

Vývoj aplikácii pre mobilné zariadenia

Semestrálna práca

Mobilná aplikácia - ProgressPeak

Meno a priezvisko: Samuel Rojko

Skupina: 5ZYS34

Vedúci semestrálnej práce: doc. Ing. Patrik Hrkút, PhD.

**1.Úvod**

V súčasnej dobe je u ľudí veľký problém, že veľkú časť svojho dňa strávia v práci, kde väčšina sedí u počítačov nepretržite aj niekoľko hodín. Takáto sedavá práca môže mať negatívne dôsledky na naše fyzické aj duševné zdravie. Je preto dôležité, aby sme si našli aj čas na fyzické aktivity v priebehu dňa. Avšak, mnohí ľudia sa stretávajú s problémom motivácie a disciplíny, keď prichádza na vykonávanie fyzických aktivít. Často krát sa totižto cítia unavení, vyčerpaní alebo jednoducho nevedia, ako začať. V takýchto situáciách je často kľúčové mať k dispozícii nástroje, ktoré nás podporia a motivujú k udržiavaniu zdravého a vyváženého životného štýlu. Jedným z takýchto nástrojov by mohla byť práve mobilná aplikácia, ktorá by umožňovala používateľovi si jednoducho nastaviť zvyk, ktorý by chcel dodržiavať, alebo zlozvyk, ktorého by sa chcel zbaviť.

**2.Definícia cieľov a zámer navrhovanej aplikácie**

Ako už bolo okrajovo spomenuté v úvode, aplikácia by predovšetkým mala slúžiť ako motivátor k tomu, aby ľudia vykonávali aj aktivity, ktoré sú prospešné pre ich fyzické a duševné zdravie. Cieľom je tak povzbudiť používateľov k pravidelnejšiemu pohybu a aktívnemu životnému štýlu. Tento cieľ by sa docielil navrhnutím aplikácie, ktorá by používateľovi umožňovala sledovať jeho progres, čo by mu poskytlo prehľad o tom, ako sa mu darí v plnení jeho cieľov. Jeho progres by dokázal sledovať buď vo forme kalendára alebo diagramu. Prehľad vo forme kalendára by mu umožnil sledovať jeho úspechy v časovom kontexte, či už za posledný deň, týždeň, mesiac alebo rok. Na druhej strane pri diagrame by dokázal z neho vyčítať aj trendy. Tieto vizuálne reprezentácie by mu poskytli jasné a zrozumiteľné informácie o tom, ako sa jeho úsilie a disciplína premietajú do jeho výsledkov. Okrem samotných štatistík by ho aplikácia niečim mala odmeňovať, za to že splnil jeho ciele. Odmena by bola vo forme virtuálnej meny, za ktorú by si mohol rozširovať jeho virtuálny inventár. Samozrejme taktiež nesmie chýbať forma upozornení, ktoré by používateľa vopred upozornili na jeho ešte nesplňenú akitivu, v prípade, že by na ňu v priebehu dňa zabudol. Navrhnutie takejto aplikácie, ktorá by ponúkala tieto funkcionality, by umožnilo používateľom získať nielen lepší prehľad o ich splnených cieľoch, ale zároveň by im to aj poskytlo cennú podporu a motiváciu pri dosahovaní zdravého a aktívneho životného štýlu.

**2.Prehľad dostupných aplikácií podobného zamerania**

Ešte pred zadefinovaním presných funkcionalít navrhovanej aplikácie je dobré sa inšpirovať existujúcimi riešeniami na trhu. Preto som sa rozhodol vykonať prehľad konkrétne dvoch dostupných aplikácií s podobným zameraním. Tento prehľad nám poskytne užitočný pohľad na to, aké funkcie poskytujú jednotlivé aplikácie a tiež si tak dokážeme analyzovať ich výhody a nevýhody.

