Exercise Tracker
(Cvičebné plány)

Popis aplikácie

Rôznych aplikácií na cvičenie alebo stravovanie je veľmi veľa a napriek ich komplexným možnostiam a špecifickým zameraniam ktoré poskytujú, sú celé v anglickom jazyku a užívateľov obmedzujú v možnosti ich vlastných cvičebných plánov. Okrem toho je v dnešnej dobe výrazným trendom poskytovanie niektorých funkcií len s predplatným v kontraste s jednorazovou platbou alebo úplne bez poplatku. Aktuálne najpopulárnejšími aplikáciami sú „Domáce tréningy“ od vývojára Leap Fitness Group a Fitify. Táto aplikácia si tak dáva za cieľ priniesť väčšiu kontrolu užívateľom vrátane vytvárania ich vlastných sérií, plánov, pripomienok na ich dodržiavanie a dokonca aj pridávania vlastných cvikov.

Aplikácia Domáce Tréningy

je celá vrátane cvičebných textov po anglicky. Návrh cvičebného plánu je síce zadarmo, ale v aplikácií sú reklamy. Navyše sa cvičebný plán nedá nijako priamo zmeniť, iba vágne ovplyvniť, čo určite neuspokojí užívateľov ktorí vedia presne čo chcú.

Aplikácia Fitify

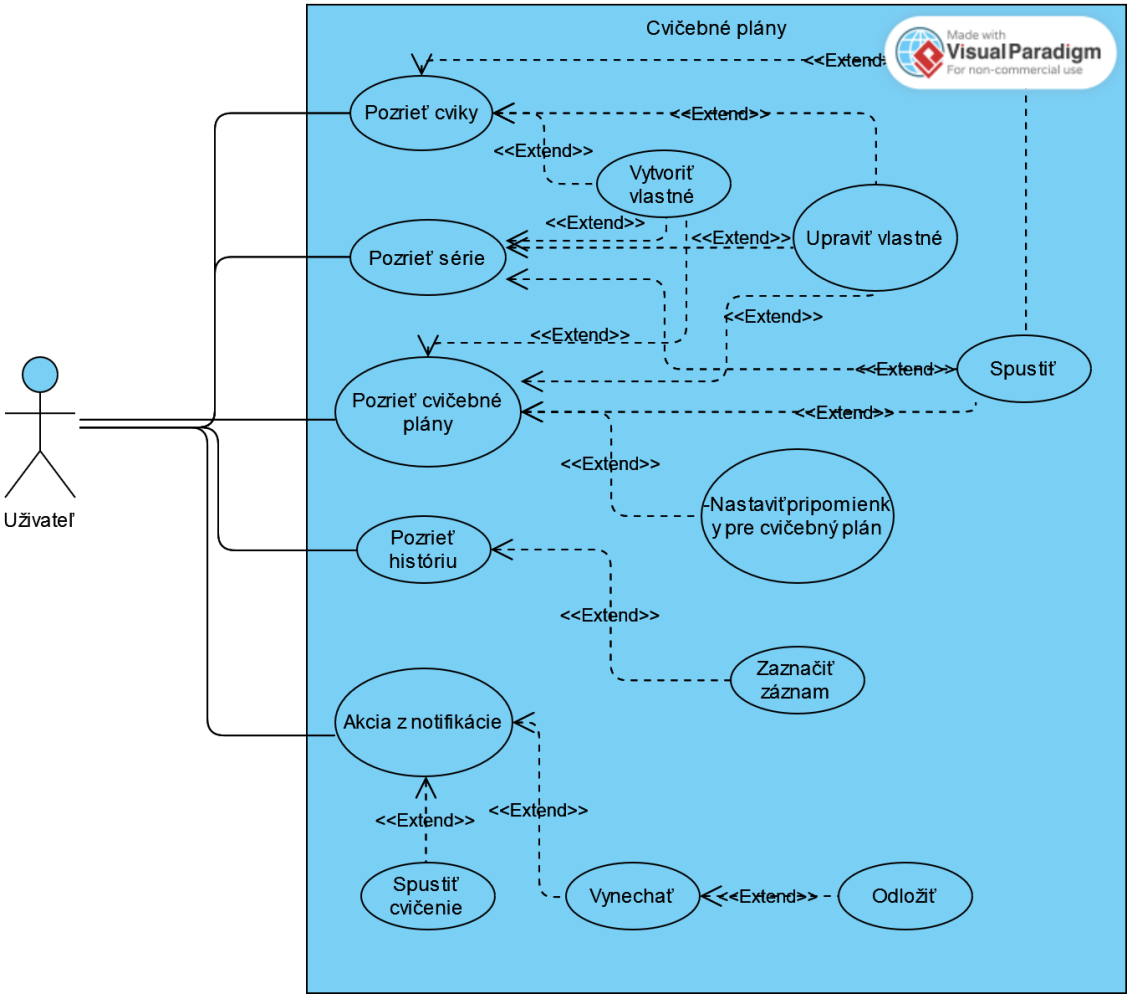
po zdĺhavom zbere údajov na začiatku, vyžaduje registráciu nového účtu a hneď po registrácií ponúka predplatné. Predplatné sa odmietnuť dá ale sú za ním zamknuté prakticky všetky cvičebné plány vrátane jednotlivých cvikov. Aplikácia je z veľkej časti po slovensky, cvičebné texty zas po anglicky alebo po česky.

