FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Aplikace pro sledování osobních cílů
Student:	Tomáš Glázr
Vedoucí:	Ing. Marek Kodr
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Cílem bakalářské práce je vytvořit mobilní aplikaci pro operační systém Android, která pomůže uživatelům s dosažením osobních cílů a pravidelném zlepšování. Aplikace umožní sledování a plánování různých aktivit a cílů, porovnání s ostatními uživateli a pravidelný report dosažených výsledků.

Proveďte následující kroky:
1) Analyzujte konkurenci na trhu a potřeby uživatelů.
2) Na základě výsledků proveďte analýzu a definujte požadavky na aplikaci.
3) Implementujte aplikaci pro operační systém Android.
4) Otestujte aplikaci a implementujte analytics.
5) Zpřístupněte aplikaci uživatelům.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 21. října 2022 v Praze.

Bakalářská práce

APLIKACE PRO SLEDOVÁNÍ OSOBNÍCH CÍLŮ

Tomáš Glázr

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Marek Kodr
11. května 2023

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především vedoucímu práce Ing. Markovi Kodrovi za cenné rady a vstřícnost při konzultacích. Také bych chtěl vyjádřit svou vděčnost přátelům za podporu, kterou mi poskytovali během celého mého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 11. května 2023

Abstrakt

Předmětem této bakalářské práce je analýza, návrh a implementace Android aplikace pro sledování osobních cílů. V práci je provedena analýza konkurence na trhu, na základě které jsou stanoveny požadavky aplikace a vytvořen návrh aplikace. Aplikace umožňuje zaznamenávání svých osobních cílů a potřebných úkolů k jejich dosažení. Hlavním přínosem této aplikace je možnost sdílení úspěchů s přáteli. Práce dále pojednává o použitých technologií použitých k implementaci aplikace. Výstupem této práce je funkční aplikace dostupná ke stažení z oficiálního obchodu Google Play.

Klíčová slova mobilní aplikace, osobní cíle, Android, Kotlin, Firebase

Abstract

The goal of this bachelor's thesis is the analysis, design, and implementation of an Android application for tracking of personal goals. The thesis includes a market competition analysis, on which the application's requirements are based and the design of the application is created. The application allows for tracking of personal goals and the tasks required to complete them. The main advantage of this application is the ability to share successes with friends. The thesis also discusses the technologies used to implement the application. The output of the thesis is a functional application available for download from the official Google Play store.

Keywords mobile application, personal goals, Android, Kotlin, Firebase

Seznam zkratek

ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
JSON	JavaScript Object Notation
MVC	Model-View-Controller
MVP	Model-View-Presenter
MVVM	Model-View-ViewModel
OS	Operační systém
SDK	Software Development Kit
SoC	Separation of Concerns
SQL	Structured Query Language
UI	User Interface

Úvod

V dnešní uspěchané době je čím dál těžší nezapomenout na osobní rozvoj a sebevzdělávání. Proto je v životě důležité, si řádně a přesně zadefinovat své dlouhodobé i krátkodobé osobní cíle, kterých chceme v životě dosáhnout. Abychom tyto cíle úspěšně naplnili je nezbytné vynaložit odpovídající úsilí. Mnoho lidí ale potřebuje více motivace aby zůstali vytrvalí k jejich plnění.

Cílem práce je vytvoření mobilní aplikace pomáhající uživatelům k definování a dosahování svých osobních cílů, sledování jejich průběhu, motivace k jejich plnění a sdílení svých úspěchů mezi přáteli.

V první části této práce bude provedena analýza existujících aplikací pro sledování osobních cílů, na základě které budou stanoveny funkční a nefunkční požadavky aplikace. V druhé části práce bude vytvořen návrh aplikace pro implementaci. V třetí části práce je provedena samotná implementace aplikace s popisem zvolených technologií a knihoven a dále také distribucí a monitorování aplikace. Ve čtvrté části aplikace je nakonec provedeno uživatelské testování z kterého bude poté navrhnut další možný vývoj této aplikace.