**3.1Aplikácia - Habit Tracker**

Habit Tracker je aplikácia umožňujúca používateľovi sledovať a dodržiavať svoje aktivity. Na hlavnej obrazovke si používateľ dokáže pridať aktivity. Taktiež mu táto obrazovka poskytuje prehľad o splnení úloh vo forme mallých kruhov, ktoré sú umiestnené hneď vrchným kalendárom. Po tom čo si používateľ na danom kalendári zvolí deň, zobrazia sa všetky aktivity s ich cieľmi a aktuálnym progresom. Používateľ tiež môže upravovať progres aktivity, dokáže ju preskočiť alebo aj skryť z obrazovky.

Obrázok, na ktorom je text, snímka obrazovky, softvér, počítačová ikona

Automaticky generovaný popisObrázok, na ktorom je text, snímka obrazovky, operačný systém, softvér

Automaticky generovaný popis

Ďalšia obrazovka je obrazovka štatistiky. Táto obrazovka umožňuje používateľovi sledovať progres svojej aktivity. Najprv vyberie aktivitu a následne sa mu zobrazí kalendár, kde si môže prezerať svoje dosiahnuté výsledky za dané dni. Ďalej je na tejto obrazovke zobrazený aj ročný status a tabuľka s hlavnými informáciami ako napríklad najlepší streak, aktuálny streak a podobne. Aplikácia poskytuje aj grafy pre lepšie vyhodnotenie trendov. Používateľ môže tiež meniť, archivovať alebo vymazať aktivity priamo z tejto obrazovky.

Obrázok, na ktorom je text, snímka obrazovky, diagram, softvér

Automaticky generovaný popisObrázok, na ktorom je text, snímka obrazovky, softvér, počítačová ikona

Automaticky generovaný popisObrázok, na ktorom je text, snímka obrazovky, číslo, softvér

Automaticky generovaný popis

Obrázok 5: Obrazovka pri zobrazovaní štatistík daných aktivít

Obrázok 4: Obrazovka pri zobrazovaní štatistík daných aktivít

Obrázok 3: Obrazovka pri zobrazovaní štatistík daných aktivít

Pridanie aktivity v aplikácii je jednoduché - používateľ stlačí tlačidlo + na hlavnej obrazovke. Následne používateľ môže upraviť detaily ako je meno, popis, ikonu, farbu, cieľ, periódu, typ diagramu, umožňujúc tak detailnú konfiguráciu pre každú aktivitu.

Obrázok, na ktorom je text, snímka obrazovky, písmo, číslo

Automaticky generovaný popisObrázok, na ktorom je text, snímka obrazovky, písmo, číslo

Automaticky generovaný popisObrázok, na ktorom je text, snímka obrazovky, softvér, počítačová ikona

Automaticky generovaný popis

Obrázok 11: Obrazovka pre možnosť modifikácie aktivít

Obrázok 10: Obrazovka pre možnosť modifikácie aktivít

Obrázok 9: Obrazovka pri vytváraní návykov

**3.3 Aplikácia - Habitica**

Na rozdiel od aplikácie Habit Tracker, Habitica tiež odmeňuje používateľov virtuálnou menou za splnenie aktivít. Na hlavnej obrazovke sú zobrazené aktivity, level a kredity používateľa. Nie je tu kalendár, iba zoznam aktivít. Splnenie aktivít zvyšuje level a kredity používateľa.

Obrázok, na ktorom je text, snímka obrazovky, softvér, multimédiá

Automaticky generovaný popisObrázok, na ktorom je text, snímka obrazovky, softvér, multimédiá

Automaticky generovaný popisObrázok, na ktorom je text, snímka obrazovky, softvér, multimédiá

Automaticky generovaný popisĎalšia obrazovka je obrazovka Dailies. Táto obrazovka zobrazuje denné aktivity, ktoré používateľ môže vykonať. Každú aktivitu je možné splniť len raz denne. Ponúka aj začiatočnícke ciele pre nových používateľov, čo ich tak dokáže viacej motivovať.

