Kristína Uglerová

ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

ProdActivity

Semestrálna práca z predmetu Vývoj aplikácií pre mobilné zariadenia

Vypracoval: Kristína Uglerová
Študijná skupina: 5ZYI23
Cvičiaci: doc. Ing. Patrik Hrkút PhD.
Termín cvičenia: streda, bloky 1-2

v Žiline dňa 11.6.2024

Obsah

Popis a analýza riešeného problému

1. Špecifikácia zadania a definovanie problému

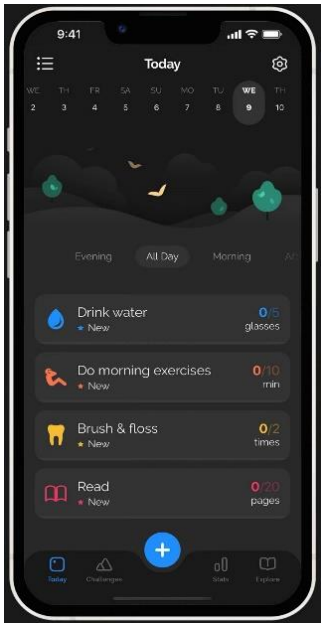
Cieľom našej semestrálnej práce bolo vytvoriť mobilnú aplikáciu slúžiacu na podporu produktivity. Implementovali sme možnosť vytvoriť jednorazové alebo dlhodobé ciele. Pre dlhodobé ciele je možné pripočítavať streak – koľko dní na nich používateľ pracoval. Táto možnosť zabezpečuje pre používateľa každodennú dávku motivácie. Pre lepší prehľad aplikácia poskytuje štatistiky ako napríklad počet splnených cieľov, cieľ s najvyšším streakom, graf pre porovnanie cieľov s najvyššími streakmi a podobne. Ďalšou časťou je prehrávač, ktorý umožňuje prehrávanie rôznych farebných šumov, slúžiacich na zvýšenie produktivity, čo používateľovi umožní lepšie sa koncentrovať pri plnení svojich úloh. Aplikácia taktiež poskytuje widget aplikáciu s motivačnými citátmi.

Vďaka týmto funckionalitám ProdActivity má používateľ možnosť urobiť si prehľad vo svojich úlohách a cieľoch a aplikácia taktiež slúži ako motivačný prostriedok pre používateľa.

2. Prehľad dostupných aplikácií podobného zamerania

2.1 Productive

Aplikácia Productive ponúka možnosť spravovania návykov používateľa. Jednotlivé návyky je možné personalizovať zmenou farby a ikony. Taktiež je k nim možné pridávať poznámky. Aplikácia ponúka motivačné prostriedky ako súťaže s inými používateľmi, upozornenia a výzvy. Používateľ má prístup k štatistikám a rôznym odborným článkom o produktivite, ktoré mu ponúkajú možnosť sebapoznania.



V platenej verzii používateľ získa vylepšené vyššie spomenuté funkcionality ako napríklad vylepšené pripomienky, motivujúce štatistiky, možnosť neobmedzeného počtu návykov.

Oproti tejto aplikácii je v našej možné púšťať si rôzne farebné šumy podporujúce produktivitu a taktiež si o nich čítať informácie. ProdActivity tiež okrem dlhodobých cieľov umožňuje vytvárať aj jednorazové ciele a tiež využiť widget s motivačnými citátmi.

