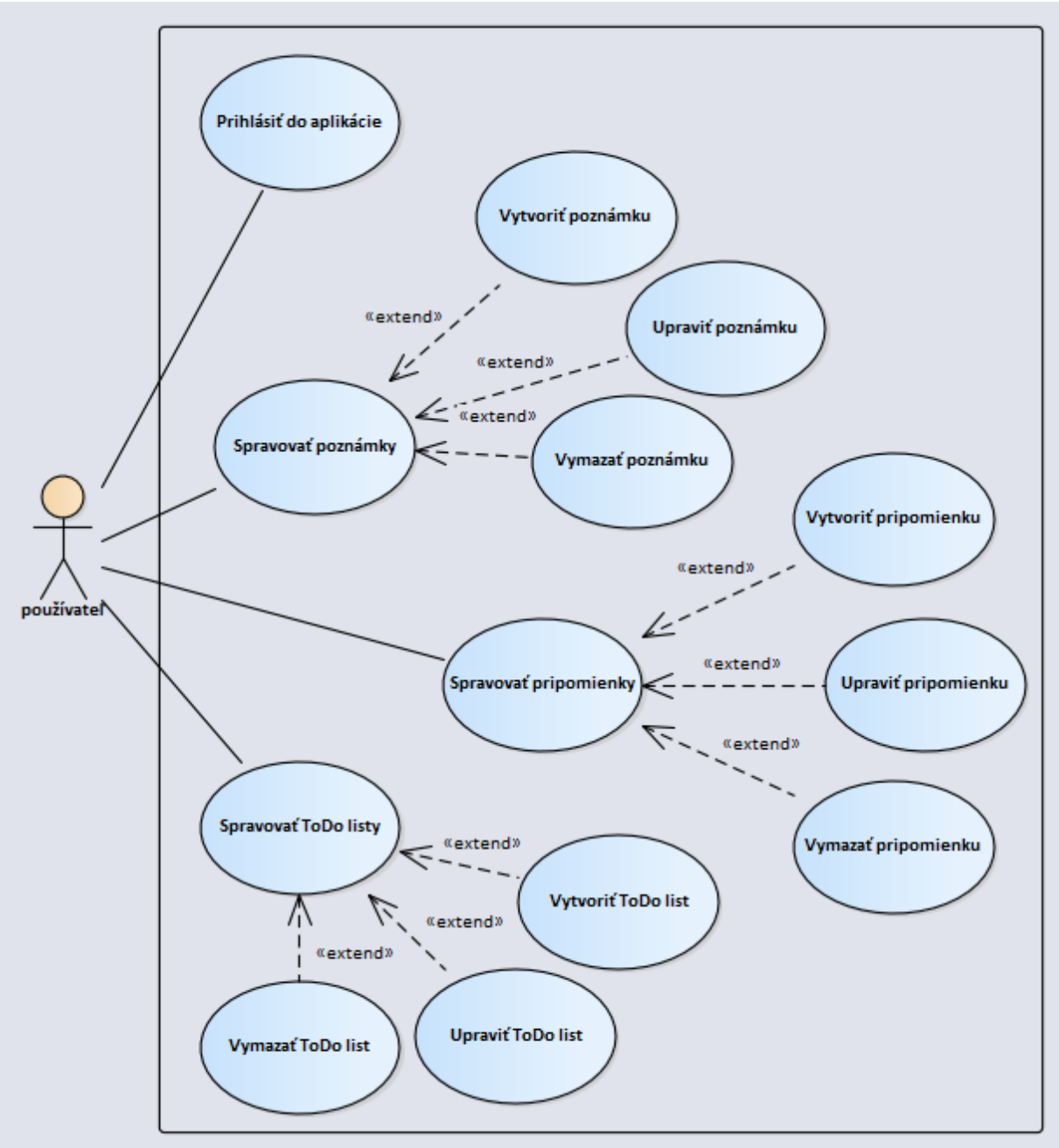
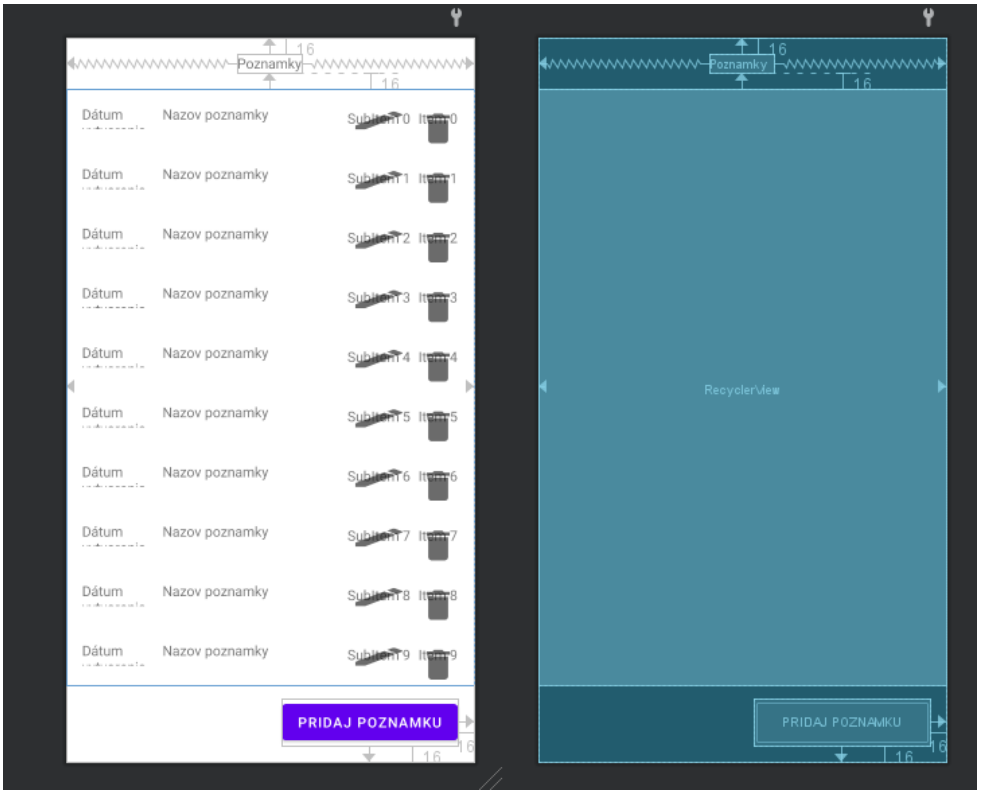
Diar