Výsledkem práce bude funkční aplikace na android, která bude dostupná z oficiálního obchodu Google Play.

Kapitola 1
Analýza

1.1 Motivace

Základem úspěchu ke splnění osobních cílů je jejich důkladné zaznamenání a pravidelné sledování. Bohužel mnoho lidí se po stanovení cílů potýká s nedostatkem motivace a později i s frustrací, když nedosáhnou očekávaných výsledků. Ve studii provedené American Psychological Association (APA) v roce 2020 bylo zjištěno, že sdílení cílů s lidmi ke kterým vzhlížíme vede k vyšší motivaci a odhodlání v jejich plnění. [1] Výzkum provedený ASTD (American Society of Training and Development) dokonce ukázal, že lidé mají až o 65 % vyšší šanci dosáhnout svého cíle pokud jej sdílí s jinou osobou a až o 95 % pokud cíli dají termín splnění a sdílí svůj průběžný pokrok. [2]

Aplikace pro sledování osobních cílů by tedy měla obsahovat možnost sdílení svých cílů a pokroku s přáteli. Důležitou funkcí by mělo být také možnost rozdělení cílů na dlouhodobé a krátkodobé, což umožní uživatelům jasně oddělit cíle, které se stanou zvykem pro uživatele a budou jej doprovázet celý život a cíle, kterých chce uživatel dosáhnout v omezeném čase. Dále by aplikace měla mít přehledný a intuitivní design, možnost zálohovat data účtem v aplikaci aby v případě ztráty nebo změny zařízení uživatel nepřišel o svá data. Protože by aplikace měla být dostupná co nejširšímu okruhu lidí, měla by být zdarma.

Na trhu je dnes k dispozici široká škála aplikací zaměřujících se na osobní rozvoj a plánování osobních cílů. Tato kapitola se zaměří na analýzu existujících řešení na trhu a následně na stanovení funkčních a nefunkčních požadavků nově vyvíjené aplikace.

1.2 Existující řešení

Výběr aplikací byl proveden na základě několika kritérií. Prvním kritériem byla popularita aplikace, která byla určena na základě počtu stažení, hodnocení a recenzí od uživatelů. Druhým kritériem byly nabízené funkcionality aplikací. Analýza se zaměřuje pouze na aplikace dostupné na operační systém Android. Protože v obchodu Google Play existuje mnoho aplikací, bylo vybráno 5 aplikací pro analýzu. Další aplikace ale zpravidla nabízejí stejné nebo podobné funkcionality.

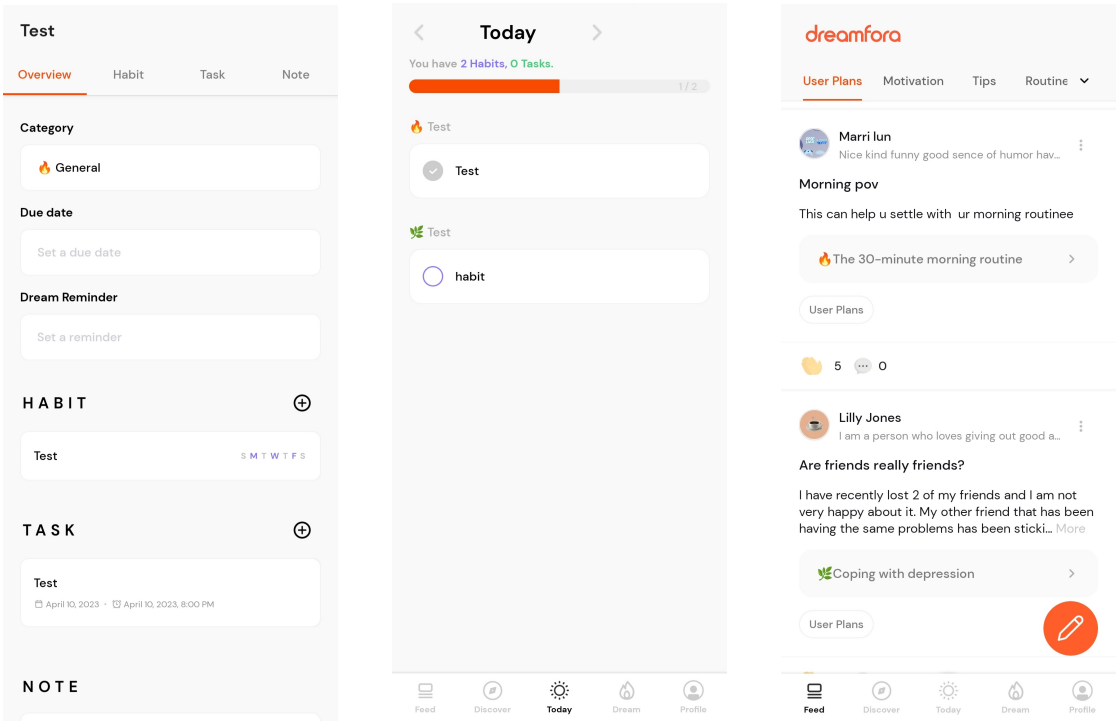
1.2.1 Dreamfora

Počet stažení: více než 1 000 000
Hodnocení: 4,2 / 5

Aplikace Dreamfora patří mezi jednu z nejstahovanějších aplikací v oblasti rozvoje osobních cílů. Hlavní sekcí aplikace je záložka „Dream“, kde si uživatel zadá své cíle s jejich termínem splnění. K těmto cílům lze poté přidat „Tasks“, což jsou jednorázové úkoly, které lze uskutečnit právě jednou a po splnění jsou dokončené, a „Habits“, což jsou opakující se úkoly, kterým lze nastavit dny v týdnu a čas jejich opakování.

Po otevření aplikace uživatele přivítá obrazovka s úkoly, které má uživatel splnit během aktuálního dne. Dále aplikace umožňuje vytváření cílů pomocí odborníky předpřipravenými šablonami na cíle i s radami, jak nejlépe těchto cílů dosáhnout.

Důležitou součástí aplikace jsou notifikace na plnění úkolů. Pro každý úkol v nastavené době uživateli pošle oznámení na telefon, že by měl daný úkol splnit. V nastavení lze pak nastavit i 2 speciální oznámení - ranní motivační oznámení obsahující povzbudivý citát na daný den a večerní notifikace se souhrnem dosažených úspěchů v současném dni. Dle popisu vývojářů aplikace obsahuje až 1000 motivačních citátů. Nakonec aplikace disponuje záložkou „Feed“, kde uživatel nalezne příspěvky, motivace a plány ostatních uživatelů. Bohužel feed funguje pouze mezi náhodnými lidmi používající aplikaci. Není zde možnost přidávání přátel a vzájemná motivace mezi kamarády. Mezi další nevýhody patří nemožnost odkliknutí splnění úkolu přímo z notifikace, odložení notifikace na později a noční notifikace neobsahující žádné údaje – jedná se pouze o proklik do aplikace. [3]



Obrázek 1.1 Ukázka aplikace Dreamfora

1.2.2 Reach it: Goals, Habit Tracker

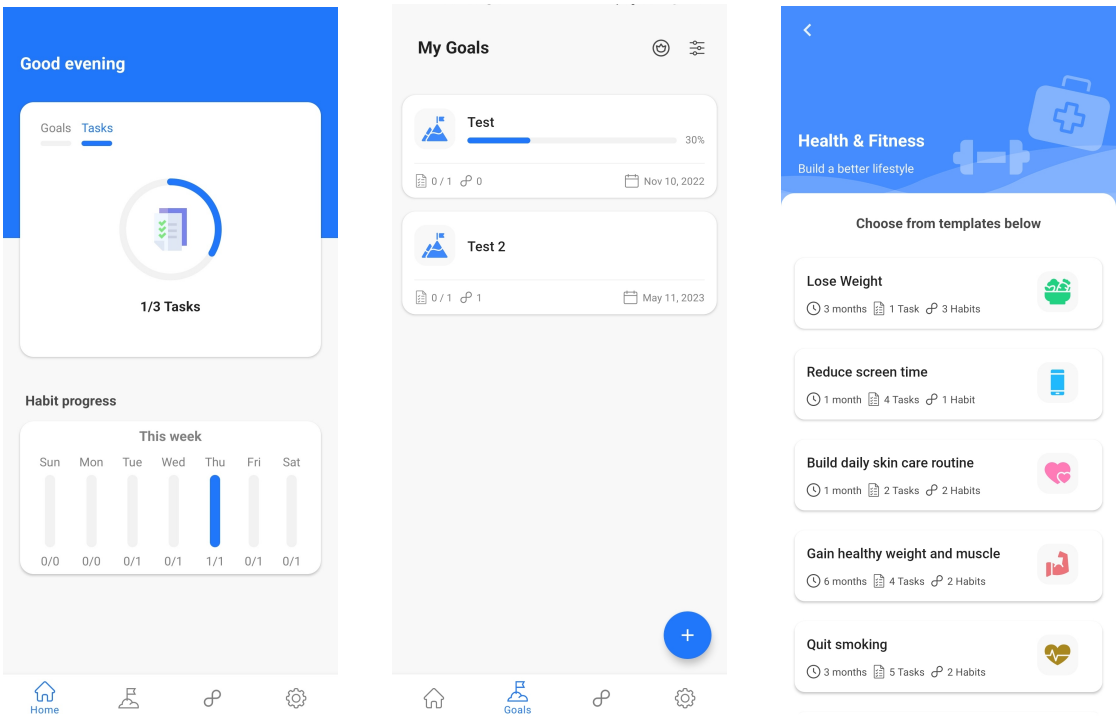
Počet stažení: více než 50 000
Hodnocení: není veřejné

Podobně jako v aplikaci Dreamfora i zde lze najít stejný koncept vytváření cílů s jednorázovými i opakujícími se úkoly (zvyky). Liší se akorát způsobem definování zvyků, kde v aplikaci Reach It jsou opakující se úkoly odděleny na samostatnou záložku. Uživatel si nejprve nastaví své osobní cíle a poté na této oddělené kartě definují své opakující se úkoly, které chtějí sledovat a plnit. Tato oddělená záložka pro opakující se úkoly umožňuje uživatelům snadno sledovat svůj pokrok a zaměřit se na své zvyky zvlášť od svých osobních cílů. Aplikace rovněž poskytuje předpřipravené cíle, které uživateli slouží jako inspirace při vytváření plánu seberozvoje.

Aplikace je rozdělena do 3 hlavních částí:

- domovská stránka, která obsahuje velmi základní statistiky – celkový počet splněných cílů, úkolů a počet splněných zvyků v aktuálním týdnu (viz obrázek 1.2),
- záložka cíle, kde uživatel nalezne své zadefinované osobní cíle s přiřazenými úkoly,
- záložka zvyky na které se nacházejí opakované úkoly uživatele s časovou osou jejich plnění. Volitelně lze tyto úkoly přiřadit k příslušným cílům. Navíc lze na této kartě vidět i jednorázové úkoly a jejich stav splnění pro daný den.

Aplikace využívá velmi minimalistický design s důrazem na jednoduchost a intuitivnost uživatelského rozhraní. Tmavý režim aplikace bohužel patří mezi placené funkce. Další nevýhodou aplikace je nemožnost přidávání přátel a absence kanálu příspěvků přátel či ostatních uživatelů což značně omezuje schopnost aplikace motivovat uživatele k dosažení jeho cílů. [4]



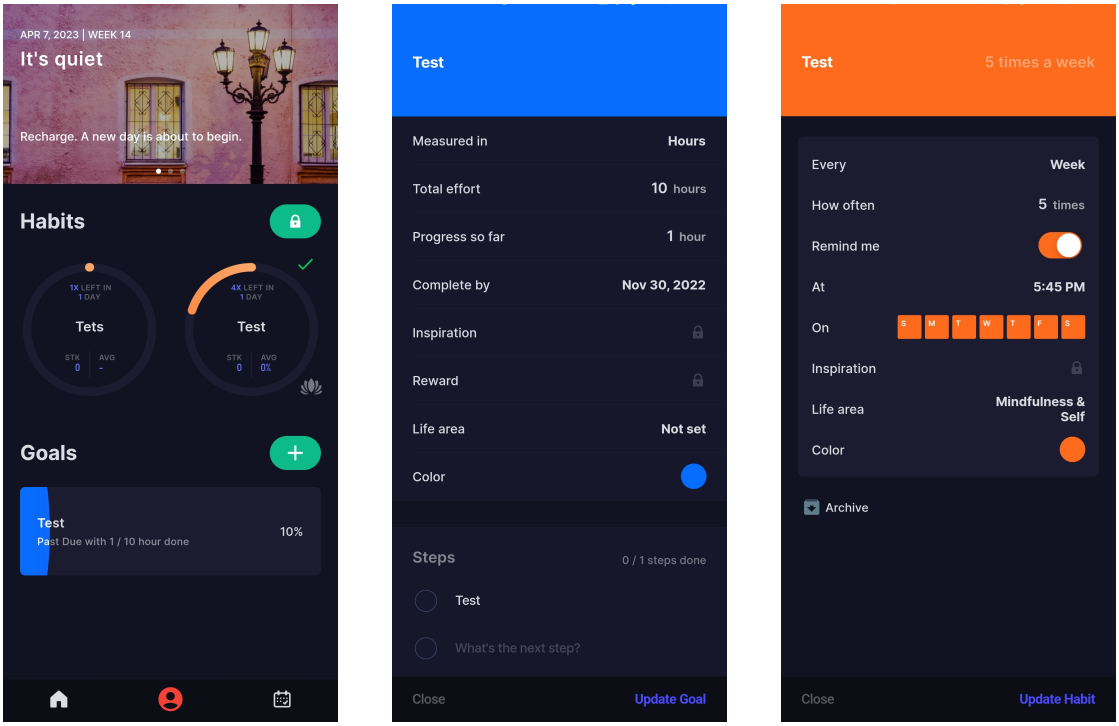
Obrázek 1.2 Ukázka aplikace Reach it

1.2.3 Higher Goals: Inspiring Habits

Počet stažení: více než 100 000
Hodnocení: 4,0 / 5

Higher Goals je další aplikace pro sledování osobních cílů. Na domovské stránce aplikace uživatel vidí seznam svých osobních cílů spolu s naplánovanými zvyky v podobě grafů zobrazující stav počtu plnění úkolů k aktuálnímu dni a statistikami plnění daného úkolu. Další záložka profilu umožňuje uživateli nastavit preference aplikace dle svých potřeb, například vlastní myšlenky (nazývané „Mindsets“) zobrazované na domovské obrazovce. Kromě toho mohou uživatelé na záložce profilu organizovat své cíle a zvyky. Poslední záložka kalendář zobrazuje uživateli graficky jeho plnění úkolů přímo v kalendáři.

Mezi nevýhody aplikace patří omezenost neplacené verze, která umožňuje uživatelům vytvořit pouze 2 osobní cíle a 2 zvyky. Navíc historie plnění úkolů a kalendář je také pouze placená funkce. Design a uživatelské rozhraní nepůsobí na první pohled příliš intuitivně a chybí zde možnost přidávání přátel a kanálu příspěvků. [5]



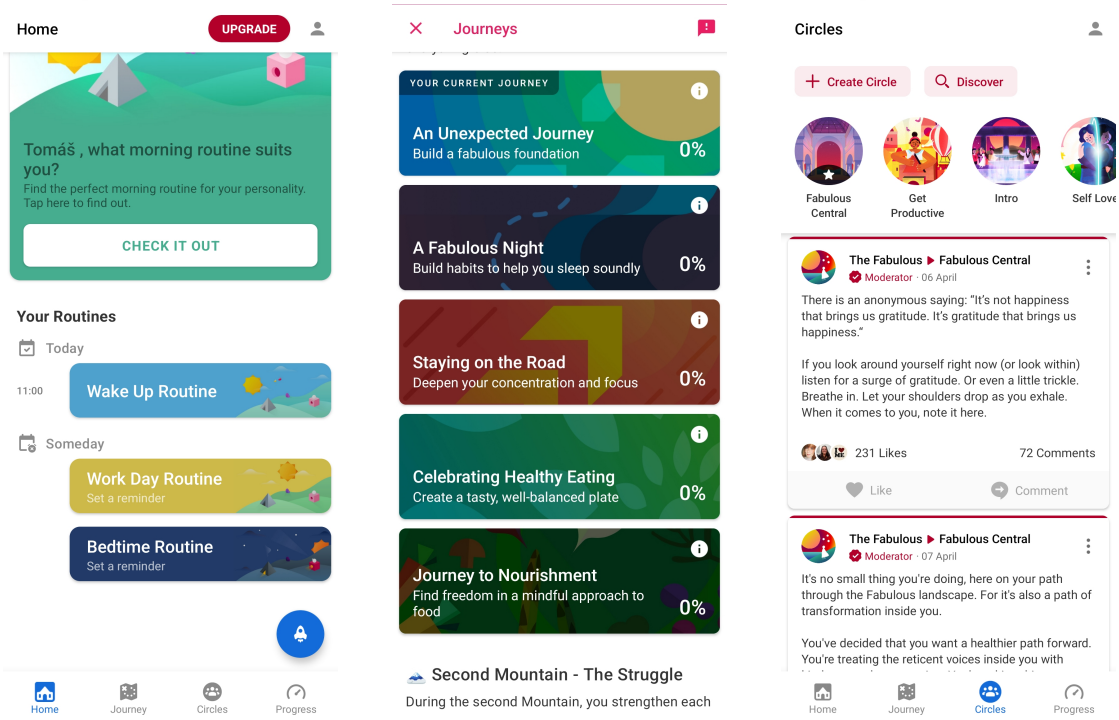
Obrázek 1.3 Ukázka aplikace Higher Goals

1.2.4 Fabulous Daily Routine Planner

Počet stažení: více než 10 000 000
Hodnocení: 4,4 / 5

Aplikace Fabulous je aplikace sloužící k seberozvoji s důrazem na předpřipravené plány a personalizaci. Již po prvním otevření aplikace je uživatel osloven sérií otázek, na základě kterých je vytvořen jeho vlastní personalizovaný plán. Další předpřipravené plány poté lze nalézt v „Journeys“, kde jsou jednotlivé plány rozděleny do kategorií. V neplacené verzi aplikace je ale uživatel

omezen na jeden plán, který mu aplikace zvolila. Dále aplikace obsahuje kanál příspěvků (tzv. „Circles“), kde uživatelé můžou přidávat motivační příspěvky a tyto příspěvky lajkovat a komentovat. Aplikaci chybí možnost přidávání přátel. Další nevýhodou aplikace je její nepříliš snadné ovládání, které může být pro uživatele zpočátku matoucí. [6]



