ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Semestrálna práca

VÝVOJ APLIKÁCIÍ PRE MOBILNÉ ZARIADENIA

vypracovala: Emma Boorová

študijná skupina: 5ZYI26

cvičiaci: doc. Ing. Patrik Hrkút, PhD.

termín cvičenia: štvrtok bloky 4-5, šk. rok 2023/2024

v Žiline dňa 10.6.2024

Obsah

1. Analýza riešeného problému

1.1. Koncept

Aplikácia „Random Productivity“ je nástroj na správu a sledovanie produktivity z pohľadu úloh. Hlavným cieľom aplikácie je pomôcť používateľom zostať motivovaný a organizovaný pomovou sledovania plnených úloh, stanovenia cieľov a vizualizácie pomocou rôznych štatištík. Má taktiež možnosť vybrania náhodnej úlohy keď sa používateľ nevie rozhodnút alebo potrebuje motiváciu navyše.

1.2. Funkcie

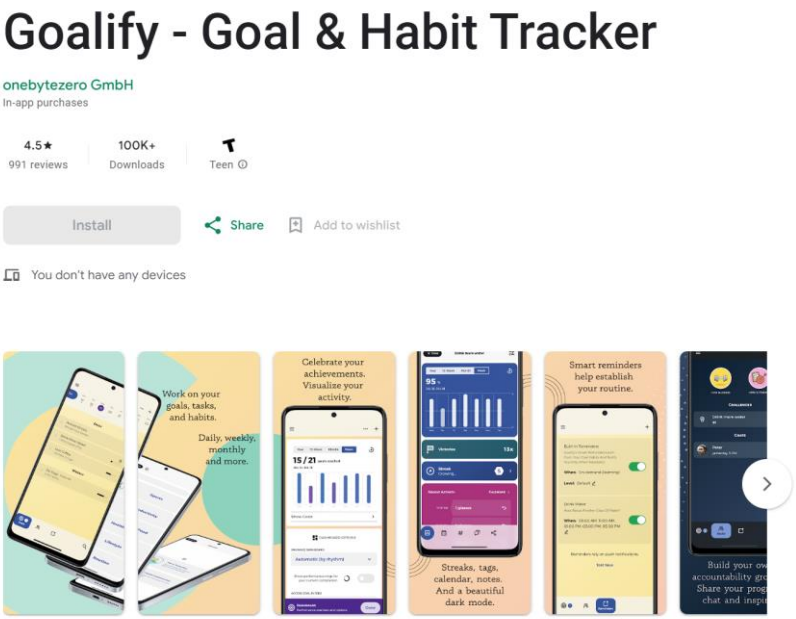
Kľúčové funkcie aplikácie sú vytváranie a správa úloh, možnosť náhodného výberu úlohy, na ktorej sa má pracovať, a detajlné zobrazenia a nastavenia jednotlivých úloh. Tieto používateľovi umožňujú sledovať čas strávený na úlohe, nastaviť jej prioritu, termín a iné. V rámci aplikácie sú ťaktiež nástroje na nastavenie konkrétnych cieľov pre úlohy a prehľad štatistík, vrátane týždenných a mesačných.

1.3. Praktické využitie

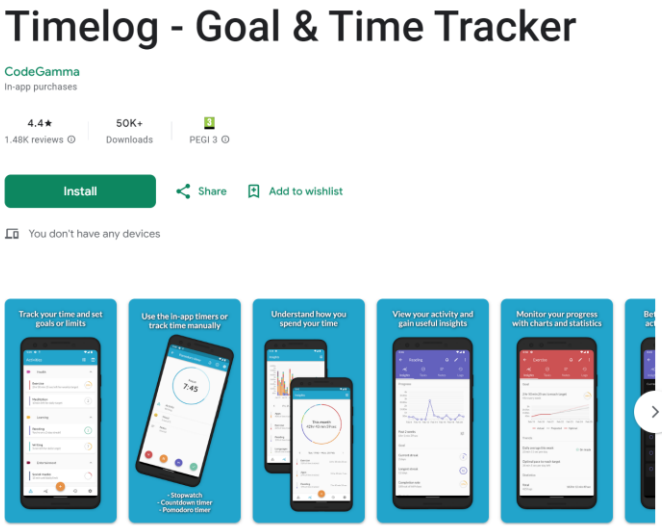
Aplikácia je praktická hlavne pre používateľov, ktorí chcú manažovať svoje osobné úlohy, koníčky a projekty. Používa metodický prístup, odmeny v potobe grafov a dokončených cielov, ale taktiež pridáva aspekt náhody ktorý môže rozprúdiť motiváciu.