Obrázok 19: Obrazovka s možnosťou kúpy veci

Obrázok 18: Obrazovka zobrazujúca odmeny

Obrázok 14: Hlavná obrazovka aplikácie Habitica pre vytváranie aktivít

Posledné okno, ktoré spomeniem, je okno ktoré sa používateľovi zobrazí v prípade ak si chce nejakú aktivitu vytvoriť. Tu má používateľ možnosť pomenovať aktivitu, označiť ju za pozitívnu alebo negatívnu, pridať periódu, kedy sa aktivita zresetuje, možnosť vybratia obtiažnosti, a voľba označiť aktivitu pomocou značiek.

Obrázok, na ktorom je text, snímka obrazovky, softvér, multimédiá

Automaticky generovaný popisObrázok, na ktorom je text, snímka obrazovky, softvér

Automaticky generovaný popis

Obrázok 25: Obrazovka zobrazujúca možnosť vytvorenia novej aktivity

Obrázok 24: Obrazovka zobrazujúca možnosť vytvorenia novej aktivity

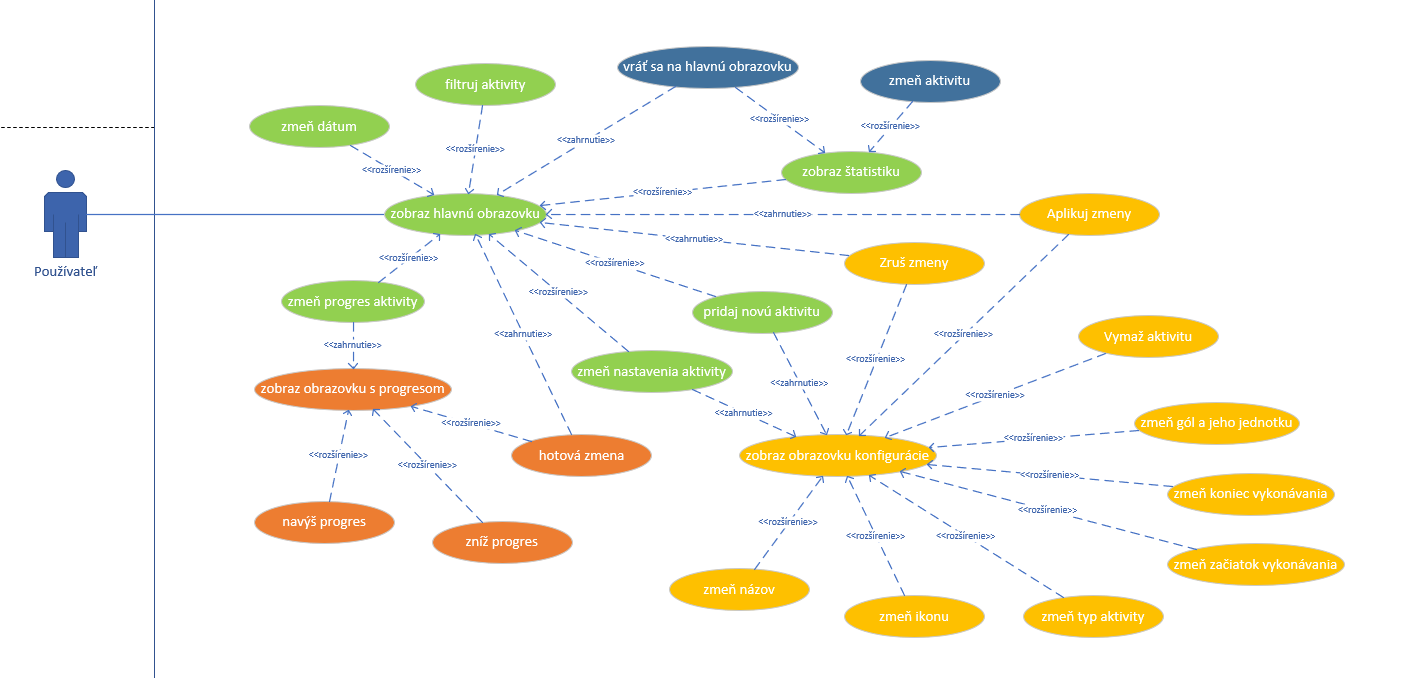
**4 Návrh riešenia**

Po tom ako som si spravil prehľad dostupných aplikácii a v skratke som ich poskytované funkcionality zhodnotil, je správny čas na to tiež definovať používateľské požiadavky mojej aplikácie. Tieto požiadavky by som v skratke zapísal a následne z nich navrhol diagram prípadov použitia, ktorý bude poskytovať prehľad dostupných funkcionalít navrhovanej aplikácie.