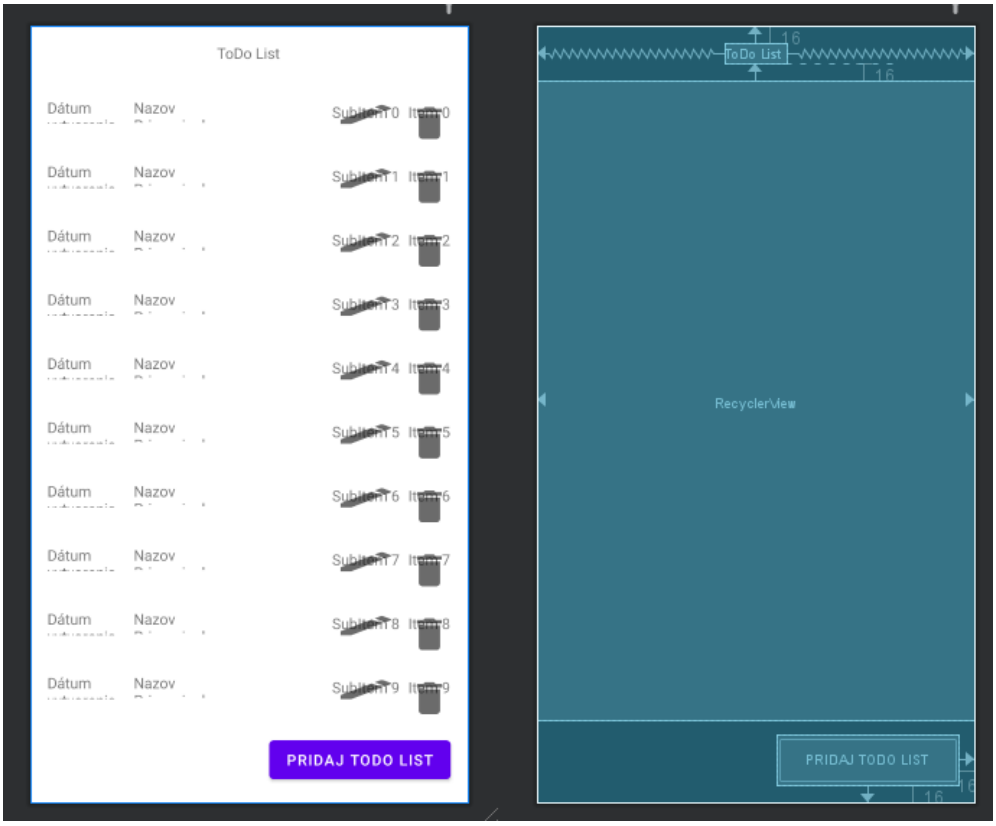
Aplikácia bude pracovať s externou databázou kde sa budú poznámky, pripomienky a ToDo listy ukladať, bude obsahovať aj možnosť prihlásenia sa. Obrazovka prihlásenia bude prvá obrazovka po štarte aplikácie kde sa používateľ bude musieť prihlásiť. Hlavné obrazovky kde budú poznámky, pripomienky a ToDo listy zobrazene budú fragmentami medzi ktorými sa bude prechádzať posúvaním do strán, každý fragment bude obsahovať RecyclerView na zobrazenie prvkov. Prvky budú mať priradené tlačidla na vymazanie a úpravu a kliknutím na prvok sa zobrazí obrazovka detailu. Pripomienky by mali vytvárať notifikácie o prichádzajúcich udalostiach, mala by tam byť aj možnosť viacerých ToDo listov. Triedy by mali byť Poznámka, Pripomienka, ToDoList, a Úloha ktorá bude predstavovať jeden prvok jedného ToDoListu. ToDoList bude obsahovať List jeho úloh, a každý fragment bude obsahovať zoznam prvkov.

