Fitness journey
semestrálna práca

Fakulta: Fakulta riadenia a informatiky

Program: Informatika a riadenie

Predmet: VAMZ

Zadanie a špecifikácia: Vytvorenie mobilnej aplikácie

Vypracoval: Dominik Dírbák

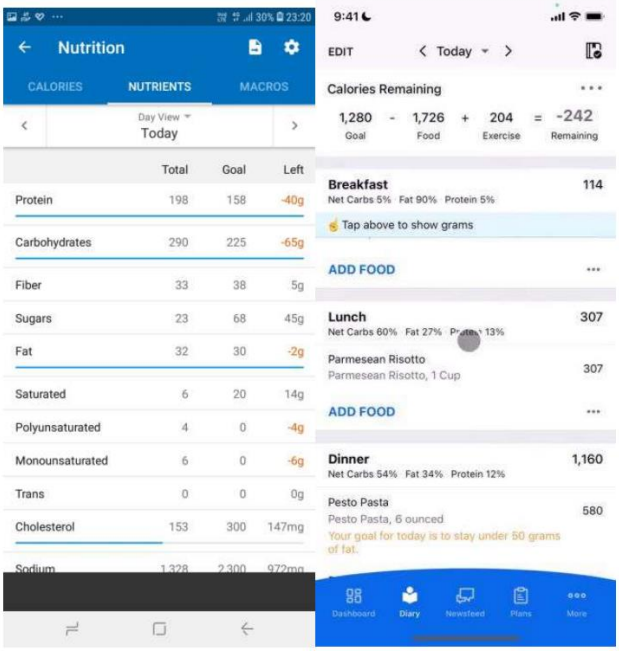
Skupina: 5ZYR24

Contents

1 PREHĽAD APLIKÁCIÍ S PODOBNÝM ZAMERANÍM:

1.1 MyFitnessPal:

MyFitnessPal je populárna aplikácia na sledovanie výživy, ktorá používateľom umožňuje sledovať príjem potravy a cvičenie. Má veľkú databázu potravín a ponúka personalizované výživové poradenstvo na základe vašich cieľov.



1.2 Lose It!:

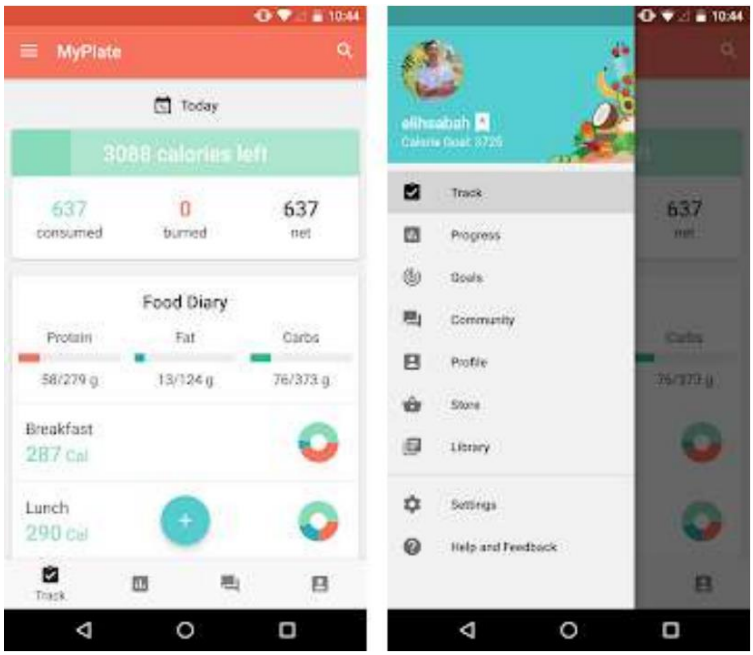
Lose It! je ďalšia aplikácia na sledovanie výživy, ktorá pomáha používateľom stanoviť si ciele, sledovať príjem potravy a cvičenie a spojiť sa s komunitou ostatných používateľov.

1.3 Calory:

Calory je výživová aplikácia, ktorá pomáha používateľom sledovať ich príjem kalórií a ponúka personalizované odporúčania na základe ich cieľov. Má tiež veľkú databázu potravín a ponúka plánovanie jedál a návrhy receptov.

1.4 MyPlate by Livestrong:

MyPlate by Livestrong je výživová aplikácia, ktorá pomáha používateľom sledovať príjem potravín a ponúka personalizované odporúčania na základe ich cieľov. Ponúka aj sledovanie cvičenia a spája používateľov s komunitou ostatných používateľov.