2.2 Habit tracker

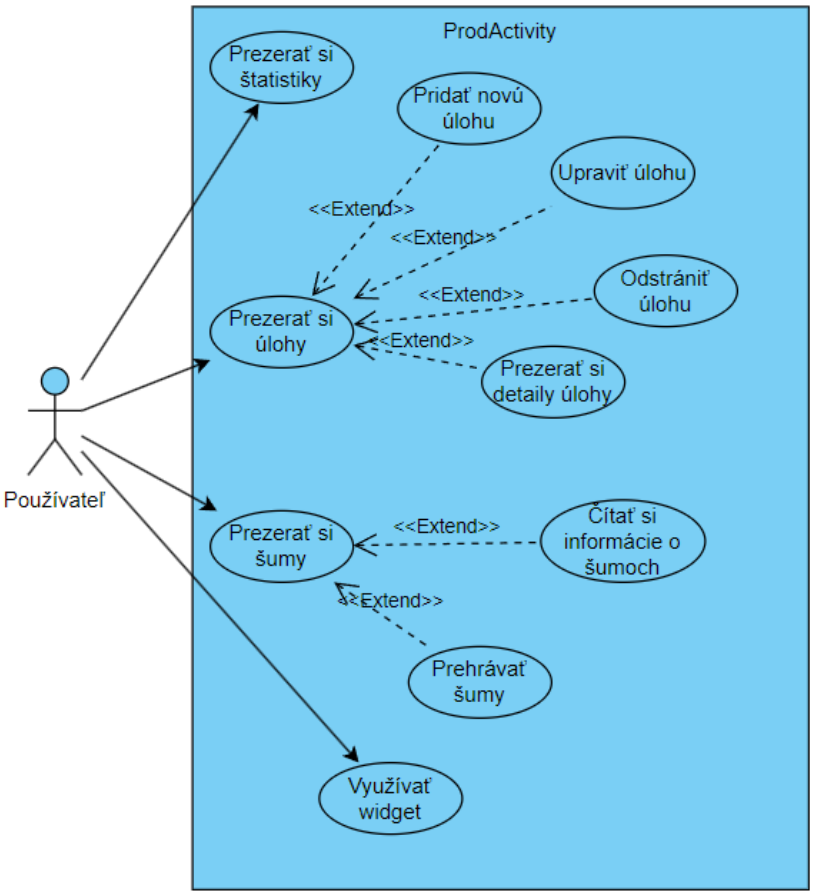
Aplikácia Habit tracker slúži na ukladanie svojich cieľov – dlhodobých návykov. Používateľ si môže vybrať, či sa chce zbaviť zlého návyku, alebo si vybudovať dobrý návyk. Hlavným motivačným prostriedkom pre používateľa by mali byť streaks – počet dní za sebou kedy sa používateľovi podarilo splniť svoj cieľ. Aplikácia posiela používateľovi upozornenia, ktoré mu jeho jednotlivé úlohy pripomínajú. Taktiež poskytuje množstvo štatistík, widgetov, či zdieľanie jednotlivých návykov s priateľmi.

Používateľ si tiež môže zaplatiť prémiovú verziu, ktorá poskytuje funkcionality ako ročný prehľad jednotlivých návykov, uzamknutie aplikácie či možnosť nastavenia viacerých upozornení pre jeden návyk.

Podobne ako pri Productive naša aplikácia navyše ponúka farebné šumy a jednorazové ciele.

Návrh riešenia problému

1. Analýza – diagram prípadov použitia



2. Návrh aplikácie

Aplikácia pozostáva z troch obrazoviek. Pre každú obrazovku je vytvorený ViewModel. Ten zabezpečuje logické operácie pre danú obrazovku. Medzi obrazovkami sa používateľ môže pohybovať pomocou navigácie v spodnej časti obrazovky.

Pre ukladanie a evidenciu údajov v rámci aplikácie sme tiež vytvorili dátové triedy.

Vstupy od používateľa sú riešené pomocou dialógových okien a ciele, ktoré používateľ vytvára sú ukladané do databázy.

Projekt sa skladá z troch základných balíčkov.

Data

V tomto balíku sa nachádzajú všetky triedy, ktoré slúžia na uchovávanie dát – ide o dátové triedy.

Goal – Informácie o cieľoch vytvorených používateľom, je zároveň entitou v databáze.

Noise – Informácie pre jednotlivé farebné šumy.

Ďalej sa v tomto balíku nachádza aj balík database, v ktorom sa nachádzajú všetky triedy a rozhrania potrebné pre funkčnosť databázy. Na jeho implementáciu sme využili codelab, s ktorým sme pracovali na hodine.

UI

V balíku ui sa nachádza celé grafické rozhranie aplikácie – obrazovky a tiež widget.

GoalsScreen – Táto obrazovka zobrazuje prehľad cieľov používateľa, tie môže pridávať pomocou floating action button a dialógového okna. Používateľ má tiež možnosť ciele upraviť, odstrániť alebo dokončiť. Pri dlhodobých cieľoch tiež vidí svoj streak a môže si ho navýšiť.

NoisesScreen – Slúži ako prehrávač farebných šumov, okrem toho má používateľ možnosť prečítať si o nich informácie zobrazované pomocou dialógového okna.

StatsScreen – Zobrazuje štatistiky o cieľoch používateľa a tiež graf cieľov s najvyšším počtom streakov.

ProdActivityNavigation – Ide o menu v spodnej časti, ktoré slúži na navigáciu medzi obrazovkami popísanými vyššie.

V tomto balíku sa tiež nachádza balík Widget. Nachádza sa v ňom widget aplikácia, zobrazujúca náhodne generované motivačné citáty, ktoré sú uložené v res/values/arrays.

ViewModel

GoalsViewModel – Zabezpečuje logické operácie pre GoalsScreen. Spolupracuje s databázou – vkladá, updateuje a maže z nej dáta.