**Požiadavky mojej navrhovanej aplikácie z hľadiska funkcionalít:**

* Aplikácia bude poskytovať možnosť si vytvoriť zvyk alebo zlozvyk, kde používateľ dokáže špecifikovať jej meno, ikonu, typ zvyku/zlozvyku, začiatok a koniec vykonávania, stanovenie cieľa ako aj jeho jednotku (km,h,...).
* Na hlavnej obrazovke bude mať používateľ možnosť si prehľadávať jeho zvyky/zlozvyky, pričom mu bude aplikácia poskytovať aj náhľad o progrese danej aktivity za vybraný deň. Okrem toho si dokáže filtrovať tieto zobrazené aktivity. Filtrovať si ich dokáže podľa mena alebo podľa veľkosti progresu a taktiež si tieto aktivity vie vzostupne alebo zostupne zoradiť.
* Nakoniec si používateľ dokáže zobraziť aj jednoduchú štatistiku o vykonaných aktivitách. Táto štatistika bude zahŕňať priemerný progres denne, maximálny progres za deň, počet aktívnych dní a aj celkový progres danej aktivity.

**Nižšie je uvedený Use Case diagram:**

****

**5 Použité technológie**

Aplikácia sa skladá zo štyroch obrazoviek. Všetky tieto obrazovky správne reagujú aj na otočenie displeja, ako aj na zmenu režimov (Light mode/Dark mode). Všetky zdroje, ktoré aplikácia využíva, ako napr. farby a texty, sú uvedené v zložke resources, čo urobia našu aplikáciu flexibilnejšou, napr. pri zmene jazyka aplikácie a podobne.

* **Architektúra aplikácie**

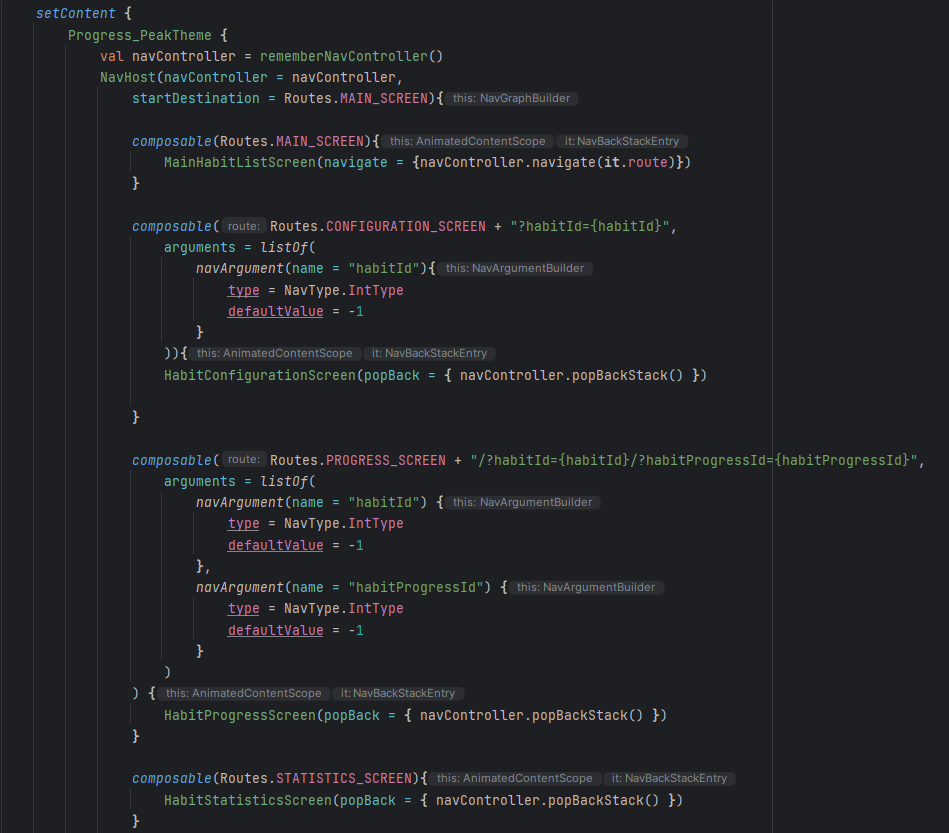
Aplikácia je taktiež postavená na architektúre VVMM (View ViewModel Model), čo nám umožňuje separovať komunikáciu medzi databázou a používateľským rozhraním. Takáto architektúra teda programátorovi poskytuje takzvaný ViewModel, ktorý bude celú túto komunikáciu riadiť.