2 ANALÝZA NAVRHOVANEJ APLIKÁCIE:

2.1 Funkčnosť:

Aplikácia by mala mať jasný a komplexný súbor funkcií, ktoré používateľom pomôžu sledovať ich denný príjem potravy, stanoviť si výživové ciele, nájsť zdravé recepty a získať personalizované odporúčania. Aplikácia by tiež mala poskytovať presné informácie o výživových hodnotách a mala by si poradiť s rôznymi stravovacími preferenciami a obmedzeniami.

2.2 Používateľské rozhranie:

Používateľské rozhranie by malo byť intuitívne a užívateľsky prívětivé, s jednoduchou navigáciou a jasnými pokynmi. Aplikácia by mala mať aj vizuálne príťažlivý dizajn s farebne odlíšenými systémami a grafmi, ktoré používateľom pomôžu sledovať ich pokrok.

2.3 Personalizácia:

Aplikácia by mala byť schopná poskytovať personalizované odporúčania na základe údajov, preferencií a cieľov používateľa. To môže zahŕňať prispôsobené plány stravovania, návrhy receptov a výživové ciele.

2.4 Ukladanie údajov:

Aplikácia bude musieť uchovávať údaje používateľov, ako sú napríklad výživové ciele, plány stravovania a denné záznamy o príjme potravín. Na ukladanie údajov zvážte použitie cloudovej databázy, napríklad Firebase alebo MongoDB.

3 NÁVRH ARCHITEKTÚRY APLIKÁCIE:

3.1 Zaznamenávanie stravy:

Aplikácia umožní používateľom zaznamenávať príjem potravy vyhľadávaním potravín a receptov alebo skenovaním čiarových kódov. Aplikácia bude poskytovať presné nutričné informácie o každej potravine a umožní používateľom sledovať ich denný príjem potravín.

3.2 Plánovanie jedál:

Aplikácia poskytne používateľom možnosť plánovať svoje jedlá vopred vytvorením plánu jedál na týždeň alebo deň. Používatelia si môžu vybrať potraviny a recepty z databázy alebo si vytvoriť vlastné jedlá.

3.3 Výživová analýza:

Aplikácia bude analyzovať nutričný obsah prijatých potravín používateľa a poskytovať spätnú väzbu o tom, či používateľ plní svoje nutričné ciele. Aplikácia poskytne odporúčania na úpravu príjmu potravín používateľa, aby splnil svoje ciele.

3.4 Personalizácia:

Aplikácia personalizuje skúsenosti používateľa poskytovaním odporúčaní na základe jeho cieľov, preferencií a stravovacích obmedzení.

4 OBRAZOVKY

4.1 Popis triedy CalendarScreen

Trieda CalendarScreen je komponent, ktorý zobrazuje kalendár a štatistiky spojené s vybraným dátumom. Táto obrazovka využíva viaceré pohľadové modely (ViewModel) na získavanie údajov o cvičení a jedle pre konkrétny dátum.

4.2 Popis triedy CaloriesScreen

Trieda CaloriesScreen je komponent, ktorý zobrazuje obrazovku na sledovanie kalórií skonzumovaných používateľom. Používa rôzne komponenty na zobrazenie počítadla kalórií, zoznamu potravín a umožňuje pridávať nové potraviny cez dialógové okno.

4.3 Popis triedy SettingsScreen

Trieda SettingsScreen je komponent, ktorý zobrazuje obrazovku s nastaveniami používateľa. Táto obrazovka umožňuje používateľom meniť ich profilové údaje, nastavenia limitov pre cvičenie a kalórie, a tiež notifikácie.

4.4 Popis triedy SportsScreen

Trieda SportsScreen je komponent, ktorý zobrazuje obrazovku pre športové aktivity používateľa. Poskytuje funkcie na sledovanie času stráveného cvičením a zobrazuje zoznam aktivít.

5 NAVIGÁCIA

5.1 Popis triedy MainActivity

Trieda MainActivity je hlavnou aktivitou aplikácie, ktorá sa stará o inicializáciu a spúšťanie jednotlivých komponentov a obrazoviek. Používa sa na nastavenie obsahu aplikácie a správu navigácie medzi obrazovkami.