Analýza navrhovanej aplikácie

Aplikácia Cvičebné plány má cieľom dať užívateľom slobodu a jednoduchosť. S pripravenými základnými cvikmi je jednoduché z nich vyskladať série s vlastným časovaním, alebo aj celý cvičebný plán. Možnosť pravidelných pripomienok na cvičenie podľa vlastného plánu vrátane kalendára, umožňujú rýchly a prehľadný spôsob prezerania histórie svojich výkonov.

PP

- Pozrieť si konkrétny cvik
- Pridať vlastný cvik
- Vytvoriť novú sériu
- Spustiť sériu
- Vytvoriť cvičebný plán
- Nastaviť pripomienky pre cvičebný plán
- Zmeniť cvičebný plán
- Pozrieť si históriu
- Spustiť cvičenie podľa plánu z notifikácie.



Návrh architektúry aplikácie

Aplikácia má vďaka Jetpack Compose a MaterialUI 3 všetky hlavné funkcie dostupné priamo na jednej obrazovke prostredníctvom jednoduchej spodnej navigačnej lišty s ikonami a popiskami – Cviky, Vlastné série, Cvičebné plány, História, ...

V sekcii Cviky by sa malo nachádzať pár základných cvikov, vrátane tlačidla + prostredníctvom ktorého bude možné pridať akýkoľvek vlastný cvik s názvom, textom, a obrázkom.

V sekcii Vlastné série bude možné vytvoriť si novú sériu ktorá bude pozostávať z jednotlivých reťazených cvikov, s dodatočnými možnosťami časovania a opakovaní.

Uložené série je potom možné kedykoľvek vybrať a spustiť.

V sekcii Cvičebné plány je možné vytvoriť si vlastný cvičebný plán, rôznou kombináciou sérií a cvikov s možnosťami plánovania na každých X dní alebo podľa dní v týždni.

V sekcii História je viditeľný kalendár informujúci o absolvovaní alebo absencii predošlých cvikov, sérií a cvičebných plánov.

Ukážka návrhu obrazoviek aplikácie

Nasledujúce obrazovky sú už WIP stav aplikácie. Vzhľadom na to že všetky budú mať pravdepodobne formu zoznamu a menu / navigácia je dole, vyzerajú dosť podobne. Niektoré by mali mať naľavo namiesto ikony obrázok nahraný užívateľom.

UX plány: Zobrazovanie detailov je zatiaľ plánované jednoduchým ťuknutím kdekoľvek na príslušný riadok. Úprava a mazanie jednotlivých položiek buďto v zobrazení detailu, alebo cez plávajúce menu po dlhom podržaní. Mazanie alternatívne možno aj posunutím (flick) do strany.

Počas vývoju zatiaľ len v angličtine:

FAB (floating action button) sú aktuálne súčasťou jednotlivých obrazoviek, takže sa pri prepnutí medzi nimi adekvátne menia.

