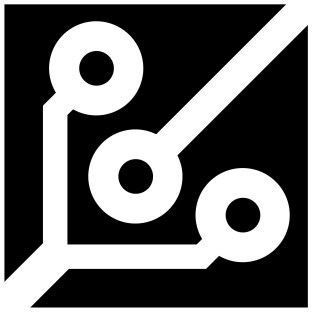
Take Your Pill

mobilní aplikace

SPŠE V Úžlabině



Vojtěch Hořánek

I4.D

13.03.2021

„Prohlašuji, že jsem tuto práci vypracoval samostatně a použil jsem literárních pramenů a informací, které cituji a uvádím v seznamu použité literatury a zdrojů informací.“

V Praze dne ....

podpis autora

Anotace

Předložená práce je mobilní aplikace pro systém Android, která slouží k připomínání užití léků. V aplikaci je implementovaná historie připomínání a užívání léků, statistiky a grafy. Uživatel snadno získá přehled, jaké a kdy léky vynechává a může si nastavit opakované připomínání, aby již lék nezmeškal. Design aplikace je udělán tak, aby odpovídal material designu a jeho nejnovějším trendům. Aplikace je napsána v jazyce Kotlin a využívá moderních knihoven a technologií.

Anotation

The presented work is a mobile application for the Android operating system which reminds its users to take their pills. History, statistics, and graphs are implemented in the application. The user can effortlessly get an overview of which pills at what time did they miss and can set repeating reminders, so they do not forget them the next time. The application follows the material design guidelines and its latest trends. Kotlin was used as the programming language and utilizes modern libraries and technologies.

Obsah

Úvod **1**

1 Návrh aplikace **2**

2 Implementace aplikace **3**

2.1 Uživatelské rozhraní 4

2.1.1 Úvodní obrazovka 4

2.1.2 Domovská obrazovka 4

2.1.2.1 Detail léku 5

2.1.2.2 Nový lék 5

2.1.3 Historie 7

2.1.3.1 Přehled 7

2.1.3.2 Grafy 7

2.1.4 Nastavení 7

2.1.4.1 O aplikaci 7

2.2 Programová implementace 7

2.2.1 Databáze 7

2.2.2 Připomínání 7

Závěr **8**

Úvod

Cílem této práce bylo vytvořit aplikaci, která uživatelům usnadní pravidelné užívání léků jejich připomínáním, sledováním historie a statistik. Pro každý lék lze nastavit aby se připomínal pouze určitý počet dní nebo aby se připomínal v cyklu X dní aktivní Y dní neaktivní.

Práce se skládá z mobilní aplikace pro systém android. Aplikace je navržena co nejjednodušeji a rozdělena do dvou hlavních sekcí: léky a historie. V sekci „léky“ uživatel nalezne léky, které si do aplikace přidal. V seznamu léků je zobrazen jejich název, popis, barva, fotografie, časy připomínek a příjem. V sekci „historie“ může uživatel sledovat užití svých léku, zobrazí se mu kompletní historie (včetně kdy a jestli si lék vzal, kdy mu byla poslána připomínka a kolik prášků si vzal) a grafy zobrazující souhrnné informace. Součástí práce je i propagační a informační plakát.

Kapitola 1

Návrh aplikace

Design aplikace jsem navrhoval před samotnou implementací, nutno ale dodat, že při implementaci prošel design několika iteracemi. Aplikaci jsem původně koncipoval jako jedinou hlavní obrazovku, kde by se uživateli ukázali všechny důležité informace. Postupem času se toto řešení ukázalo jako nevhodné a nepraktické, zvolil jsem proto více tradiční postup a to rozdělení aplikace do tří přehledných sekcí: *Léky*, *Historie* a *Nastavení*. Každá obrazovka obsahuje velký nadpis a teprve potom samotný obsah. Mimo spodních dialogů[1] není nadpis nijak ohraničen, pouze je odsazen. Změny mezi různými obrazovkami doprovázejí animace, které jsou implementovány podle Material Designu. Aplikace díky těmto animacím vypadá svižněji.

Celé rozhraní jsem upravil pomocí vlastních stylů, které vycházejí ze stylů Material Design [3]. Pro některé prvky aplikace, jmenovitě titulky a tlačítka, jsem použil písmo *Jost* [8]. Ikony použité v aplikaci jsou z knihovny Material Design Icons [9], ikonu aplikace jsem získal od Austina Andrewse [10] a grafika léků je dostupná na GitHubu pod názvem *material-icons* [11]. Nechybí ani podpora světlého a tmavého designu, který se dá přepínat v nastavení (VOJTO TODO REFERENCE).