Na základe podobných aplikácií by bolo vhodné implementovať možnosť kategorizácie poznámok.

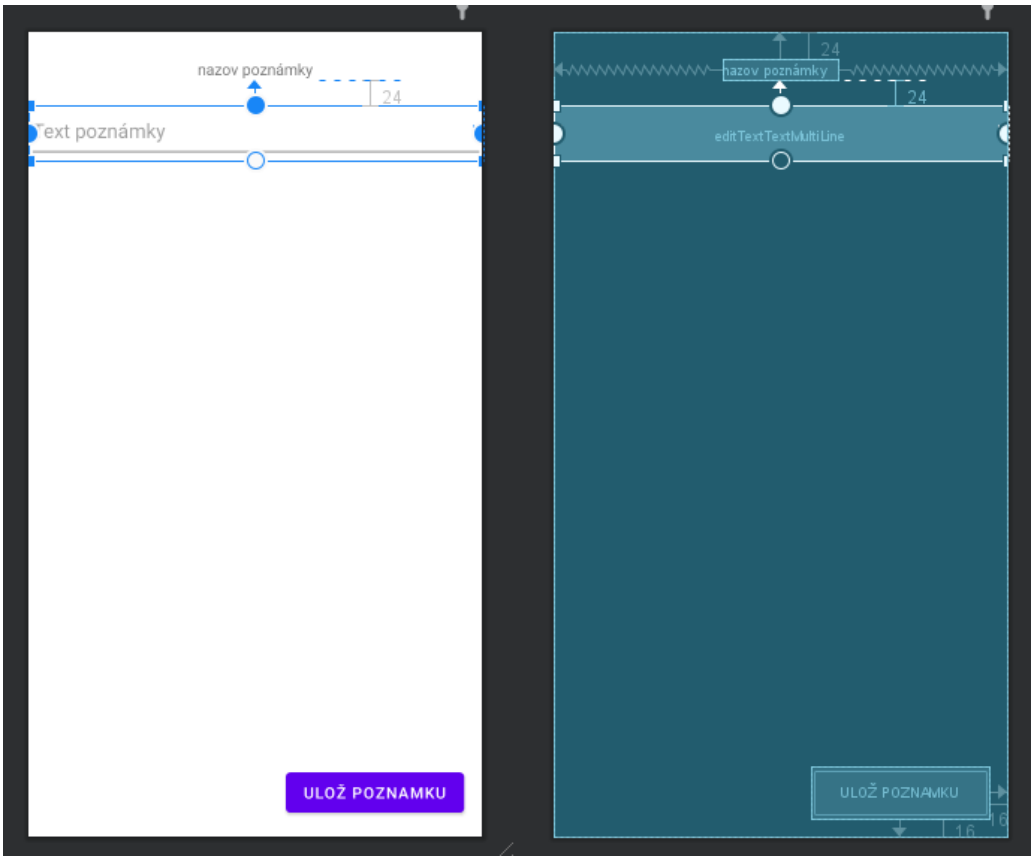




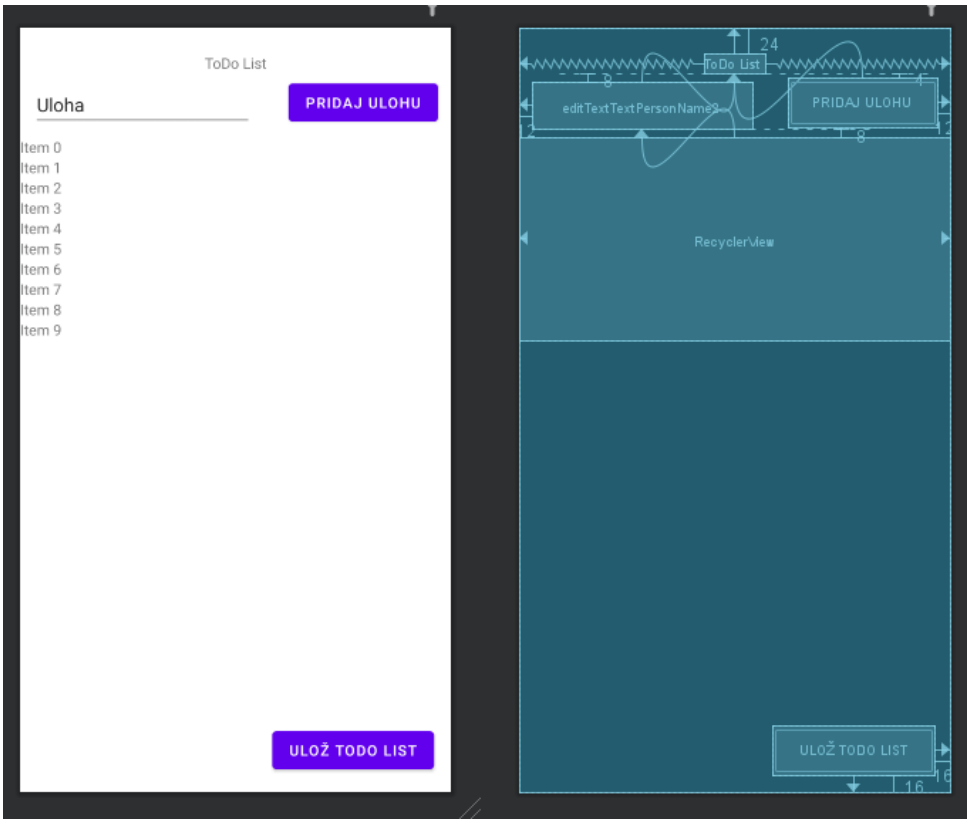
Návrh obrazovky poznámok