5.2 Popis triedy AppNavHost

Trieda AppNavHost je zodpovedná za správu navigácie v aplikácii pomocou Jetpack Compose Navigation. Definuje rôzne obrazovky (komponáty) a ich navigačné trasy. Táto trieda umožňuje používateľom prepínať medzi rôznymi sekciami aplikácie, ako sú kalórie, športové aktivity, kalendár, nastavenia a zoznam aktivít.

```
package com.example.vamzsem.ui.screens


import ...

SaKlasneMoj
@Composable
fun AppNavHost(
    navController: NavHostController = rememberNavController(),
    foodViewModel: FoodViewModel = viewModel(),
    timerViewModel: TimerViewModel = viewModel(),
    profileViewModel: ProfileViewModel = viewModel()
) {
    NavHost(navController = navController, startDestination = Screen.Calories.route) {
        composable(Screen.Calories.route) {
            CaloriesScreen(foodViewModel = foodViewModel, navController = navController, profileViewModel = profileViewModel)
        }
        composable(Screen.Sports.route) {
            SportsScreen(navController = navController, timerViewModel = timerViewModel,  profileViewModel = profileViewModel)
        }
        composable(Screen.Calendar.route) {
            CalendarScreen(navController = navController, timerViewModel = timerViewModel, foodViewModel = foodViewModel,profileViewModel = profileViewModel)
        }
        composable(Screen.Settings.route) {
            SettingsScreen(navController = navController, profileViewModel = profileViewModel)
        }
        composable( route: "activities") {
            ActivityListScreen(timerViewModel = timerViewModel)
        }
    }
}
```

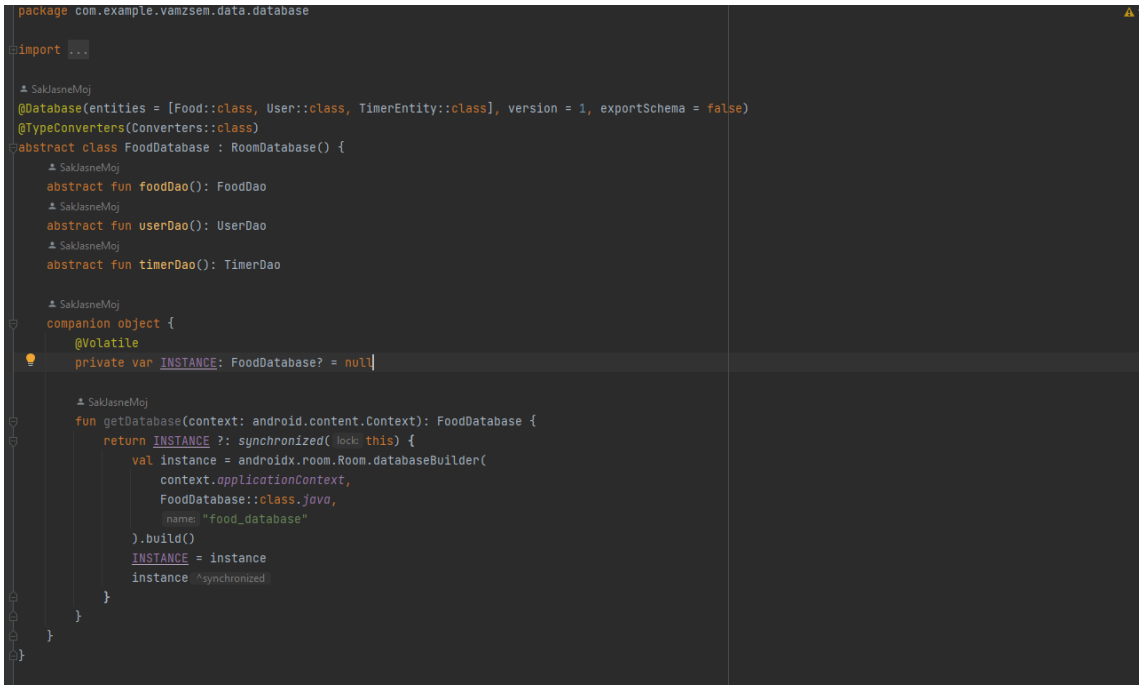
5.3 Popis triedy NavigationDrawer

Kód obsahuje dva hlavné kompozitné komponenty, DrawerHeader a DrawerBody, ktoré sú súčasťou používateľského rozhrania zásuvky (drawer). Tieto komponenty sú zodpovedné za zobrazenie profilovej hlavičky a položiek menu v zásuvke, ktorá umožňuje používateľovi navigáciu medzi rôznymi sekciami aplikácie.

6 ROOM DATABÁZA

6.1 Popis triedy FoodDatabase

Trieda FoodDatabase je abstraktná trieda, ktorá definuje Room databázu pre aplikáciu. Táto databáza zahŕňa entity pre potraviny (Food), používateľov (User) a časovače (TimerEntity). Okrem toho poskytuje prístup k príslušným DAO (Data Access Object) rozhraniam.



```
package com.example.vamzsem.data.database

import ...

SakJasneMoj
@Database(entities = [Food::class, User::class, TimerEntity::class], version = 1, exportSchema = false)
@TypeConverters(Converters::class)
abstract class FoodDatabase : RoomDatabase() {
    SakJasneMoj
    abstract fun foodDao(): FoodDao
    SakJasneMoj
    abstract fun userDao(): UserDao
    SakJasneMoj
    abstract fun timerDao(): TimerDao

    SakJasneMoj
    companion object {
        @Volatile
        private var INSTANCE: FoodDatabase? = null

        SakJasneMoj
        fun getDatabase(context: android.content.Context): FoodDatabase {
            return INSTANCE ?: synchronized( lock: this) {
                val instance = androidx.room.Room.databaseBuilder(
                    context.applicationContext,
                    FoodDatabase::class.java,
                     name: "food_database"
                ).build()
                INSTANCE = instance
                instance ^synchronized
            }
        }
    }
}
```

6.2 Popis triedy TimerDao

Trieda TimerDao je rozhranie pre prístup k databáze, ktoré je definované pomocou Room knižnice pre Android. Toto rozhranie umožňuje vykonávať CRUD operácie (Create, Read, Update, Delete) nad entitami TimerEntity v databáze.

6.3 Popis triedy FoodDao

FoodDao je Data Access Object (DAO) rozhranie pre entitu Food, ktorá predstavuje databázovú tabuľku food. Toto rozhranie poskytuje metódy na vykonávanie CRUD operácií (Create, Read, Update, Delete) a ďalšie dotazy pre entitu Food v databáze.

6.4 Popis triedy FoodRepository

Táto trieda je pre správu operácií s jedlami v databáze. Používa FoodDao na vykonávanie databázových operácií a umožňuje prístup k údajom o potravinách na základe rôznych kritérií.

6.5 Popis triedy TimerRepository

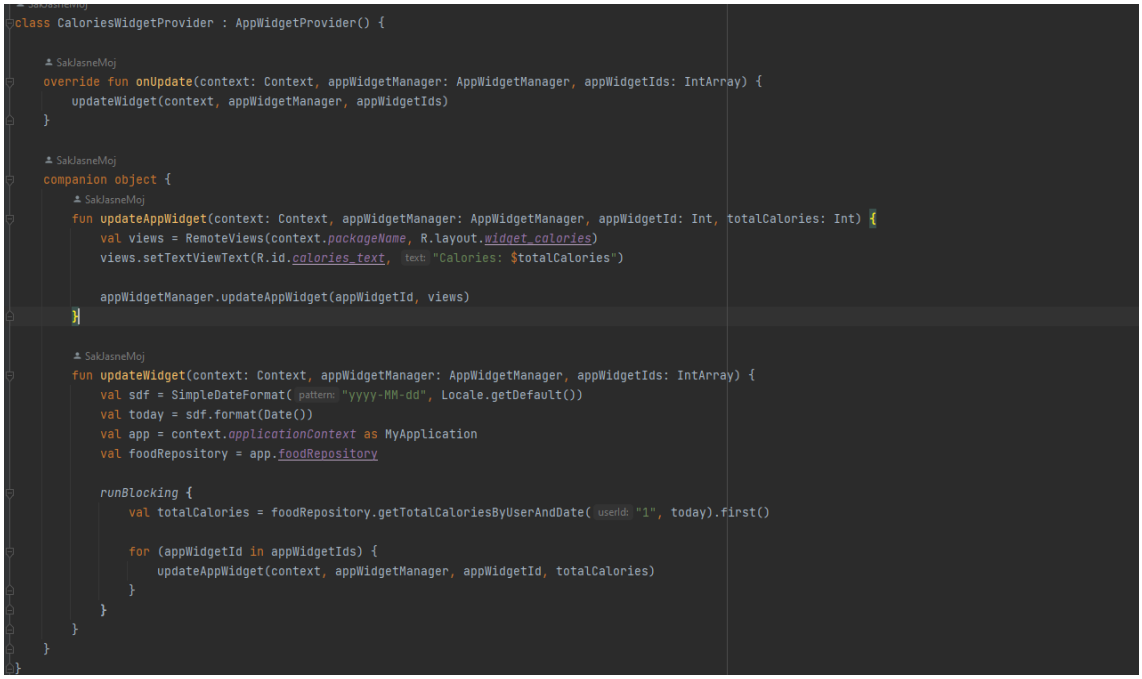
Táto trieda je pre správu operácií s časovačmi v databáze. Používa TimerDao na vykonávanie databázových operácií a umožňuje prístup k údajom o časovačoch na základe rôznych kritérií. Taktiež využíva SharedPreferences na uloženie a načítanie stavu časovača.

```
SakJasneMoj
fun getSavedTime(): Int {
    val sharedPref = context.getSharedPreferences( name: "timer_prefs", Context.MODE_PRIVATE)
    return sharedPref.getInt( key: "saved_time",  defValue: 100 * 60) // Default to max time
}

SakJasneMoj
fun isTimerRunning(): Boolean {
    val sharedPref = context.getSharedPreferences( name: "timer_prefs", Context.MODE_PRIVATE)
    return sharedPref.getBoolean( key: "is_running",  defValue: false)
}

SakJasneMoj
fun saveTime(seconds: Int) {
    val sharedPref = context.getSharedPreferences( name: "timer_prefs", Context.MODE_PRIVATE)
    with(sharedPref.edit()) {
        putInt("saved_time", seconds)
        apply()
    }
}

SakJasneMoj
fun saveTimerRunningState(isRunning: Boolean) {
    val sharedPref = context.getSharedPreferences( name: "timer_prefs", Context.MODE_PRIVATE)
    with(sharedPref.edit()) {
        putBoolean("is_running", isRunning)
        apply()
    }
}
```

