ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIE A INFORMATIKY

ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Vývoj aplikácii pre mobilné zariadenia: Semestrálna práca

Študijný program: Informatika a Riadenie

Vypracoval: Zimen Dávid

Školský rok: 2021/2022
Študijná skupina: 5ZYR22

1 PREDSTAVENIE APLIKÁCIE

Pre semestrálnu prácu som si vymyslel aplikáciu pre FRI-čkárov s názvom *„FRI To-Do“*. Bude to špeciálny To-Do list. Tento To-Do list bude mať už od začiatku definované jednotlivé kategórie, aby si študenti mohli pridávať veci, ktoré musia urobiť pridávať ku jednotlivým predmetom, ktoré majú v škole. Nazvem to pre tento prípad *„povinnosť“*.

Budú si môcť pridať povinnosť na spravenie. Pri danej povinnosti si budú môcť pridať dátum, do kedy je potrebné ju vykonať a krátky popis, aby po určitom čase nezabudli na čo je vlastne daná povinnosť.

Jednotlivé povinnosti si budú môcť prezerať dvoma spôsobmi a to vyhľadávaním v konkrétnych zoznamoch podľa predmetov alebo v kalendári podľa dátumu.

Povinnosti si budú môcť odškrtávať ako splnené, vtedy povinnosť ostane stále v systéme. Alebo za splnenú povinnosť sa bude považovať aj povinnosť, ktorá bola odstránená z akéhokoľvek zoznamu.

2 POROVNANIE S PODOBNÝMI APLIKÁCIAMI

Aplikácii, ktoré sú určené na vytváranie To-Do listov je v Google Play obchode nespočetne veľa a v podstate všetky ponúkajú totožnú alebo pri najmenšom veľmi podobnú funkcionalitu.

Asi jednu z najzaujímavejších vychytávok som našiel v aplikácii *„Any.do – Úlohy + Kalendár“*, ktorá má v sebe mód na sústredenie. Je to veľmi pekná pomôcka, kedy sa na obrazovke zobrazí časovač a užívateľ akože sadí strom a tým získa aj akože dobrý pocit, že robí niečo pre planétu.

Ďalšiu super vychytávku som našiel v aplikácii *„Taskito: To-Do List, Planner“*, kde sa z jednotlivých zoznamov úloh mohol vytvoriť akože projekt, kde boli jednotlivé úlohy zoradená a podľa odškrtnutých úloh si užívateľ môže sledovať svoj progres.

To sú asi najzaujímavejšie vychytávky. Okrem toho každá aplikácia má ako štandard prihlásenie pomocou rôznych účtov ako Facebook, Google, ... Medzi štandard taktiež patria widgety na ploche telefónu a možnosti upozornenia v akomkoľvek čase pred danou úlohou.

3 ARCHITEKTÚRA APLIKÁCIE

Na začiatku je aktivita CheckStateActivity, ktorá skontroluje, či používateľ ostal prihlásený alebo nie, podľa toho rozhodne, či sa pôjde prihlasovať, alebo rovno zobrazí zoznamy.

Do aplikácie sa bude dať prihlásiť alebo registrovať. Na toto je vyhradená aktivita s názvom LoggingActivity, ktorá sa sklaá z dvoch fragmentov. Jeden na prihlásenie a druhý určený na registráciu.