NoiseViewModel – Spolu s MediaController zabezpečuje správne prehrávanie farebných šumov v NoisesScreen.

StatsViewModel – Podobne ako GoalsViewModel spolupracuje s databázou, čerpá z nej potrebné dáta pre jednotlivé štatistiky zobrazené v StatsScreen.

Okrem týchto balíkov sa nachádza ešte MainActivity a ProdActivityApplication, ktoré zabezpečujú chod aplikácie.

Popis implementácie

Ako prvé sme v rámci implementácie aplikácie ProdActivity vytvorili navigáciu. Následne sme začali s implementáciou jednotlivých obrazoviek a k nim patriacim ViewModelom. Podľa potreby sme tiež vytvárali dátové triedy a databázu. Aplikácia bola naprogramovaná pomocou Jetpack Compose v jazyku Kotlin. Pri jej implementácii sme sa snažili dbať na funkcionalitu aplikácie popísanú v kontrolnom bode.

V aplikácii sú implementované nasledujúce prvky:

Aplikácia obsahuje 3 obrazovky a ku každej je implementovaný aj ViewModel. Všetky tieto časti sú popísané vyššie. Okrem ViewModel sme z AndroidX komponentov tiež využili Room databázu a Navigation. Za pomoci Jetpack Glance sme vytvorili widget aplikáciu.

```
class GoalsViewModel(private val goalsRepository: GoalsRepository) : ViewModel() {

    private val _parameters: MutableStateFlow<GoalEditDialogParameters> = MutableStateFlow(
        GoalEditDialogParameters()
    )
    val parameters = _parameters.asStateFlow()
    val goalsUiState: StateFlow<GoalsUiState> =
        goalsRepository.getAllItemsStream().map { GoalsUiState(it) }
            .stateIn(
                scope = viewModelScope,
                started = SharingStarted.WhileSubscribed(TIMEOUT_MILLIS),
                initialValue = GoalsUiState()
            )
```

Príklad jedného z implementovaných ViewModelov – GoalsViewModel

```
    NavHost(
        navController = navController,
        startDestination = ProdActivityScreen.NoisesScreen.name,
        modifier = Modifier
            .fillMaxSize()
            .padding(innerPadding)
    ) { this: NavGraphBuilder
        composable(route = ProdActivityScreen.NoisesScreen.name) { this: AnimatedContentScope  it: NavBackStackEntry
            NoisesScreen(
                options = noiseViewModel.noises,
                onItemClick = { it: Noise
                    noiseViewModel.updateNoiseItem(it.copy(isPlaying = !it.isPlaying))
                },
                modifier = Modifier.fillMaxHeight()
            )
        }
        composable(route = ProdActivityScreen.GoalsScreen.name) { this: AnimatedContentScope  it: NavBackStackEntry
            GoalsScreen(
                modifier = Modifier.fillMaxHeight()
            )
```

Implementácia NavHost v aplikácii ProdActivity, v rámci každej obrazovky sa zobrazuje ako spodná lišta s ikonkami, pomocou ktorých je možné pohybovať sa medzi nimi.

```
@Database(entities = [Goal::class], version = 2, exportSchema = false)
abstract class GoalDatabase : RoomDatabase() {
    kikau
    abstract fun goalDao(): GoalDao

    kikau
    companion object {
        @Volatile
        private var Instance: GoalDatabase? = null

        kikau
        fun getDatabase(context: Context): GoalDatabase {
            return Instance ?: synchronized( lock: this) {
                Room.databaseBuilder(context, GoalDatabase::class.java,  name: "goal_database") Builder<GoalDatabase>
                    .fallbackToDestructiveMigration()
                    .build() GoalDatabase
                    .also { Instance = it }
            }
        }
    }
}
```

Implementácia Room database v aplikácii ProdActivity zabezpečujúca správu cieľov a úloh vytvorených používateľom. Databáza tiež poskytuje potrebné údaje pre štatistiky v aplikácii.

```
class MyAppWidgetReceiver : GlanceAppWidgetReceiver() {
    override var glanceAppWidget: GlanceAppWidget = QuotesWidget()
}

kikau
class QuotesWidget : GlanceAppWidget() {
    kikau
    override suspend fun provideGlance(context: Context, id: GlanceId) {
        val quotes = context.resources.getStringArray(R.array.quotes)
        val randomQuote = quotes[Random.nextInt(quotes.size)]
        provideContent {
            MyContent(randomQuote)
        }
    }
```

Implementácia widget aplikácie v aplikácii ProdAcitivty.

Zoznam použitých zdrojov

Pri implementácii aplikácie sme pracovali iba s Andorid developer dokumentáciou a kurzom predmetu.