7 WIDGET

7.1 Popis triedy CaloriesWidgetProvider

Trieda CaloriesWidgetProvider je widget poskytovateľ, ktorý zobrazuje kalórie na domovskej obrazovke pomocou AppWidgetProvider. Táto trieda obsahuje logiku pre aktualizáciu widgetu s údajmi o kalóriách zadaného používateľa. Používa RemoteViews na aktualizáciu vzhľadu widgetu a kotlinx.coroutines na získanie údajov z FoodRepository v korutine.



```
class CaloriesWidgetProvider : AppWidgetProvider() {

    SakJasneMoj
    override fun onUpdate(context: Context, appWidgetManager: AppWidgetManager, appWidgetIds: IntArray) {
        updateWidget(context, appWidgetManager, appWidgetIds)
    }

    SakJasneMoj
    companion object {
        SakJasneMoj
        fun updateAppWidget(context: Context, appWidgetManager: AppWidgetManager, appWidgetId: Int, totalCalories: Int) {
            val views = RemoteViews(context.packageName, R.layout.widget_calories)
            views.setTextViewText(R.id.calories_text,  text: "Calories: $totalCalories")

            appWidgetManager.updateAppWidget(appWidgetId, views)
        }

        SakJasneMoj
        fun updateWidget(context: Context, appWidgetManager: AppWidgetManager, appWidgetIds: IntArray) {
            val sdf = SimpleDateFormat( pattern: "yyyy-MM-dd", Locale.getDefault())
            val today = sdf.format(Date())
            val app = context.applicationContext as MyApplication
            val foodRepository = app.foodRepository

            runBlocking {
                val totalCalories = foodRepository.getTotalCaloriesByUserAndDate( userId: "1", today).first()

                for (appWidgetId in appWidgetIds) {
                    updateAppWidget(context, appWidgetManager, appWidgetId, totalCalories)
                }
            }
        }
    }
}
```

7.2 Popistried CaloriesWidgetServic a CaloriesRemoteViewsFactory

Tieto triedy zabezpečujú funkčnosť widgetu pre zobrazenie kalórií na domovskej obrazovke. Trieda CaloriesWidgetService je služba pre widget, ktorá vytvára inštanciu RemoteViewsFactory, zatiaľ čo CaloriesRemoteViewsFactory implementuje logiku pre získanie údajov a vytváranie zobrazení pre widget.

8 ZDROJE

https://www.youtube.com/@PhilippLackner

https://www.geeksforgeeks.org/building-ui-using-jetpack-compose-in-android/