Návrh obrazovky ToDo listov



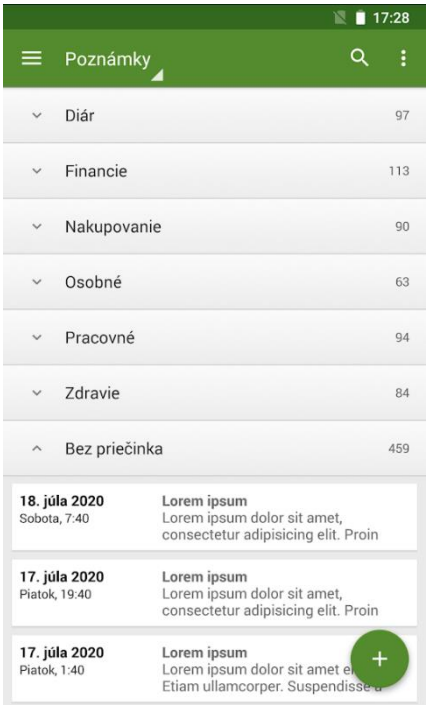
Návrh obrazovky vytvorenia novej poznámky



Návrh obrazovky vytvorenia nového ToDo listu.

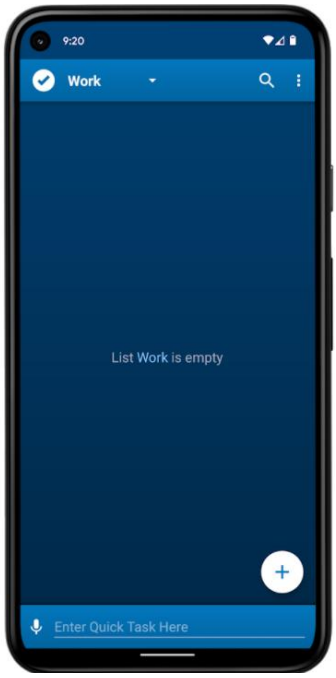
Podobné aplikácie:

Moje Poznámky - Poznámkový Blok



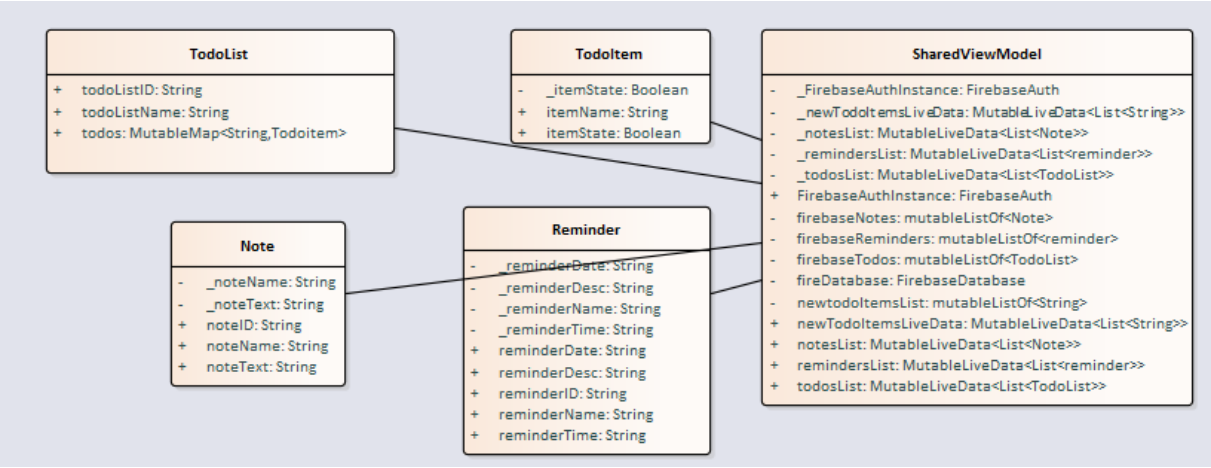
Jednoduchá aplikácia na zapisovanie poznámok avšak pekne prevedená s možnosťou rozdelenia do kategórií, synchronizácia s databázou, vlastné heslo.

Zoznam úloh



Aplikácia zoznam úloh ponúka vytvorenie viacerých samostatných zoznamov úloh

Diagram tried riešenia



Popis implementácie

Aplikácia obsahuje dátové triedy: TodoList, TodoItem, Reminder a Note pre uchovanie dát predstavujúcich objekty aplikácie a SharedViewModel ktorý obsahuje všetky inštancie týchto tried, zabezpečuje ich komunikáciu s databázou, ukladanie a získavanie dát z databázy. Aka databáza sa využíva google služba Firebase ktorá poskytuje jednoduchú možnosť ukladania dát a aj autenfikácie používateľov pre registráciu a prihlásenie.
Dáta načítane z databázy sa uložia do premennych MutableLiveData, observer vo fragmentoch aplikácie tieto dáta načíta do adaptéra ktorý je priradný recyclerViewu.
Pre prácu s recyclerView sa využíva Groupie Adapter ktorý sa stará o zobrazovanie prvkov recyclerviewu a ich aktualizáciu.

Pre navigáciu sa využívajú tlačidlá a navigačný graf.

Ukážka live dát na poznámkach.

```
private var _notesList = MutableLiveData<List<Note>>()
val notesList: MutableLiveData<List<Note>>
    get() = _notesList

private val firebaseNotes= mutableListOf<Note>()
```

```
fireDatabase.getReference("${FirebaseAuthInstance.currentUser!!.uid}/Notes").addChildEventListener(
    object : ChildEventListener {
        /**
         * This method is triggered when a new child is added to the location to which this listener was
         * added.
         *
         * @param snapshot An immutable snapshot of the data at the new child location
         * @param previousChildName The key name of sibling location ordered before the new child. This
         * will be null for the first child node of a location.
         */
        override fun onChildAdded(snapshot: DataSnapshot, previousChildName: String?) {
            val note = snapshot.getValue(Note::class.java)
            firebaseNotes.add(note!!)
            _notesList.postValue(firebaseNotes)
        }
```

Vo fragmente:

```
/**
 * Setup observer for live data of Notes and store it in adapter for recyclerView
 */
private fun loadNotes() {
    viewModel.notesList.observe(viewLifecycleOwner, Observer {  it: List<Note>!
        adapter.update(it.toNoteItem())
    })
}
```

Využitie navigačného grafu:

```
/**
 * creates action to navigate from notesFragment to createNewNoteFragment
 */
private fun createNewNote() {
    val action = NotesFragmentDirections.actionNotesFragmentToNewNoteFragment(note = null)
    findNavController().navigate(action)
}
```