1.4. Analýza podobných aplikácií

Aplíkacií pre produktivitu a budovanie návykov je v Google Play mnoho. Jedným z príkladov je Goalify, ktorý má vyše 100 tisíc stiahnutí. Dokáže vizualizovať dosiahnutie rôznych cieľov zadaných používateľom, zatiaľ čo aplikácia, ktorá je predmetom tejto dokumentácie, „Random Productivity“, bude vytvárať štatistiky pre všetky úlohy. Goalify taktiež ponúka možnosť kalendárov, poznámok a „streaks“ pre sledované ciele.

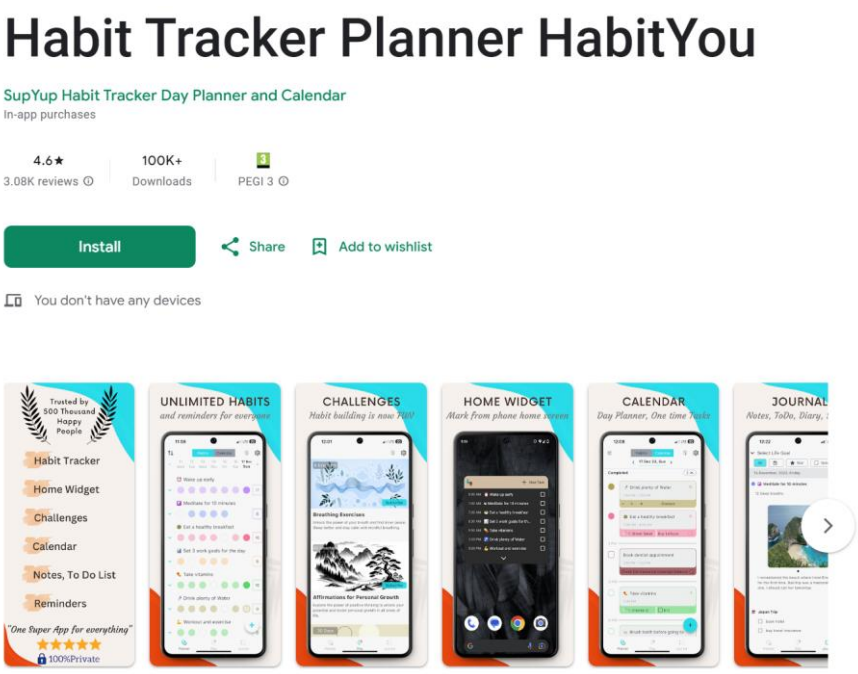


Timelog je aplikácia zameraná hlavne na štatistiku. Strávený čas meria presne v hodinách a minútach, a v tých aj implementuje sledované ciele a osobné výzvy. Rôzne „aktivity“ umožňuje zaradiť do kategórií a zaznačuje aj informácie ako je únava a nálada. Je to jednoduchšia aplikácia so zaujímavými funkciami pre sústredenie a časovanie aktivít.



Tretia podobná aplikácia je najvyššie hodnotená s 4,6/5 hviezdami z vyše tritisíc hodnotení. „HabitYou“ implementuje funkcie kalendára pre ciele, úlohy a produktivtu.

Má ale aj pridané funkcie ako „home widget“, pridávanie poznámok, osobný diár, nastavovanie pripomienok a sledovanie pokroku prostredníctvom grafov. Úlohy zoskupuje podľa kategórií, a má aj platené integrované kurzy a výzvy.



„Random Productivity“ sa najviac líši náhodným prístupom k motivácii. Poskytuje jednoduché možnosti organizácie a cieľov, ale aj náhodu a zábavu v podobe výberu náhodnej úlohy, keď sa používateľ cíti nerozhodný alebo zaseknutý.