[1] tím myšleno BottomSheetDialogFragment

Kapitola 2

Implementace aplikace

Při vytváření aplikace jsem vycházel ze zadání a využil jsem vlastních znalostní a zkušeností. Na naprogramování aplikace jsem použil programovací jazyk Kotlin. Zvolil jsem ho proto, že je preferovaný společností Google a oproti jazyku Java má mnoho výhod. Mnoho Android knihoven vychází právě pro Kotlin a tak mi jeho použití ulehčilo mnoho práce při programování. Jmenovitě knihovny z rodiny Android Jetpack [4] jsem použil hojně.

Uživatelské rozhraní aplikace je napsáno v jazyce XML. Pro jeho manipulaci jsem použil knihovny *ViewBinding* [1] a *DataBinding* [2].

Při samotném vývoji jsem používal vývojové prostředí *Android Studio* [5] a emulátor *Android Emulator*.

Aplikace je napsána tak, aby odpovídala architektuře **Model-View-ViewModel**. Znamená to, že každá obrazovka má svůj *ViewModel* a každá datová sekce má svůj *Repozitář*. Tyto třídy jsou odděleny od samotných fragmentů a aktivit. Pro získávání dat asynchronně používá aplikace třídu `LiveData` [16]

ViewModel je třída obsahující data a metody pro její fragment/aktivitu, která má vlastní životní cyklus. Díky ViewModelu data přežijí změnu konfigurace jako například otočení obrazovky.

Repozitář (repository) je třída, která shromažďuje data z různých zdrojů a nabízí je ve vhodné formě ostatním třídám (například může data uchovat v mezipaměti). V této aplikaci přistupuje repozitář pouze do databáze a ve většině případů přímo volá metody implementované v databázové vrstvě.

2.1 Uživatelské rozhraní

Hlavní obrazovka aplikace je rozdělena na tři části: *Léky*, *Historie* a *Nastavení*. Uživatel mezi těmito sekcemi přepíná pomocí prvku `BottomNavigationView`. Do sekce *Historie* a na obrazovku *Přidat lék* se může uživatel dostat i pomocí zkratky[1] z domovské obrazovky. Celá aplikace obsahuje pouze tři aktivity: MainActivity, AboutActivity a AppIntroActivity. Všechny ostatní obrazovky jsou implementovány jako fragmenty a pro jejich navigaci byla použita knihovna *Navigation* [6].

2.1.1 Úvodní obrazovka

Pro přidání úvodní obrazovky jsem použil knihovnu *material-intro* [7]. Tato knihovna se postará o všechen layout a logiku úvodní obrazovky. Má jednoduché API a tak jediné, co jsem do aplikace přidal, byla aktivita `AppIntroActivity` dědící ze třídy `IntroActivity` a v ní přidal slidy pomocí metody `addSlide()`. Obrázky ve slidech jsem vyfotil v android emulátoru a upravil v programu *GIMP*. Úvodní obrazovka se spustí pouze když uživatel aplikaci spustí poprvé. Toto je zajištěno tak, že do trvalé paměti aplikace[2] se po ukončení této aktivity uloží proměnná `firstRun` s hodnotou `false`. Snímek obrazovky výsledku lze vidět na straně 9 (2.1a).

2.1.2 Domovská obrazovka

Domovská obrazovka neboli sekce „Léky“ (`HomeFragment`) je fragment obsahující pouze `RecyclerView` (dále jen recycler) a `ExtendedFloatingActionButton` (dále jen FAB). FAB se při posunutí recycleru zmenší. Zvětší se až když seznam posuneme na začátek.

Pro zobrazení dat v recycleru je potřeba mít `RecyclerAdapter`. Tato třída se stará o zobrazování dat s příslušným `ViewHolderem`. Adaptér, který je nastavený na tomto recycleru se jmenuje `AppRecyclerAdapter`. Každý *ViewHolder* pro třídy *Pill* obsahuje kartu, jejichž layout je definován v souboru `item_pill.xml`. Na kartě se zobrazují všechny potřebné informace o léku: název, popis, barva, fotografie, připomínky a příjem. Pokud uživatel nepotvrdil nejnovější připomínku v posledních 30 minutách, zobrazí se na kartě i výzva k potvrzení. Po kliknutí na kartu se otevře detail příslušného léku. Snímek obrazovky sekce

[1] zktratky využívají App Shortcuts API
[2] trvalou pamětí je myšleno úložiště SharedPrefs

„Léky“ lze vidět na straně 9 (2.1a).

ViewHolder (doslovný překlad „držitel pohledu“) je třída, která se stará o zobrazení jedné položky v RecyclerView Adaptéru. Má za úkol nastavit layout tak, aby odpovídal vstupním datům (např. jednomu léku). Třídu si musíme definovat sami pro každý typ položky, které chceme zobrazovat.

AppRecyclerAdapter je `RecyclerAdapter` založený na `ListAdapter`. Tento adaptér se používá pro většinu seznamů v aplikaci, jelikož umožnujě přidat titulek a zobrazit prázdný stav. Toto je dosaženo přepsáním metody `submitList` a dosazením speciálních položek (`HeaderItem` a `EmptyItem`). Adaptér podporuje třídy, které dědí z `BaseModel`, jmenovitě `Pill`, `HistoryPillItem`, `HeaderItem` a `EmptyItem`.

2.1.2.1 Detail léku

Obrazovka léku (`DetailsFragment`) je implementována jako fragment s layoutem `fragment_details.xml`. Na obrazovce lze vidět název léku, jeho popis a fotografii (pokud tyto položky má), připomínky a příjem. Při dlouhém podržení na fotografii se fotografie zobrazí v plné velikosti. Pokud obrazovku otevřeme z oznámení, nebo má lék nepotvrzenou připomínku v posledních 30 minutách, zobrazí se nad titulkem karta vyznívající k potvrzení této připomínky. Na spodní části obrazovky jsou tlačítka „smazat“, „historie“ a „upravit“. Tlačítko „smazat“ otevře dialog, kde uživatel může zvolit, zda chce smazat pouze lék a zachovat jeho historii, nebo ho smazat i s historií. Dialog je implementovaný ve třídě `DeleteDialog`. Tlačítko „historie“ otevře dialog, ve kterém se zobrazí historie pro tento lék. Více o tomto dialogu naleznete v sekci „Přehled“ na straně 7. Tlačítko „upravit“ otevře `EditFragment`, kde může uživatel lék upravit. Více o této obrazovce v následující sekci.

2.1.2.2 Nový lék

Obrazovka „Nový lék“ (`EditFragment`) má dvě funkce. Slouží jako obrazovka pro přidávání nového léku a zároveň se používá pro úpravu léku. Titulek obrazovky se mění podle použití. Opět je implementována jako fragment. Pro každý lék je možno nastavit název a popis. Tyto dvě hodnoty se zapisují do prvku `TextInputLayout` z knihovny *material*. Následuje vybrání barvy pro lék. Uživatel má na výběr ze 7 barev: modrá, tmavě modrá, tyrkysová, zelená, žlutá, oranžová a červená. Vybírání barvy je implementováno pomocí

prvku `RecyclerView` s atributem `orientation` nastaveným na hodnotu `horizontal`. Jednotlivé barvy jsou definované třídou `PillColor`. Dále si uživatel nastaví připomínky. Při kliknutí na připomínku nebo na tlačítko „přidat připomínku“ se zobrazí `ReminderDialog` kde uživatel může upravit/vytvořit připomínku. U připomínky lze nastavit i množství léku, jaké si v daný čas má uživatel vzít. Pro jeden lék nelze nastavit dvě připomínky se stejným časem, každá připomínka musí mít unikátní čas. V neposlední řade lze léku nastavit příjem. Uživatel si může zvolit, zda lék bere neustále, jen určitý počet dní a nebo v cyklu X dní aktivních, Y dní neaktivních. Tento prvek[1] je implementován v `PillOptionsView`, který také dědí z `LinearLayout` a používá layout `layout_pill_options_view.xml`. Veškerá logika výběru příjmu je implementována v této třídě. Jediné o co se `EditFragment` musí postarat, je získání `ReminderOptions` (vysvětleno v kapitole 2.2.2) z tohoto prvku pomocí metody `getOptions()`.

Výběr fotografie. K léku lze přidat i fotografii. Prvek na výběr fotografie je implementován v `ImageChooserView`. Tato třída dědí z `LinearLayout` a používá layout definovaný v `layout_image_chooser.xml`. Pokud lék již nějakou fotografii obsahuje, prvek zobrazí tlačítko na její odstranění. Při výběru fotografie je použita knihovna *EasyPermissions* [13] pro zajištění potřebných oprávnění a upravená verze knihovny *imagepicker* [14]. Knihovnu jsem upravil tak, aby respektovala vzhled aplikace. Zaprvé již nepoužívá standardní `AlertDialog` nýbrž `BottomSheetDialog`. Také vzhled tohoto dialogu byl upraven, aby odpovídal všem ostatním dialogům v aplikaci. Uživatel si může zvolit, zda chce fotografii vybrat z galerie, nebo chce vyfotit fotografii novou. Po vybrání/vyfocení se uživateli ukáže obrazovka, kde může fotografii upravit. Pro upravení fotografii jsem užil knihovnu *uCrop* [15], kterou jsem také upravil. Oproti originální verzi se liší v použití prvků na navigaci, ikonách, podporou automatického tmavého vzhledu a sladěním do vzhledu aplikace. Knihovna uživateli umožní fotografii oříznout, otočit a škálovat. Pro plynulejší úpravy jsem zvolil „native“ verzi knihovny. Knihovna tak využívá kód napsaný v C++, který je rychlejší a optimalizovaný pro jednotlivé architektury, avšak přidá k velikosti aplikace cca 1.5 MB.

[1] prvkem je myšlen prvek uživatelského rozhraní, neboli `view`

PillColor je třída, používaná pro ukládání barvy léku. Obsahuje atributy `resource` a `isChecked`. Atribut `resource` ukládá `id` barvy uložené v *App resources* [12]. Atribut `isChecked` vyjadřuje, zda je barva vybraná. Tento atribut se používá k zobrazování seznamu barev a zjištění, jakou barvu uživatel vybral.

2.1.3 Historie

2.1.3.1 Přehled

2.1.3.2 Grafy

2.1.4 Nastavení

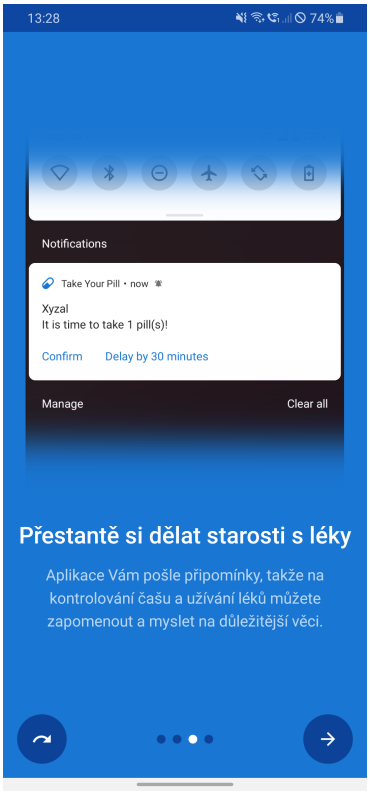
2.1.4.1 O aplikaci

2.2 Programová implementace

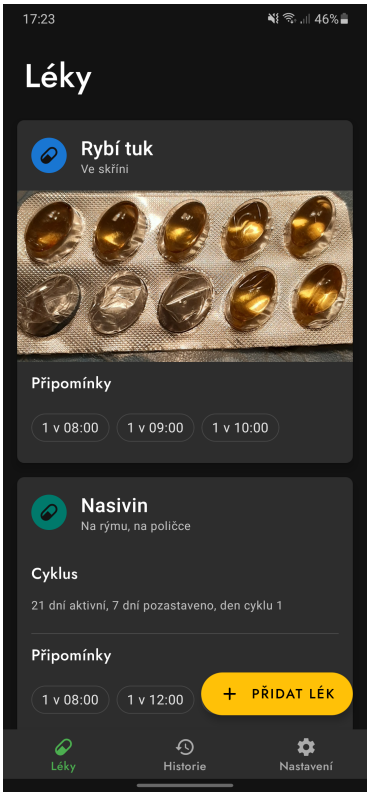
2.2.1 Databáze

2.2.2 Připomínání

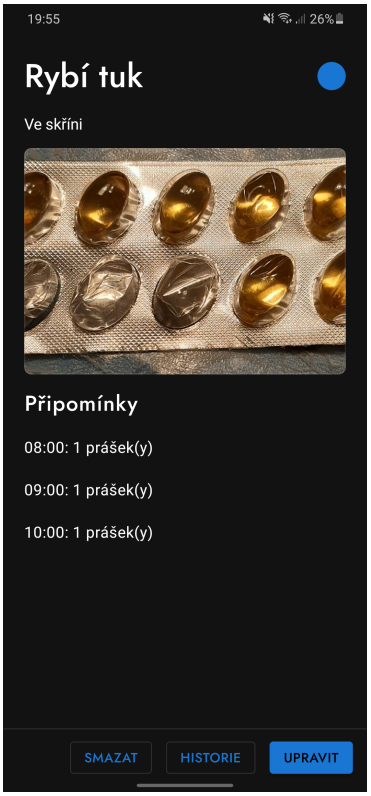
Závěr



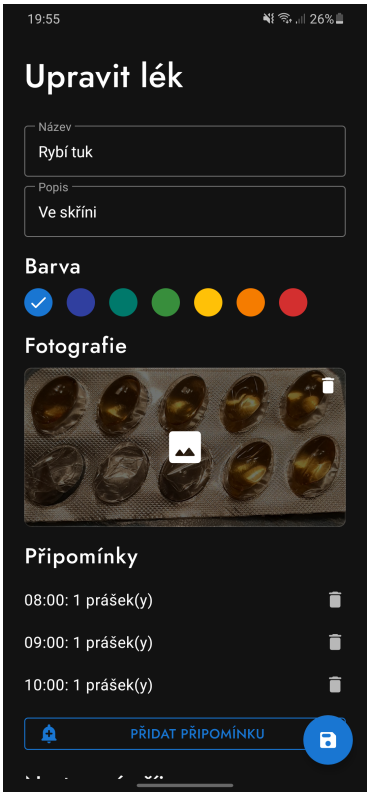
(a) Úvodní obrazovka



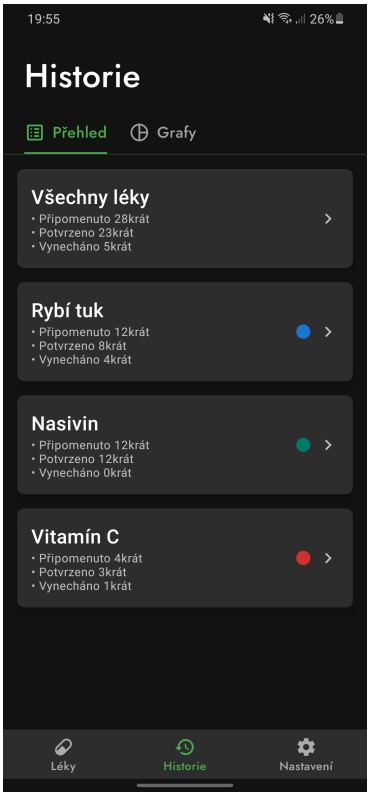
(b) Domovská obrazovka



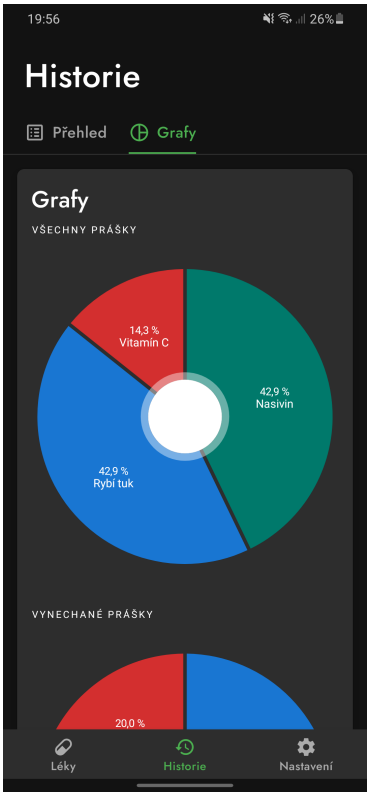
(c) Detail léku



(d) Úprava léku



(e) Přehled historie



(f) Grafy

Obrázek 2.1: Snímky obrazovky z aplikace

Bibliografie

[1] View Binding

https://developer.android.com/topic/libraries/view-binding

[2] Data Binding Library

https://developer.android.com/topic/libraries/data-binding

[3] Material Design

https://material.io/design

[4] Android Jetpack

https://developer.android.com/jetpack

[5] Android Studio

https://developer.android.com/studio

[6] Navigation

https://developer.android.com/guide/navigation

[7] material-intro

https://github.com/heinrichreimer/material-intro

[8] Jost - Google Fonts

https://fonts.google.com/specimen/Jost

[9] Material Design Icons

https://material.io/resources/icons/

[10] pill - Austin Andrews

https://materialdesignicons.com/icon/pill

[11] ShimonHoranek - material-icons

https://github.com/ShimonHoranek/material-icons

[12] App resources overview

https://developer.android.com/guide/topics/resources/providing-resources

[13] EasyPermissions

https://github.com/googlesamples/easypermissions

[14] Image Picker Library for Android

https://github.com/Dhaval2404/ImagePicker

[15] uCrop - Image Cropping Library for Android

https://github.com/Yalantis/uCrop

[16] LiveData Overview

https://developer.android.com/topic/libraries/architecture/livedata