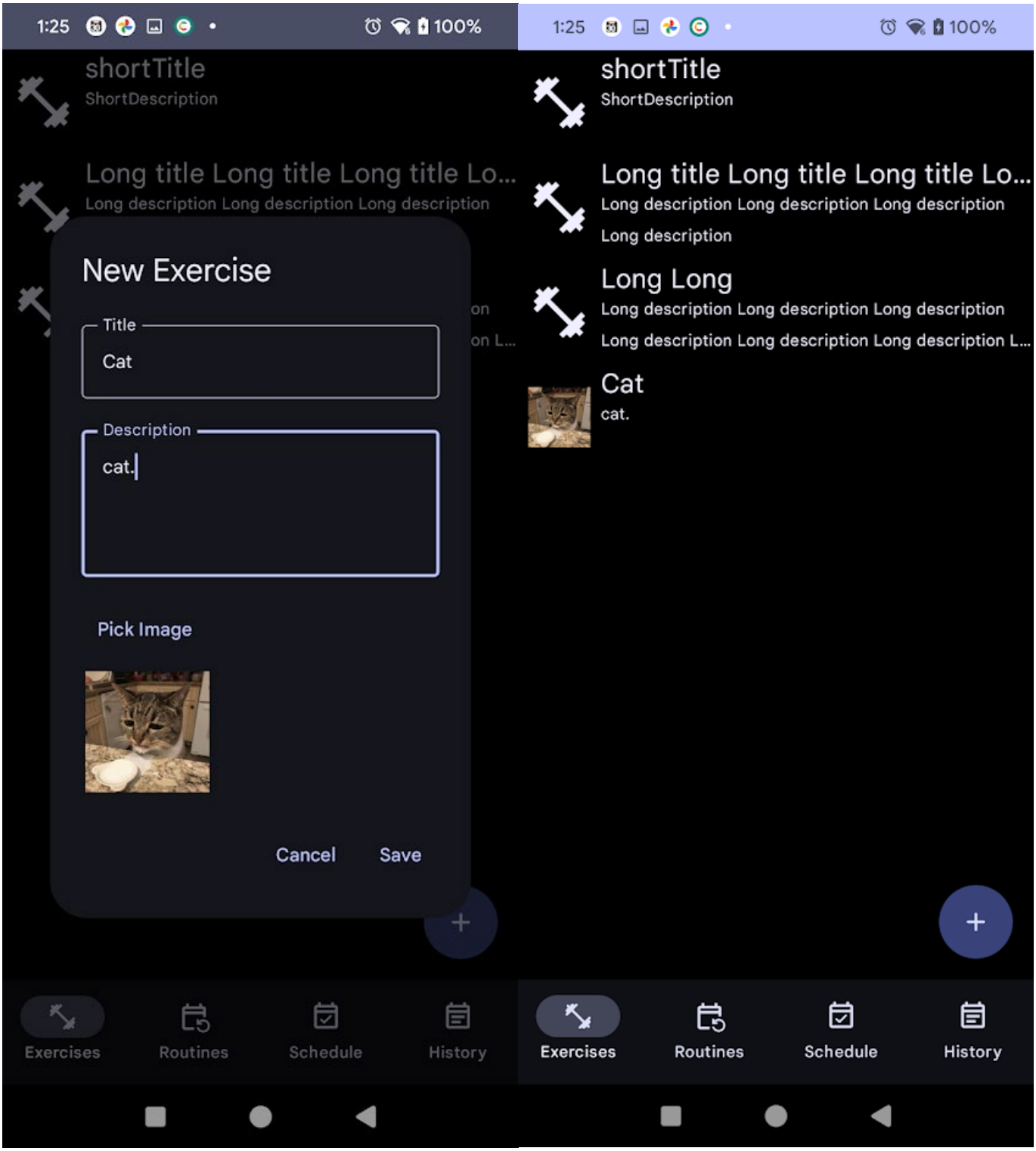
Obrazovka Sérií (Cvičebných rutín). Série by mali „spúšťateľné“ sety cvikov, ideálne s automatickým časovaním.

Obrazovka cvičebných plánov, aktuálne tiež ako zoznam. Je trochu otázne či to tak aj ostane, buďto budú plány jednoduchšie (v zozname), alebo bude plán iba jeden s trochu komplexnejšími možnosťami plánovania – (nie zoznam).

Nie všetky obrazovky musia mať FAB. V histórii Jedine asi jedine pre pridanie záznamu o cvičení mimo aplikácie. Uvidí sa.

Náhľad do developementu:

Objekty „Exercise“ sú už takmer hotové, ako aj ich pridávanie, mazanie, aj vlastná užívateľská rastrová grafika.



Popis implementácie

Aplikácia bola od začiatku navrhovaná s dôrazom na najnovšie dostupné komponenty ktoré Android development v súčasnosti poskytuje. Okrem ComposeUI to zahŕňa aj Material3 designové prvky a best UI/UX practices.

S použitím Material3 knižníc je napríklad použitie material design vektorových ikon takéto jednoduché:

```
floatingActionButton = {
    FloatingActionButton(
        onClick = {
            navCtl.navigate(Screens.ExerciseItemEditScreen.name)
        },
        shape = CircleShape
    ) {
        Icon(Icons.Rounded.Add, contentDescription = "Add")
    }
}
```

Namiesto obvyklého procesu sťahovania ikon zo material Google webu a ich manuálne vkladanie/odoberanie do/z resources v projekte.

Použitie komponentov ako napríklad Proto-DataStore, Kotlin flows a Kotlin de/serialization umožnilo výrazné uľahčenie vývoja aplikácie oproti riešeniam ako napríklad Room DB. Vďaka integrácií týchto komponentov sa napríklad v kóde nie je potrebné špecificky starať o manuálnu aktualizáciu údajov, o ktorú sa automaticky starajú Kotlin flows a ich integrácia s ComposeUI ktoré spustí automatickú re kompozíciu.

Napríklad na ukladanie „dátových objektov“ dokonca bez definície prototypov stačí kód:

```
@Serializable
data class HistoryList(
    @Serializable(with = PersistentListSerializer::class)
    val historyList: PersistentList<HistoryItem> = persistentListOf()
)

@Serializable
data class HistoryItem(
    val id: String = UUID.randomUUID().toString(),
    val routineId: String,
    val routineTitle: String,
    val startTime: Long,
    val endTime: Long,
    val completed: Boolean
)
```

```
object PersistHistoryListSerializer : Serializer<HistoryList> {
    override val defaultValue: HistoryList
        get() = HistoryList()

    override suspend fun readFrom(input: InputStream): HistoryList {
        return try {
            Json.decodeFromString(
                deserializer = HistoryList.serializer(),
                string = input.readBytes().decodeToString()
            )
        } catch (e: SerializationException) {
            e.printStackTrace()
            defaultValue
        }
    }

    override suspend fun writeTo(t: HistoryList, output: OutputStream) {
        output.write(
```

```
            Json.encodeToString(
                serializer = HistoryList.serializer(),
                value = t
            ).encodeToByteArray()
        )
    }
}
```

...to celé type-safe vrátane serializácie a deserializácie.

Ich následné ukladanie pomocou Proto-DataStore inštancie:

```
private const val HISTORY_DATA_STORE = "HistoryData.json"
val exercisesDS: DataStore<ExercisesList> by lazy {
    DataStoreFactory.create(
        serializer = PersistExercisesListSerializer,
        corruptionHandler = null
    ) {
        appContext.dataStoreFile(EXERCISES_DATA_STORE)
    }
}
```

Použitie takzvaných „remember“ premenných v UI ďalej eliminovalo potrebu boilerplate kódu a použitie vecí ako napríklad LiveView pre uchovávanie dát počas operácií ako je otáčanie obrazovky, prechod medzi obrazovkami v aplikácií, prechod na domovskú obrazovku a späť do aplikácie.

Po jednoduchej definícií je možné premennú používať ako obvykle:

```
var editedTitle by remember { mutableStateOf(exerciseItem?.exTitle ?: "") }
var editedDescription by remember { mutableStateOf(exerciseItem?.exDescription ?: "") }
var editedDuration by remember {
    mutableStateOf(
        exerciseItem?.durationSeconds?.toString() ?: "10"
    )
}
OutlinedTextField(
    value = textFieldValue,
    onValueChange = {
        textFieldValue = it
        editedTitle = it.text
    }
```

Pomocou využitia statickej formy DI (Dependency Injection) je možné zdieľať relevantné inštancie medzi jednotlivými compose funkciami a teda obrazovkami v aplikácií, ktoré by inak museli byť napríklad podávané cez parametre štýlom „top-down“.

Ukážka injektovania viewModelu ako závislosti composeable funkcie „cez konštruktor“ :

```
fun ExercisesScreen(
    navCtl: NavController,
    viewModel: ExercisesViewModel = ExercisesViewModel(ExTrApplication.datastoremodule)
) {
    val exerciseData by viewModel.exerciseDataFlow.collectAsState(initial =
ExercisesList())

//...
```

Centrálne riešená navigácia pomocou komponentu NavController zaručuje jednoduchú a bezproblémovú navigáciu z akejkoľvek obrazovky na akúkoľvek inú.

Využívanie ComposeUI komponentov „Scaffold“ a vlastného „RowItem“ taktiež umožňuje pekný jednotný vzhľad celej aplikácie, vrátane prvkov ako napríklad FAB (Floating Action Button).

Jednoduchý kód rozloženia Scaffold s FAB:

```
Scaffold(
        floatingActionButton = {
            FloatingActionButton(
                onClick = {
                    //...
                },
                shape = CircleShape
            ) {
                Icon(Icons.Rounded.Add, contentDescription = "Add")
            }
        }
    ) { innerPadding ->
        Text(
            "Hello World~!",
            modifier = Modifier
            )
    }
```

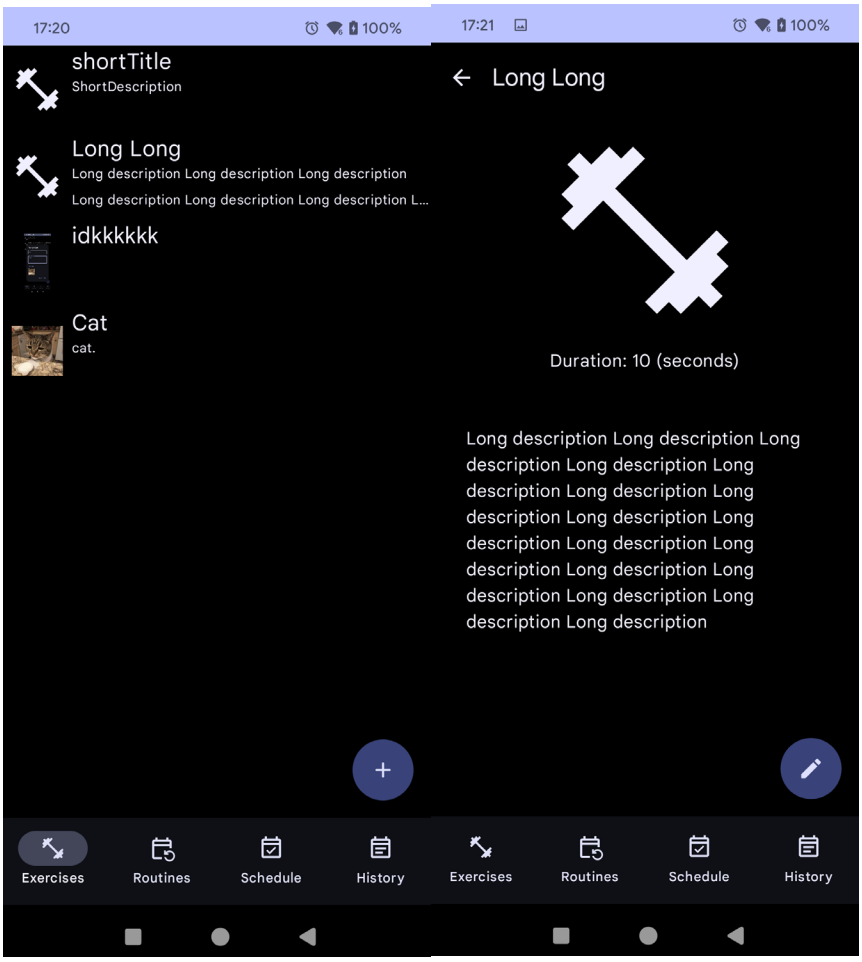
Integrácia funkcionality „Swipeable“ umožňuje pridať akcie posunutím RowItem do strán, čo je veľmi známy a populárny UX prvok.

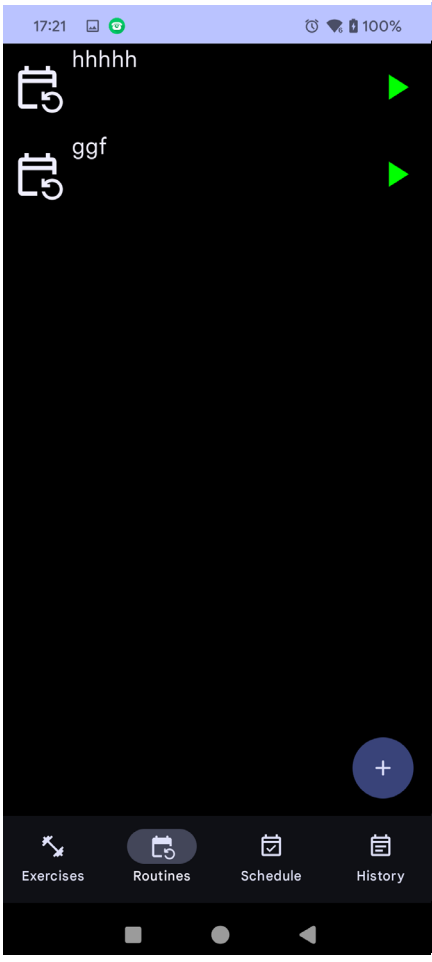
Úryvok kódu zo swipeable kontajnera:

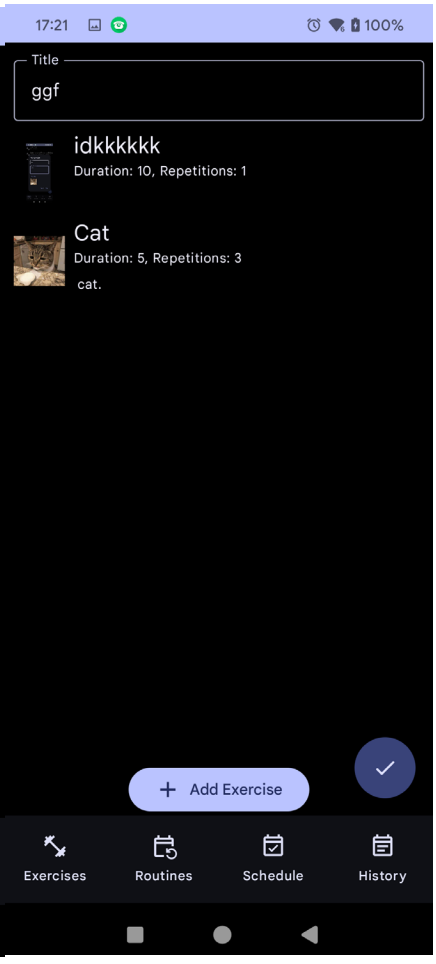
```
//kód podľa https://github.com/philipplackner/ProtoDataStoreGuide
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun <T> SwipeToDismissContainer(
    item: T,
    dismissDirections: Set<DismissDirection>,
    dismissToStartAction: (T) -> Unit,
    dismissToEndAction: (T) -> Unit,
    animationDuration: Int = 500,
    content: @Composable (T) -> Unit
) {
    var isDismissedToStart by remember { mutableStateOf(false) }
    var isDismissedToEnd by remember { mutableStateOf(false) }
    val state = rememberDismissState(
        confirmValueChange = { value ->
            when (value) {
                DismissValue.DismissedToStart -> {
                    isDismissedToStart = true
                    true
                }

                DismissValue.DismissedToEnd -> {
                    isDismissedToEnd = true
                    true
                }

                else -> false
            }
        }
    )
```

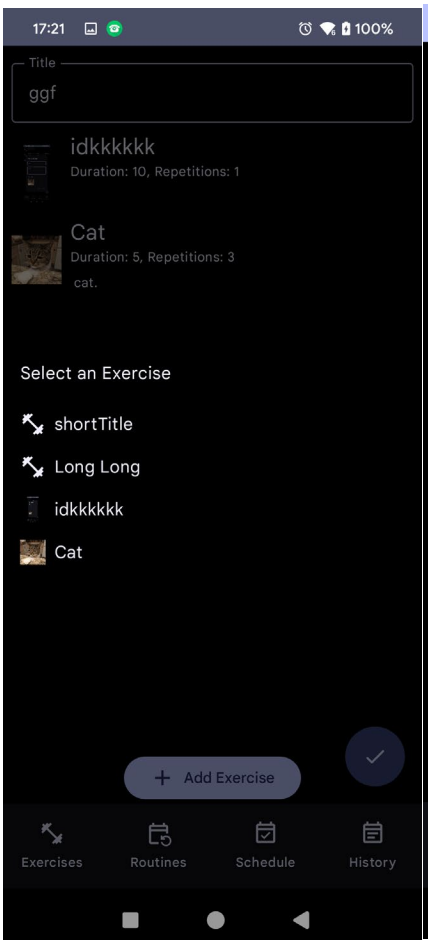
```
    LaunchedEffect(key1 = isDismissedToStart) {
        if (isDismissedToStart) {
            delay(animationDuration.toLong() - 250) //shit's bugged
https://issuetracker.google.com/issues/240599812
            dismissToStartAction(item)
        }
    }

    LaunchedEffect(key1 = isDismissedToEnd) {
        if (isDismissedToEnd) {
            delay(animationDuration.toLong() - 250)
            dismissToEndAction(item)
        }
    }

    AnimatedVisibility(
        visible = !isDismissedToStart && !isDismissedToEnd,
        enter = expandVertically(
            animationSpec = tween(durationMillis = animationDuration),
            expandFrom = Alignment.Top
        ),
        exit = shrinkVertically(
            animationSpec = tween(durationMillis = animationDuration),
            shrinkTowards = Alignment.Top
        ) + fadeOut()
    ) {
        SwipeToDismiss(
            state = state,
            background = {
                DismissBackground(swipeDismissState = state)
            },
            dismissContent = { content(item) },
            directions = dismissDirections
        )
    }
}
```

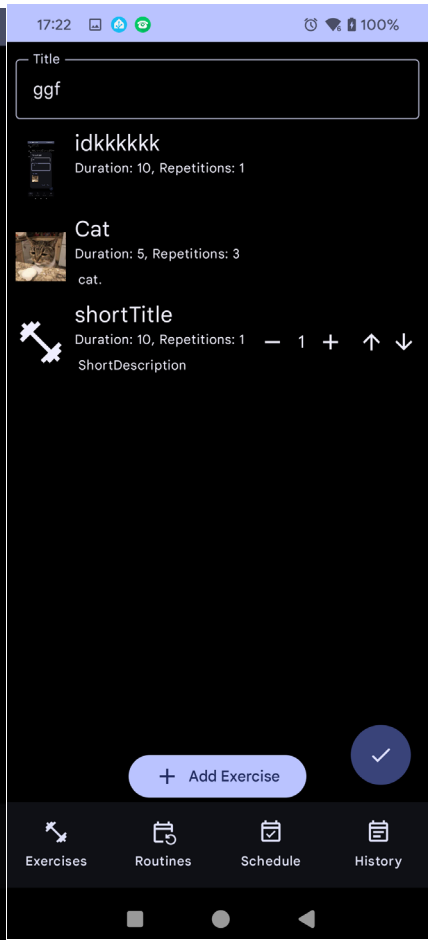
Snímky obrazoviek aplikácie:

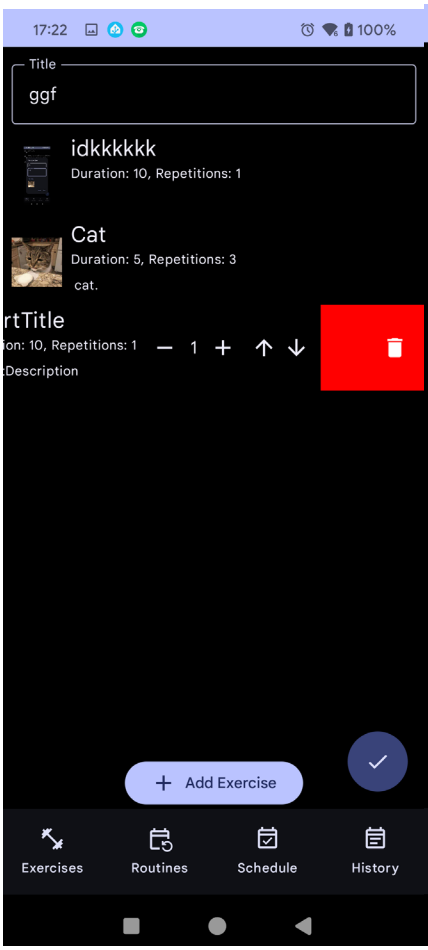


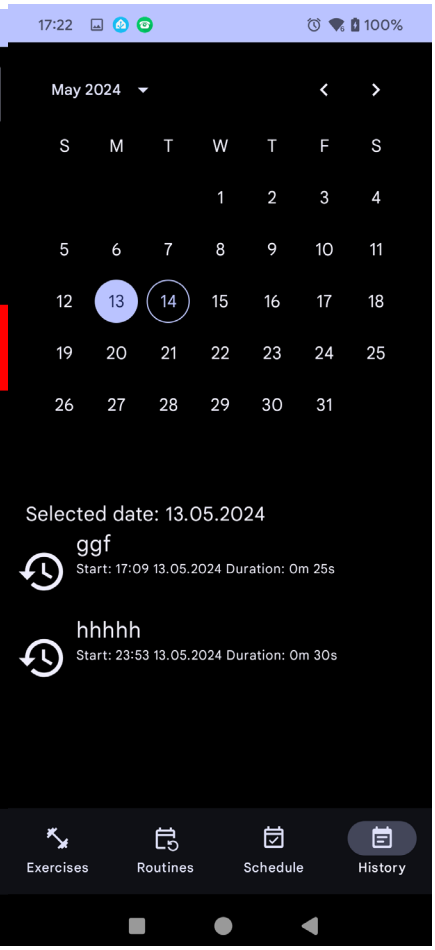





17:21
100%
Title
ggf
idkkkkkk
Duration: 10, Repetitions: 1
Cat
Duration: 5, Repetitions: 3
cat.
Select an Exercise
shortTitle
Long Long
idkkkkkk
Cat
Add Exercise
Exercises
Routines
Schedule
History


17:22
100%
Title
ggf
idkkkkkk
Duration: 10, Repetitions: 1
Cat
Duration: 5, Repetitions: 3
cat.
shortTitle
Duration: 10, Repetitions: 1
1
ShortDescription
Add Exercise
Exercises
Routines
Schedule
History


17:22
100%
Title
ggf
idkkkkkk
Duration: 10, Repetitions: 1
Cat
Duration: 5, Repetitions: 3
cat.
rtTitle
on: 10, Repetitions: 1
1
Description
Add Exercise
Exercises
Routines
Schedule
History


17:22
100%
May 2024
S M T W T F S
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
Selected date: 13.05.2024
ggf
Start: 17:09 13.05.2024 Duration: 0m 25s
hhhhh
Start: 23:53 13.05.2024 Duration: 0m 30s
Exercises
Routines
Schedule
History

Záver:

Pri návrhu aplikácie som sa snažil aby bola prívetivá nielen pre užívateľov, ale aj pre vývojárov. Verím že sa mi to podarilo. Aplikácia taktiež splnila väčšinu svojich cieľov keďže umožňuje pridávanie vlastných cvikov s názvami, popismi, trvaním aj rastrovými obrázkami, úpravu cvikov, mazanie cvikov, podobne aj všetky tieto operácie s vlastnými sériami vrátane „prehrávania“ sérií počas cvičenia, a históriou so záznamami o spustených sériách.

Časť aplikácie ohľadom plánovania z časových dôvodov nie je dorobená. Implementácia by však bola podobne jednoduchá ako pri ostatných funkcionalitách t.j. pridanie UI pre nastavenie času, frekvencie pripomienok a konkrétnej série cvikov, nastavenie notifikácie pre najbližší vhodný čas, a v notifikácií buď prehranie konkrétnej série jej otvorením, alebo jej odstránením z notifikácií so záznamom o odmietnutí v histórií.