2. Návrh riešenia

2.1. Prípady použitia

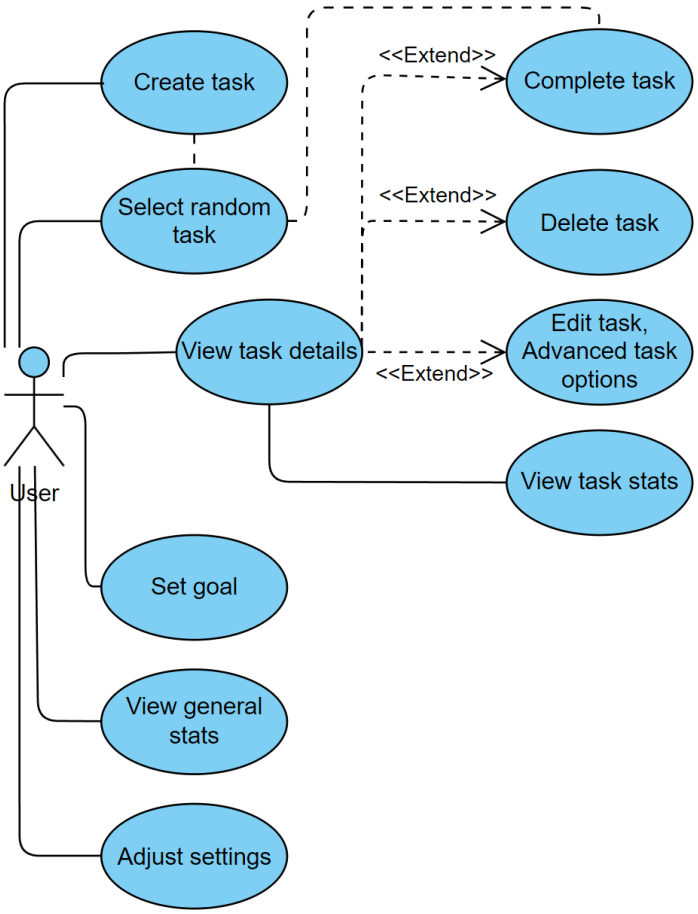
Základnou funkcionalitou aplikácie je vytváranie "tasks" (úloh) používateľom z hlavnej obrazovky aplikácie. Výber náhodnej úlohy je podmienený tým, že je vytvorený istý počet úloh.

Keď je úloha vybraná, používateľ ju môže označiť ako dokončenú, čo ju pripočíta do štatištík a povolí vybranie novej.

Zobrazenie podrobností o úlohe poskytne komplexný prehľad jedinej úlohy. Tu má používateľ prístup k rôznym informáciám, štatistikám a nastaveniam týkajúcim sa úlohy.

Úlohy je možné prispôsobiť prioritou a termínom. Používateľ nôže vylúčiť konkrétne úlohy z náhodného výberu a taktiež sledovať ich pokrok k cieľom, ak je nejaký cieľ nastavený pre danú úlohu. Ak má úloha termín, upozornenia je možné povoliť alebo zakázať. Je taktiež možné vybrať, či chce používateľ merať čas strávený na úlohe v hodinách.

Ako už bolo spomenuté, aplikácia umožňuje nastavenie cieľov pre jednotlivé úlohy. Ciele možno kvantifikovať podľa množstva dokončení alebo celkového počtu strávených hodín. Samostatná obrazovka na prezeranie všeobecných štatistík ponúka prehľad o celkovom výkone používateľa vrátane mesačných a týždenných prehľadov.



2.2. Návrh aplikácie

Návrh aplikácie je zameraný na oddelenie používateľského rozhrania od biznis logiky. Aplikácia využíva pre používateľské rozhrania Jetpack Compose.

Každá zložka používateľského rozhrania je modulárna a nachádza sa v konkrétnom súbore Composable. Zložky, ako sú zoznamy úloh, obrazovky s podrobnosťami a štatistické vizualizácie, sú opakovane použiteľné a udržiavateľné.

Navigácia je riešená pomocou komponentu Navigate zo Jetpack Compose. Aplikácia používa ViewModels na správu údajov súvisiacich s používateľským rozhraním, čo zabezpečí, že používateľské rozhranie zostane aktualizované pri zmenách údajov.

Na lokálne ukladanie údajov sa používa Room s jasne definovanými entitami a rozhraniami DAO na interakciu s databázou.

WorkManager spravuje úlohy na pozadí, napríklad plánovanie notifikácií. Používatelia môžu teda byť upozornení na termíny úloh, aj keď aplikácia nie je spustená. Závislosti sú injektované pomocou Hilt-u. Všetky zdroje vrátane reťazcov sú uložené ako „resource“.