* **Využité komponenty aplikácie**

Z hľadiska komponentov moja aplikácia využíva následovné komponenty:

***Navigation***, ***Room***, ***ViewModel***.

***Navigation*** nám umožňuje sa jednoducho prepínať medzi jednotlivými obrazovkami. Z hľadiska Navigácie je najskôr potrebné zadefinovať si takzvaný *NavHost*, ktorý bude vedieť o všetkých obrazovkách, ako aj o ich argumenty. *NavHost* pri inicializácii taktiež potrebuje *NavController* a začiatočnú cestu k hlavnej obrazovke. Následne v tele *NavHost* sú zadefinované jednotlivé obrazovky spolu s ich argumentmi.



Ďalším komponentom v mojej aplikácii je komponent ***Room***.

***Room*** je jednoduchá databáza od google, ktorá nám umožní relatívne jednoducho si uchovávať dáta o aplikácii aj po jej ukončení.

Pre správne vytvorenie Room databázy je potrebné zadefinovať všetky tabuľky, ktoré naša databáza bude používať. Následne je tiež dôležité špecifikovať všetky funkcionality, ktoré sa nad databázou budú vykonávať, a to vo forme SQL dotazou. Trieda, ktorá obsahuje všetky tieto funkcionality je známa aj pod názvom Repository. Trieda Repository sa následne ako argument pošle do všetkých ViewModelov, ktoré potrebujú s danou databázou komunikovať.



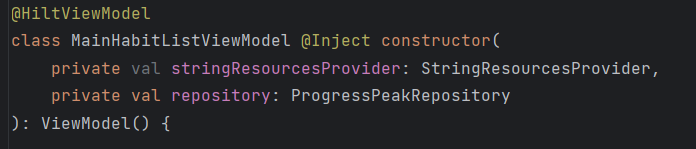
Posledným komponentom, ktorý som vo svojej aplikácii využil je komponent s názvom ***ViewModel***. Ako som už vyššie písal, tento komponent slúži ako také rozhranie medzi používateľským rozhraním a databázou našej aplikácie. Pri mojej implementácii taktiež uchováva veľkú časť stavou, čo zabezbečí aktualizáciu dát, ako aj ich správne zachovanie pri rotácii obrazovky. Takiež jeho dôležitou súčasťou je spracovávanie všetkých eventov, ktoré sa vykonali z používateľského rozhrania (a samozrejme tiež dokáže posúvať eventy aj naspäť používateľskému rozhraniu).

* **Externý framework / knižnice**

Z hľadiska externých knižníc som použil nasledovné:

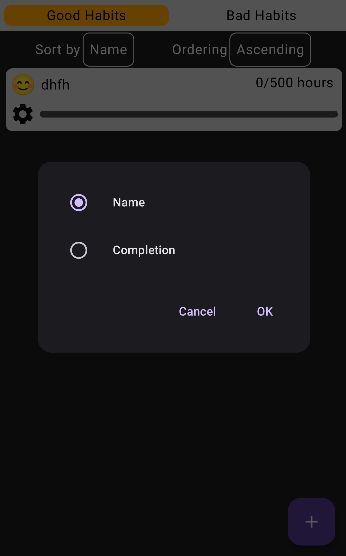
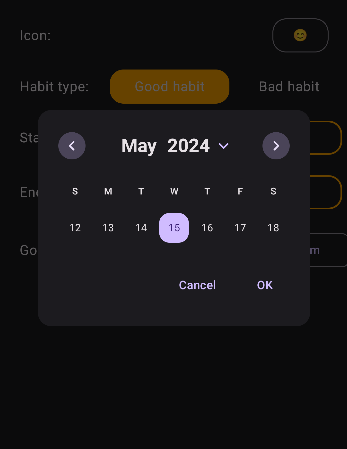
***Dagger Hilt***, ***sheets-compose-dialogs***

Knižnica ***Dagger Hilt*** umožňuje jednoducho vytvárať triedy ako sú napríklad ViewModel, Screens, a podobne. Totižto pomocou neho sa vyhneme zložitému spôsobu inicializovania tried, kde by sme museli kotlinu jasne povedať ako sa majú vytvoriť dané triedy, v prípade ak obsahuje parametre iných objektov. Dagger Hilt nám tak tieto inicializácie konštruktorov dokáže zjednodušiť a urobiť ich omnoho prehľadnejšie.



***Sheets-compose-dialogs***

Knižnica ***Sheets-compose-dialogs*** umožňuje jednoducho vytvárať dialógy, ktoré sú postavené na jetpack compose. V mojom prípade, túto knižnicu používam napríklad pri zobrazovaní dialógového okna pri filtrovaní aktivít, pri zmene ikony pri konfigurácii, a taktiež pri zmene dátumu pre začiatok a koniec vykonávania aktivity.



**5 Štruktúra aplikácie a implementácia**

Aplikácia je kvôli prehľadnosti rozdelená do viacerých balíčkov.

Všetky funkcionality, ktoré prislúchajú databáze sú umiestnené v balíčku **Database**.

**Balíčky:**

* **Balíček Database:** Tento balíček následne obsahuje podbalíčky. Podbalíček Dao obsahuje definované funkcionality vykonávané nad databázou, DatabaseAssociations obsahuje definované vzťahy medzi tabuľkami, DatabaseTables obsahuje všetky tabuľky, a nakoniec RoomDatabase obsahuje samotnú databázu so všetkými definovanými tabuľkami.
* **Balíček Screens:** Tento balíček obsahuje všetky obrazovky, ktoré sú v mojej aplikácii implementované. Všetky tieto balíčky obsahujú rovnaké podbalíčky. Týmito podbalíčkami sú MainComponents, OtherComponents a ScaffoldBars. Podbalíček MainComponents vždy pri každej obrazovke obsahuje Event triedu, pre spracovávanie eventov, ďalej samotnú Screen composable, a nakoniec aj jej ViewModel.
* **Balíček Utils:** Tento balíček obsahuje triedy UiEvent, ktorý definuje eventy, ktoré sa posielajú používateľskému rozhraniu z ViewModel. Taktiež obsahuje triedu StringResourcesProvider, pre jednoduchú možnosť získavania reťazcov z ViewModel. Následne obsahuje ešte triedu Routes, ktorá obsahuje len konštanty, ktoré definujú cesty k jednotlivým obrazovkám vo forme reťazcov. A ako posledné tiež obsahuje tiež triedu DialogListOptionUtils, ktorá sa využíva pre získavanie a aktualizovanie dát v rámci dialógových okien. Čo sa týka podbalíčkov tak obsahuje jeden podpalíček s názvom FunctionExtensions, ktorý obohacuje triedu Number o jednu metódu, pre formátovanie čísel.