Samotná aplikácia by sa skladá z troch aktivít. MainFuncionalityActivity na prezeranie zoznamov, povinností v zoznamoch a pridávanie povinností. CalendarActivity, kde sa zobrazujú povinností pre jednotlivé dni

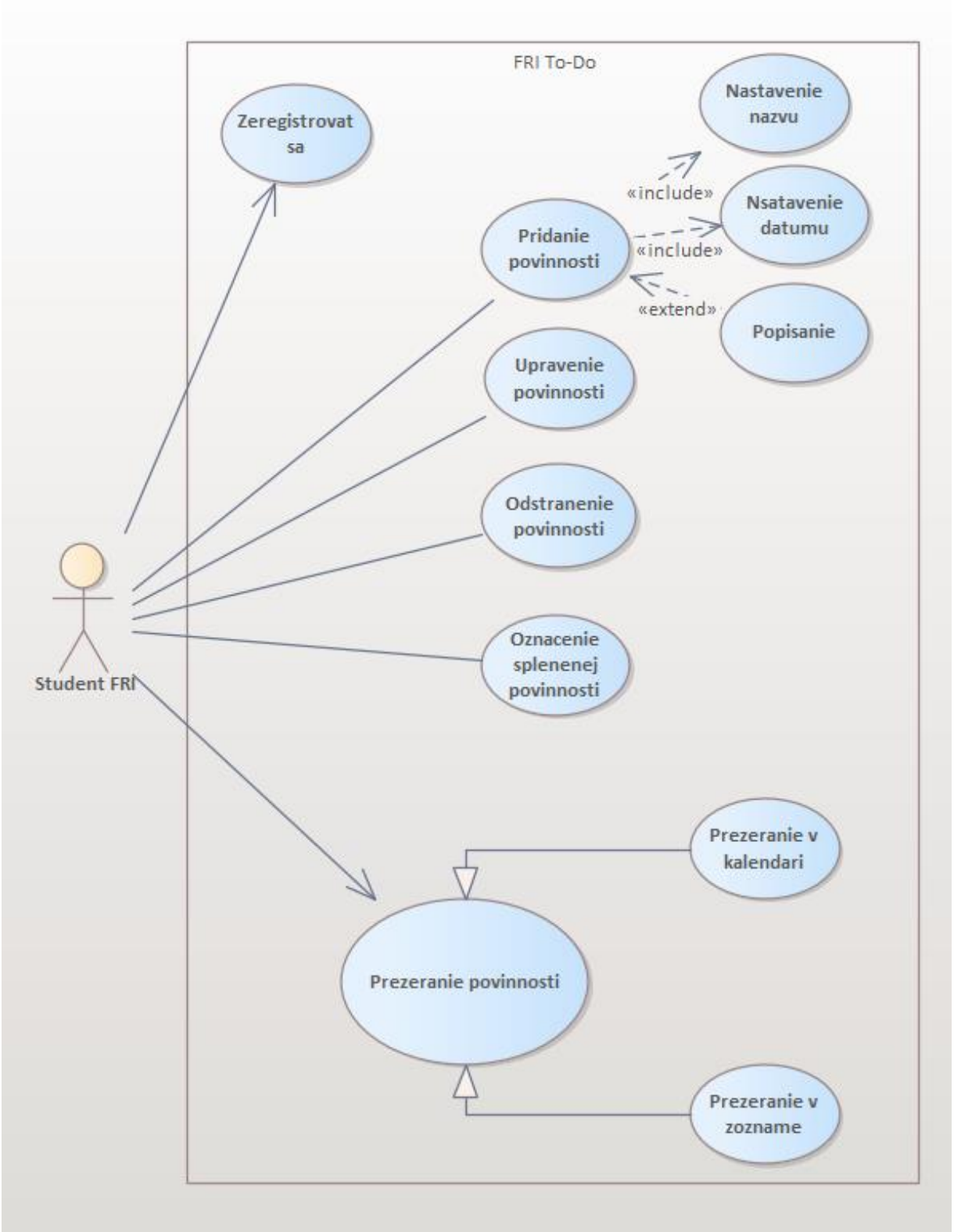
a StudentInformationActivity, kde sa zobrazia informácie o aktuálne prihlásenom študentovi.

Údaje sú potom uchovávané v data class triedach, pričom jedna je pre študenta, jedna pre zoznam a jedna pre povinnosť. Sú v nich uvedené potrebné informácie pre správny chod aplikácie.

Následne sa z týchto tried vytvoria entity v Room databáze a tieto informácie sa využívajú na korektný beh aplikácie.

4 USE CASE DIAGRAM

Na nasledujúcom obrázku je znázornený use-case diagram, kde je graficky zobrazené, čo všetko môže študent s aplikáciou vykonávať.



Obrázok 1 - Use case diagram pre aplikáciu

5 IMPLEMENTÁCIA

V nasledujúcom odseku popíšem jednotlivé prvky použité pri programovaní aplikácie.

5.1 AKTIVITY

Jednoduché popísanie aktivít a fragmentov v rámci jednotlivých aktivít.

5.1.1 StateCheckActivity

Jednoduchá aktivita, ktorá zobrazí na začiatku prázdnu obrazovku, keď vyhodnocuje, či používateľ ostal prihlásený alebo nie. Na vyhodnotenie používa informáciu zo SharedPreferences, ktorá je súčasťou Androidom ponúkaných tried a podľa toho vyhodnotí ďalší postup.

Aktivita v sebe neobsahuje žiadne fragmenty.

5.1.2 LoggingActivity

Aktivita, ktorá vyhodnocuje prihlásenie alebo registráciu do aplikácie, ak už používateľ neostal prihlásený.

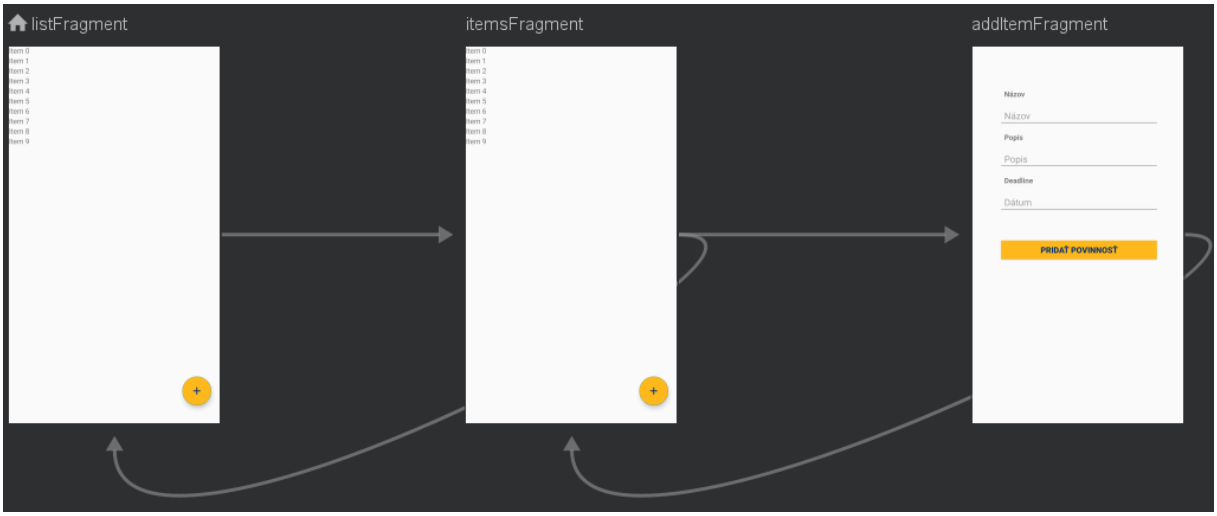
Obrázok 2 - Navigačný graf pre LoggingActivity

Aktivita má v sebe nasledovný navigačný graf, ktorý začínam fragmentom, kde si používateľ môže vybrať, či sa chce prihlásiť alebo zaregistrovať.

Fragmenty následne na základe zadaných údajov vyhodnotia požiadavku od používateľa a presunú sa do MainFuncionalityActivity, kde cez intent pošlú inštanciu študenta.

5.1.3 MainFuncionalityActivity

Preberie cez intent inštanciu študenta od jednej z predošlých aktivít, aby vedela korektne pracovať s údajmi.



Obrázok 3 - Navigačný graf pre MainFuncionalityActivity

Aktivita má v sebe takýto navigačný graf.

ListFragment preberie údaje o používateľovi a zobrazí jeho zoznamy s povinnosťami pomocou RecyclerView, kde má zoznam pri sebe aj ikonku.

Na tomto fragmente je aj blokované tlačidlo späť, aby nebolo možné sa dostať naspäť do LoggingActivity.

Po kliknutí na tlačidlo + sa zobrazí dialógové okno, do ktorého je možné zadať názov nového zoznamu.

Pre odstránenie zoznamu je možné pomocou potiahnutia z pravej strany obrazovky, pričom je potrebné potvrdiť výstražnú správu.

Po kliknutí na konkrétny zoznam sa aktivita presunie na ItemsFragment, kde sa taktiež pomocou RecyclerView zobrazia povinnosti vo vybranom zozname.

Po kliknutí na prvok sa zobrazí okno, kde sú napísané informácie o povinnosti.

Po kliknutí na tlačidlo + sa aktivita presunie na AddItemFragment, kde sa vyplnia polia a pridá sa nová povinnosť do daného zoznamu.

Po pridaní povinnosti sa následne fragment odstráni z BackStacku, aby nebolo možné sa doň vrátiť tlačidlom späť.

5.1.4 CalendarActivity

Aktivita, ktorá v sebe zobrazí mesačný kalendár a po kliknutí na konkrétny deň zobrazí pod kalendárom v RecyclerView povinnosti, ktoré pripadajú na daný deň.

Aktivita si preberá dáta z MainFuncViewModel.

5.1.5 StudentInformationActivity

Preberie pri vstupe informácie o aktuálne prihlásenom študentovi, ktoré následne zobrazí do fragmentu.

5.2 ANDROIDX KOMPONENTY

Využité AndroidX komponenty v aplikácii.

5.2.1 LiveData

LiveData sú v aplikácii použité na vracanie výsledkov z databázy, kde sa vracia napríklad z transakcie LiveData<StudentWithLists?> a tieto dáta sa následne posielajú do ViewModelov a tým sa zaručujú aktuálne dáta na obrazovke zariadenia.

```
/**
 * Vráti študenta so všetkými svojimi zoznamami, ktoré vytvoril.
 */
@Transaction
@Query( value: "SELECT * FROM users_table WHERE osCislo = :osCislo")
fun getStudentWithLists(osCislo: Int): LiveData<List<StudentWithLists?>>
```

Obrázok 4 - príklad využitia prvku LiveData

5.2.2 Navigation

Navigácia je využitá v aplikácii v aktivitách LoggingActivity a MainFuncionalityActivity (pozri Obrázok 2 a 3). Zaisťuje to, že v aktivitách majú fragmenty dané jasné a logické poradie.

5.2.3 Room databáza

Obsahuje dátové triedy StudentDC, ToDoListDC, ToDoItemDC, ktoré majú v sebe atribúty, pre všetky dôležité vlastnosti. Z tých tried sú následne pre databázu vytvárané entity, do ktorých sú vkladané údaje.

Taktiež obsahuje väzobné dátové triedy StudentWithList, ktorá obsahuje v sebe studenta a list v ktorom sú uložené ToDo listy väzobnú podmienku cez @Relation, ktorou sú spojené 2 entity. Takto podobne sú navrhnuté aj triedy ListWithItems a StudentWithItems. Pre všetky tieto vzťahy platí kardinalita 1:N a nepovinné členstvo.

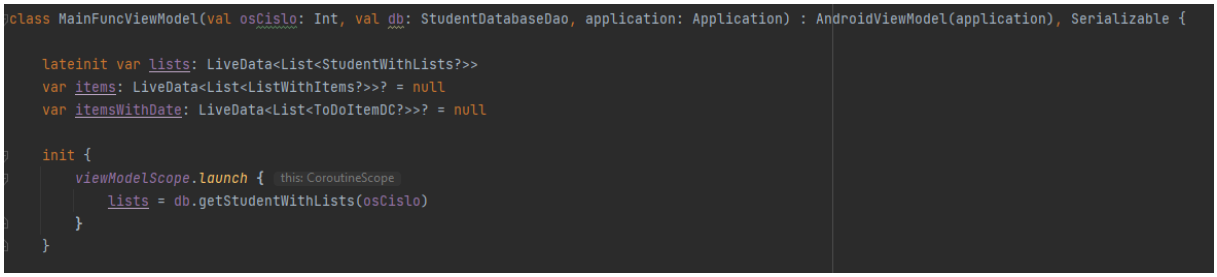
```
data class StudentWithLists (

    @Embedded val student: StudentDC,
    @Relation(
        parentColumn = "osCislo",
        entityColumn = "student"
    )

    val lists: List<ToDoListDC?>
    )
```

Obrázok 5 - príklad väzobnej dátovej triedy

Súčasťou je aj interface StudentDatabaseDao, kde sú implementované všetky queries, ktoré sú potrebné pre zaistenie správneho fungovania aplikácie.

5.2.4 ViewModel

V aplikácii sú naimplementované 2 ViewModel triedy, jedna je MainFuncionalityViewModel, ktorá obsluhuje aktivity MainFuncionalityActivity, CalendarActivity a StudentInformationActivity. Druhá je LoggingViewModel, ktorá obsluhuje LoggingActivity a StateCheckActivity.

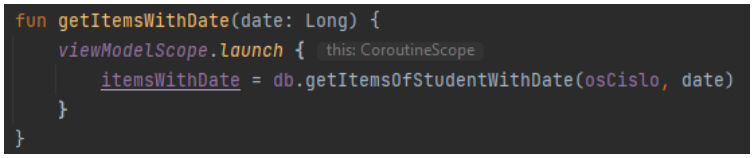


Obrázok 6 - príklad VievModelu

Vo ViewModely sa nachádzajú údaje uložené ako LiveData, aby bola zaručená ich aktualizácia a trvácnosť.

Sú udržiavané pomocou funkcií, ktoré v podstate prepisujú funkcie so StudentDatabaseDao, ale využívajú pri tom viewModelScope, ktorý zaisťuje, že transkacie v databáze sú vykonávané mimo hlavnej vetvy aplikácie.

Tieto metódy následne ukladajú výsledky do atribútov v triede.



Obrázok 7 - príklad funkciew v triede MainFuncViewModel

Ku obom ViewModel je aj samostatne implemetovaná trieda ako ViewModelFactory, ktorá má za úlohu vytvárať inštanciu ViewModelu a ten následne distibuovať.

5.2.5 RecyclerView

RecyclerView je implementovaný na fragmentoch ListsFragment, ItemsFragment a CalendarFragment.

Pre zoznamy a povinnosti sú implementované aj adaptéry s názvom ToDoListDCAdapter a ToDoItemDCAdapter.

Oba majú v sebe atribút dáta, ktorý predstavuje list konkrétnych prvkov, ktoré sa v pohľade majú zobraziť.

Taktiež majú v sebe vnorené triedy, ktoré v sebe držia layout v akom sa majú jednotlivé prvky zobrazovať na obrazovke.

Údaje sa do adaptérov dostávajú z príslušných ViewModelov, ktoré zaručujú ich aktuálnosť.

6 ZDROJE

https://kotlinlang.org/docs/home.html

https://stackoverflow.com/

https://guides.codepath.com/android/fragment-navigation-drawer

https://developer.android.com/teach#for-instructors-teaching-a-course

https://medium.com/androiddevelopers/room-coroutines-422b786dc4c5