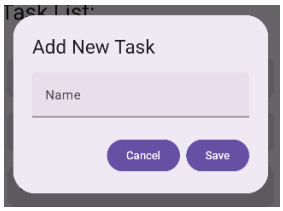
3. Popis implementácie

3.1. Obrazovky

Súbor HomeScreen.kt je vstupným bodom aplikácie. Používa LazyColumn na zobrazenie zoznamu úloh načítaných z TaskViewModel, a taktiež aby bolo pri držaní mobilu na šírku posúvaním možné zobraziť celé používateľské rozhranie obrazovky.

Používatelia môžu pridať novú úlohu kliknutím na tlačidlo, ktoré spustí komponent TaskCreationDialog. Toto dialógové okno príjme názov úlohy a po potvrdení volá viewModel.addTask(taskEntity).

Aby sa dialógové okno nezavrelo pri prevrátení mobliu, používa namiesto remember „rememberSaveable“. Tak isto túto možnosť používa samotné dialógové okno, aby sa nevymazala informácia vo formulári pri otočení displeja.



Po stlačení tlačidla sa spustí náhodný výber, ktorý je riadený stavovou premennou isShuffling. Aktuálna úloha sa potom zobrazí sa v hornej časti obrazovky a objavia sa možnosti označiť ju za dokončenú alebo ju opustiť v príslušnej obrazovke TaskDetailScreen.

Zoznam úloh sa vykresľuje pomocou komponentu TaskList, ktorý iteruje cez úlohy a každú z nich zobrazuje pomocou komponentu TaskItem.

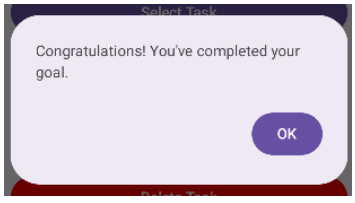
TaskDetailScreen.kt poskytuje podrobné zobrazenie jednej úlohy („task“). Pri prechode na túto obrazovku sa ID úlohy odovzdáva ako argument.

Obrazovka umožňuje používateľom meniť názov úlohy, prioritu a dátum splatnosti a nastavovať alebo resetovať ciele. Získava podrobnosti o úlohe pomocou viewModel.loadTask(taskId) a zobrazuje atribúty, ako je názov, priorita a termín. Vstupy sa spracúvajú prostredníctvom komponentov TextField a Checkbox.

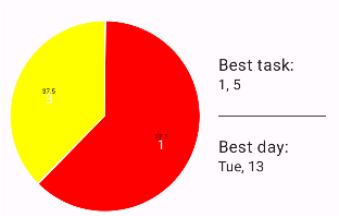
Používatelia môžu upravovať názov úlohy, nastavovať prioritu alebo meniť nastavenia upozornení. Dokončenie úlohy spustí viewModel.completeTask(taskId).

Táto obrazovka taktiež zobrazuje priebeh úloh s nastavenými cieľmi pomocou indikátora LinearProgressIndicator.

Ak má úloha cieľ, skontroluje sa, či postup spĺňa alebo prekračuje cieľ. Ak áno, zobrazí sa dialógové okno o dokončení cieľa.



StatisticScreen.kt agreguje a zobrazuje štatistiky používateľa. Sleduje LiveData zo StatisticViewModel na získanie týždenných a mesačných počtov dokončenia a času stráveného na úlohách. Štatistiky sa vizualizujú pomocou kompozitných komponentov StatisticChart a PieChart. Obrazovka obsahuje rozpis navykonávanejšej úlohy a najproduktívnejšieho dňa, prezentované v Column.



Obrazovka obsahuje aj navigáciu na návrat do hlavného zobrazenia úloh alebo iných častí aplikácie. Navigáciu v aplikácii spravuje súbor NavGraph.kt, ktorý nastavuje navigačné trasy medzi rôznymi obrazovkami pomocou NavController. Tento súbor inicializuje navigáciu pomocou komponentu Jetpack Navigation. Napríklad:

```
NavHost(navController = navController, startDestination = „home“) {
    composable(„home“) { HomeScreen(navController = navController) }
    composable(„taskDetail/{taskId}“) { backStackEntry ->
        val taskId = backStackEntry.arguments?.getLong(„taskId“) ?: 0L
        TaskDetailScreen(taskId = taskId, viewModel = taskViewModel)
    }
}
```

Táto štruktúra zabezpečuje správny prenos údajov medzi obrazovkami, pričom každá obrazovka spracováva svoju špecifickú logiku a komponenty používateľského rozhrania.