**Implementácia:**

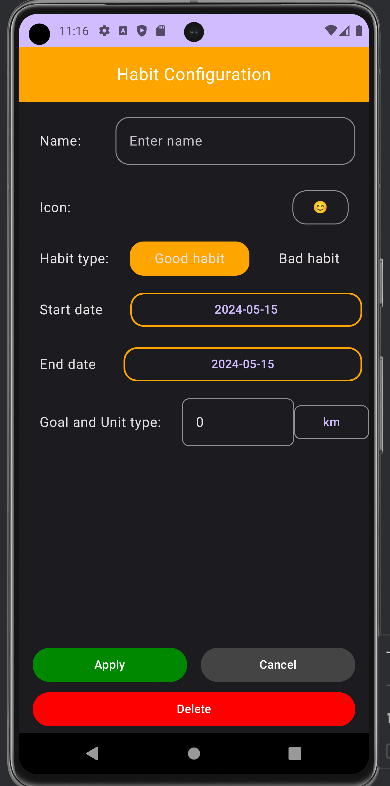
**Obrazovky:**

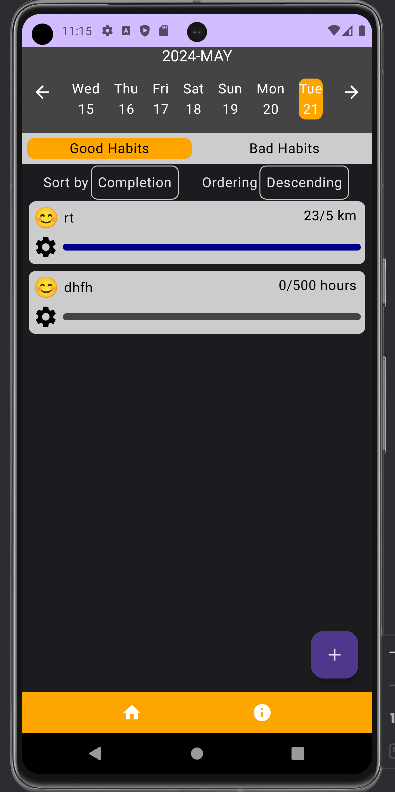
Každá obrazovku samostatné triedy (HabitConfigurationScreen, HabitProgressScreen, HabitStatisticsScreen a MainHabitListScreen). Každá z týchto tried následne využíva ich vlastné ViewModels(HabitConfigurationViewModel,HabitProgressViewModel,HabitStatisticsViewModel a MainHabitListViewModel). Nakoniec každá táto obrazovka taktiež využíva vlastnú triedu pre spracovávanie eventov, ktoré sa posielajú z používateľského rozhrania do ViewModels pre danú obrazovku (HabitConfigurationEvent, HabitProgressEvent, HabitStatisticsEvent a MainHabitListEvent).

**Databáza:**

Z hľadiska databázy som využil lokálnu Room databázu. Z hľadiska implementácie som si vytvoril triedu ProgressPeakDao, ktorá definuje všetky potrebné funkcionality, ktoré neskôr budú využívať ViewModels. Táto trieda je ešte zaobalená triedou ProgressPeakRepositoryImpl, ktorá dedí funkcionality od triedy ProgressPeakRepository. Je to z toho dôvodu, že v budúcnosti by sme možno chceli použiť aj iný typ repozitára (databázy) v našej aplikácii. Následne som si vytvoril triedy Habit a HabitProgression, ktoré reprezentujú tabuľky v databáze. Tieto tabuľky medzi sebou previazané vo vzťahu 1:N, čo je špecifikované v triede HabitWithHabitProgressions. Nakoniec všetky existujúce tabuľky sú zaznamenané v triede ProgressPeakDatabase. Taktiež ešte nemôžeme zabúdať na to, že v týchto tabuľkách sa taktiež uchováva záznam o dátume, ktorý je vo formáte LocalDate. Keďže Room databáza nevie o tom ako pracovať s týmto typom pri vykonávaní SQL dotazou, tak musíme jej špecifikovať ako s nimi má narábať. Preto sme museli tiež vytvoriť samostatnú triedu RoomDataConverters, ktorá špecifikuje napríklad to, ako sa má tento formát prekonvertovať do formátu String, a podobne.

**Výsledok implementovanej aplikácie:**

Hlavná obrazovka: Obrazovka pridávania aktivity:



Obrazovka pre editáciu progresu: Obrazovka štatistík:

