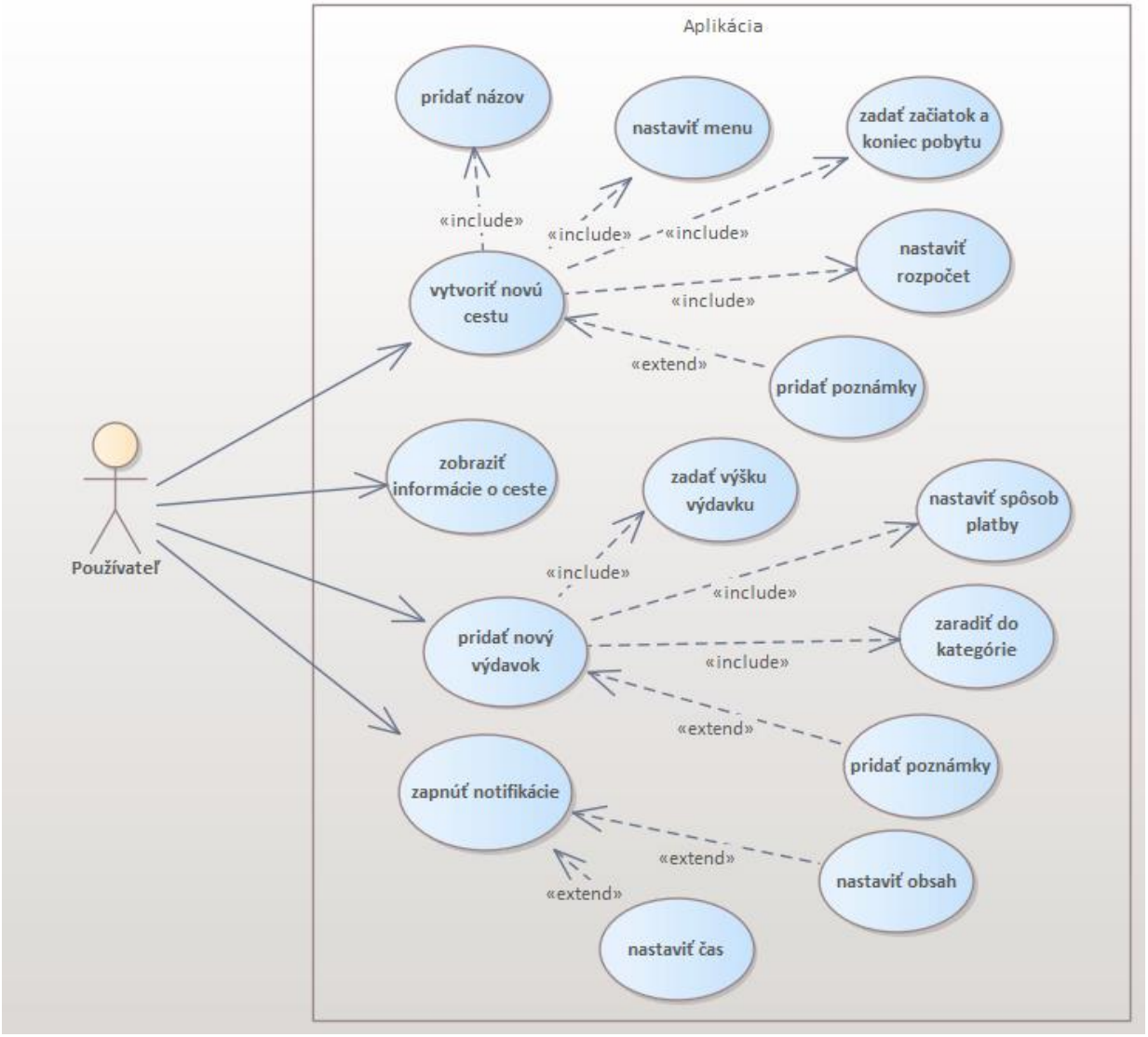




Aplikácie majú spolu s mojou rovnaký základ a zmysel – pomáhať kontrolovať financie počas cestovania. Avšak aplikácie, ktoré som uviedla vyššie sú komplexnejšie, majú veľa funkcií a sú aj vizuálne viac prepracované.

Návrh riešenia problému

Na nasledujúcom obrázku sa nachádza Use Case model zobrazujúci aktivity, ktoré môže vykonávať používateľ aplikácie. Tento UseCase model bol vytvorený počas návrhu riešenia mojej aplikácie ešte pred začatím implementácie. Podarilo sa mi naimplementovať takmer všetky funkcie z UseCasu, nepodarilo sa mi urobiť notifikácie a možnosť voľby meny.



Návrh aplikácie

Aplikáciu tvoria štyri obrazovky

- domovská obrazovka – zobrazuje zoznam ciest, v prípade, že je prázdny zobrazuje o tom informáciu s popisom, ako novú cestu pridať
- obrazovka pre vytvorenie novej cesty – je možné sa na ňu dostať z domovskej obrazovky po kliknutí tlačidla +, sú tu polia pre zadanie vstupov k ceste
- obrazovka s detailami cesty – vypísané informácie o ceste a jej výdavky, možnosť pridávať výdavky
- obrazovka pre vytvorenie nového výdavku – polia pre zadanie vstupov k novému výdavku

Pri tvorbe aplikácie som využila databázu. V databáze sú dve tabuľky – tabuľka ciest a tabuľka výdavkov. Tabuľka výdavkov má cudzí kľúč id cesty, ktorý sa odkazuje na id cesty z tabuľky ciest – tak viem pridávať výdavky pre jednotlivé cesty.

Popis implementácie

Pri tvorbe mojej aplikácie som využila tieto komponenty:

- **Navigation**
  Pre navigáciu v aplikácii som si vytvorila navigačný graf, ktorý definuje, na ktorú obrazovku je možné prejsť z danej obrazovky.

```
@Composable
fun AppNavigationGraph(
    navController: NavHostController,
    modifier: Modifier = Modifier
) {
    NavHost(
        navController = navController,
        startDestination = Home.route,
        modifier = modifier
    ) { this: NavGraphBuilder
        composable(route = Home.route) { this: AnimatedContentScope  it: NavBackStackEntry
            HomeScreen(
                navigateToItemEntry = {navController.navigate(AddTravel.route)},
                navigateToTravelDetails = {navController.navigate( route: "${InfoAboutTravel.route}/${it}")}
            )
        }
        composable(route = AddTravel.route) { this: AnimatedContentScope  it: NavBackStackEntry
            NewTravelScreen(
                navigateBack = { navController.popBackStack() },
                onNavigateUp = { navController.navigateUp() }
            )
        }
```

Room

V aplikácii využívam Room ako databázu pre ukladanie dát o ceste a výdavkoch. Mám dve tabuľky – tabuľku ciest a tabuľku výdavkov. Tabuľka výdavkov má cudzí kľúč id cesty, ktorý sa odkazuje na id cesty z tabuľky ciest.

V aplikácii som vytvárala migrácie vzhľadom na to, že som menila štruktúru databázy – prvá migrácia bolo pridanie stĺpca do tabuľky ciest, ďalšia zahŕňala vytvorenie novej tabuľky a ostatné znovu vytváranie stĺpcov.

```
@Database(entities = [Travel::class, Expense::class], version = 5, exportSchema = false)
abstract class YourTravelsDatabase : RoomDatabase() {

    Veronika Sivcakova
    abstract fun travelDao() : TravelDao
    Veronika Sivcakova
    abstract fun expenseDao() : ExpenseDao
    Veronika Sivcakova
    companion object {
        @Volatile
        private var Instance: YourTravelsDatabase? = null

        Veronika Sivcakova
        fun getDatabase(context: Context): YourTravelsDatabase {
            return Instance ?: synchronized( lock: this) {
                Room.databaseBuilder(context, YourTravelsDatabase::class.java,  name: "travel_database")
                    .addMigrations(MIGRATION_1_2, MIGRATION_2_3, MIGRATION_3_4, MIGRATION_4_5)
                    .fallbackToDestructiveMigration()
                    .build() YourTravelsDatabase
                    .also{ Instance = it}
            }
        }
```

ViewModel

V aplikácii som pre každú obrazovku vytvorila ViewModel, ktorý uchováva jej stav.

```
Veronika Sivcakova
class HomeViewModel(travelsRepository: TravelsRepository) : ViewModel() {
    val homeUiState: StateFlow<HomeUiState> =
        travelsRepository.getAllTravelsStream().map { HomeUiState(it) }
            .stateIn(
                scope = viewModelScope,
                started = SharingStarted.WhileSubscribed(TIMEOUT_MILLIS),
                initialValue = HomeUiState()
            )
    Veronika Sivcakova
    companion object {
        private const val TIMEOUT_MILLIS = 5_000L
    }
}

Veronika Sivcakova
data class HomeUiState(val travelList: List<Travel> = listOf())
```

O vytvorenie inštancií ViewModelov pre všetky obrazovky sa stará AppViewModelProvider.

Zoznam použitých zdrojov

Logo aplikácie:

https://logo-maker.freelogodesign.org/en/logo/edit/8e765988d09e4e10967544ca7bfa9cbc?template=8305ca1ede234947952f6194d21527d0&category=36&companyName=Your%20Travels

Logiku navigácie, viewModely a niektoré časti obrazoviek som robila podľa aplikácie z codelabu:

https://developer.android.com/codelabs/basic-android-kotlin-compose-persisting-data-room#0

Rozbaľovací zoznam som robila podľa:

https://medium.com/@itsuki.enjoy/android-kotlin-jetpack-compose-dropdown-selectable-list-menu-b7ad86ba6a5a

Radio buttony som robila podľa:

https://developer.android.com/codelabs/basic-android-kotlin-compose-test-cupcake#0

Logo aplikácie som pridávala podľa:

https://www.youtube.com/watch?v=bJjHgWjiAKw