Integrácia LiveData a ViewModel v týchto obrazovkách napomáha tomu, že aktualizácie používateľského rozhrania sú reaktívne a zohľadňujú životný cyklus, vďaka čomu je aplikácia efektívna a ľahko sa udržiava.

HomeScreen dáva používateľovi prehľad o úloha a možnosť navigovať na detajlnejší rozpis úlohy. TaskDetailScreen ponúka komplexnú správu úloh a StatisticScreen poskytuje prehľadnú vizualizáciu údajov. Keďže prístup je modulárny, každá obrazovka efektívne zvláda svoje špecifické povinnosti, čo uľahčuje plynulé a intuitívne používanie aplikácie.

3.2. AndroidX komponenty

Implementácia aplikácie vo veľkej miere využíva komponenty AndroidX na správu logiky zohľadňujúcej životný cyklus, uľahčenie navigácie a spracovanie úloh na pozadí.

ViewModel je použitý pre spracovanie údajov súvisiacich s používateľským rozhraním. V HomeScreen.kt, TaskDetailScreen.kt a StatisticScreen.kt sa inštancie ViewModel, ako napríklad TaskViewModel a StatisticViewModel, načítavajú pomocou funkcie hiltViewModel().

Tieto ViewModely poskytujú údaje používateľskému rozhraniu prostredníctvom LiveData, ktoré sa automaticky aktualizuje pri zmene údajov, čím sa zabezpečí, že používateľské rozhranie zostane synchronizované s najnovším stavom premenných a atribútov. Napríklad TaskViewModel používa LiveData na uchovávanie zoznamu úloh:

```
val tasks by viewModel.allTasks.observeAsState(emptyList())
```

Rozširujúca funkcia observeAsState konvertuje LiveData na objekt State, ktorému Jetpack Compose rozumie a ktorý môže použiť na rekompozíciu.

Na injekcie závislostí v aplikácii sa používa Hilt, čím sa zjednodušuje proces poskytovania závislostí. V súbore AppModule.kt Hilt definuje spôsob poskytovania inštancií AppDatabase, TaskDao, StatisticDao a WorkManager. To umožňuje injektovanie týchto závislostí tam, kde je to potrebné, bez potreby manuálnej inštancie.

Napríklad TaskViewModel injektuje TaskDao pomocou Hilt:

```
@HiltViewModel
class TaskViewModel @Inject constructor(application: Application, private val taskDao: TaskDao) : AndroidViewModel(application) {
    // Logika ViewModelu
}
```

V súbore RandomApplication.kt sa používa anotácia @HiltAndroidApp na spustenie generovania kódu Hilt. Tým sa nastaví základ aplikácie pre Hilt.

```
@HiltAndroidApp
class RandomApplication : Application()
```

Táto konfigurácia pomáha mať plynulú a efektívnu správu závislostí v celej aplikácii.

Aplikácia používa na správu databázy Room. TaskEntity a StatisticEntity definované v súboroch TaskEntity.kt a StatisticEntity.kt predstavujú databázové tabuľky.

DAO ako TaskDao a StatisticDao poskytujú metódy na interakciu s databázou, napríklad insertTask(task: TaskEntity), getAllTasks() a getStatisticsByTaskId(taskId: Long). AppDatabase.kt inicializuje databázu Room:

```
@Database(entities = [TaskEntity::class, StatisticEntity::class], version = 3)
abstract class AppDatabase : RoomDatabase() {
    abstract fun taskDao(): TaskDao
    abstract fun statisticDao(): StatisticDao }
```

Ako už bolo predtým spomenuté, navigáciu medzi jednotlivými obrazovkami riadi komponent Navigation. Súbor NavGraph.kt nastavuje NavHost s trasami pre obrazovky ako „home“, „statistics“ a „taskDetail“. Každé miesto v navigačnom grafe zodpovedá obrazovke.

WorkManager spravuje úlohy na pozadí, napríklad notifikácie. V súbore NotificationWorker.kt je definovaný CoroutineWorker, ktorý spracováva notifikácie o termínoch úloh. Tento „worker“ je vytvorený v TaskViewModel.kt pomocou OneTimeWorkRequestBuilder<NotificationWorker>(). Notifikácie sú naplánované na jeden týždeň a jeden deň pred dátumom splatnosti úlohy.

3.3. Notifikácie

Aplikácia používa NotificationManager na zobrazovanie pripomienok termínov úloh. Notifikácie spracováva NotificationWorker.kt, ktorý rozširuje CoroutineWorker.

Keď sa blíži termín splnenia úlohy, spustí oznámenie pomocou metódy showNotification. Táto metóda skonštruuje notifikáciu pomocou NotificationCompat.Builder a potom použije NotificationManager na jej zobrazenie.

V modeli TaskViewModel.kt sa vytvoria oznámenia prostredníctvom NotificationWorker. OneTimeWorkRequestBuilder sa používa s počiatočným oneskorením na základe termínu úlohy.

3.4. Externé frameworky a knižnice

V aplikácii sa používa MPAndroidChart na vizualizáciu údajov vo forme stĺpcových a koláčových grafov. Komponent StatisticChart využíva komponent BarChart na prezentáciu týždenných a mesačných štatistík. Podobne sa vo vlastnom kompozitnom komponente PieChart používa na zobrazenie percentuálneho podielu splnených úloh.

3.5. Organizácia aplikácie

Aplikácia je rozdelená do niekoľkých balíkov, z ktorých každý sa zaoberá špecifickým aspektom funkčnosti aplikácie. Toto rozdelenie pomáha zachovať oddelenie medzi logikou (biznis vrstva) a používateľským rozhraním (prezentačná vrstva).

Balík „database“ obsahuje všetky triedy súvisiace s Room databázou. Patria sem entity (TaskEntity a StatisticEntity), ktoré definujú štruktúru databázových tabuliek, DAO (TaskDao a StatisticDao), ktoré poskytujú metódy na prístup k údajom a manipuláciu s nimi, a trieda AppDatabase, ktorá inicializuje inštanciu databázy Room.

V balíku „ui“ sa nachádzajú komponenty používateľského rozhrania. Podbalík composables obsahuje opakovane použiteľné komponenty používateľského rozhrania, napríklad PieChart.kt, StatisticChart.kt a TaskCreationDialog.kt.

V podbalíku „screens“ sú hlavné obrazovky (HomeScreen.kt, TaskDetailScreen.kt, StatisticScreen.kt). Každá obrazovka načítava údaje prostredníctvom modelov ViewModel, aktualizuje používateľské rozhranie a spracúva interakcie používateľa.

V podbalíku „viewmodels“ definujeme ViewModels, ako napríklad TaskViewModel, TaskDetailViewModel a StatisticViewModel. Tieto ViewModely spracúvajú biznis logiku a manipuláciu s údajmi. Interagujú s DAO pre načítanie, vkladanie, aktualizáciu a mazanie údajov, pričom dávajú LiveData komponentom pre aktualizáciu používateľského rozhrania.

Balík „util“ obsahuje pomocnú triedy NotificationWorker.kt, ktorá je zodpovedná za zobrazovanie oznámení pomocou WorkManageru.

Balík „di“ (dependency injection) obsahuje definície modulov Hilt v súbore AppModule.kt. Definuje, ako poskytovať inštancie rôznych tried, ako sú DAO, databáza Room a WorkManager, vďaka čomu ich možno ľahko injektovať v celej aplikácii.

Koreň projektu obsahuje hlavný vstupný bod aplikácie (MainActivity.kt) a triedu aplikácie (RandomApplication.kt). MainActivity nastavuje počiatočné používateľské rozhranie volaním kompozitnej funkcie NavGraph(). Trieda RandomApplication je anotovaná pomocou @HiltAndroidApp na spustenie generovania kódu Hilt a nastavenie injekcie závislostí.

4. Zoznam použitých zdrojov

Počas vytvárania aplikácie som popri dokumentácii Android Developer často vyhľadávala možnosti, aké komponenty, metódy alebo techniky použiť pre rôzne úlohy, alebo ako vyriešiť výnimk. Napríklad komponent „HorizontalDivider“ som našla pomocou linku:

https://stackoverflow.com/questions/58898299/draw-a-line-in-jetpack-compose

„stackoverflow“ bola veľmi nápomocná stránka pre nájdenie nápadov a riešení.