Obrázek 1.4 Ukázka aplikace Fabulous

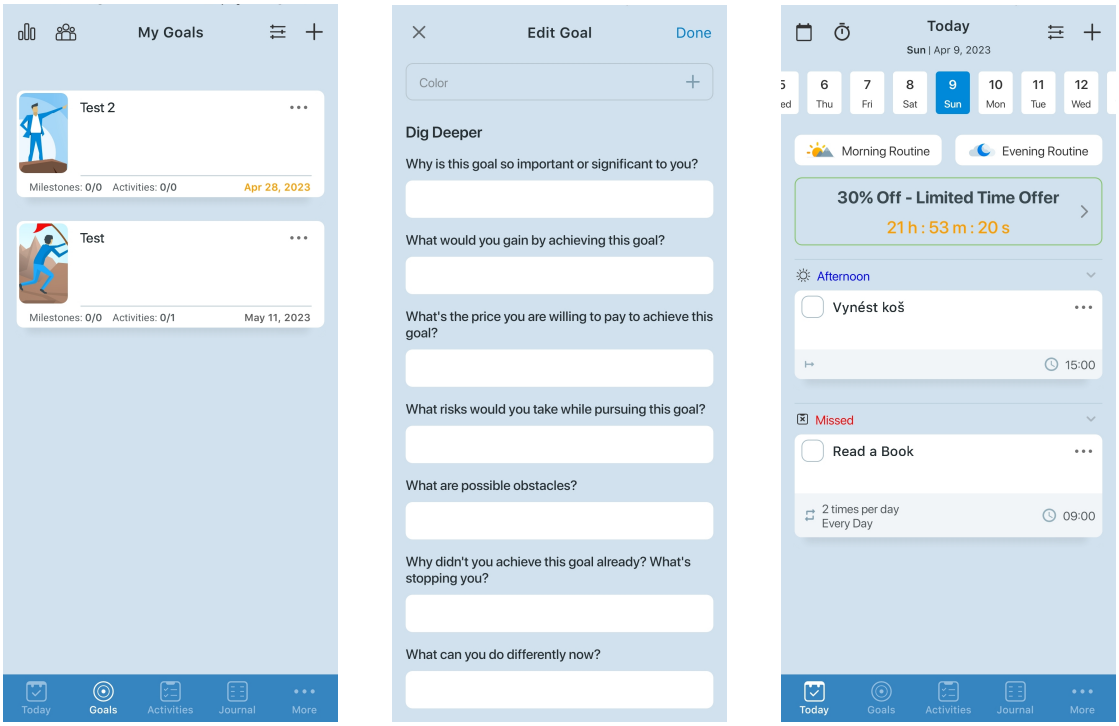
1.2.5 Goal Setting Tracker Planner

Počet stažení: více než 100 000
Hodnocení: není veřejné

Aplikace Goal Setting Tracker Planner je pokročilý plánovač osobních cílů. Při vytváření cíle je uživatel vyzván ke zvolení kategorie. Zde uživatel zvolí předpřipravený úkol nebo vytvoří svůj vlastní. Unikátní vlastností této aplikace je možnost zodpovědět sadu otázek pro každý osobní cíl související s důvody, proč je pro uživatele daný cíl důležitý, co je ochotný pro něj obětovat a investovat pro jeho dokončení, jaké překážky uživatele čekají aj. Tyto otázky pomůžou uživateli k bližšímu pochopení, proč je pro ně osobní cíl důležitý a motivují jej ke splnění.

V aplikaci Goals Setting podobně jako v aplikaci Reach It se nejdříve definuje cíl. Na záložce „Activities“ poté lze zadefinovat jednorázové úkoly a rekurentní úkoly a přiřadit je k cílům. Na záložce „Today“ uživatel vidí splněné i nesplněné aktivity pro každý den. V aplikaci je možné zapnout synchronizaci s Google kalendářem.

Aplikace je hezky zpracována s přehledným uživatelským rozhraním a velkým množstvím funkcionalit. Nevýhodou aplikace je omezená neplacená verze, kde uživatel může vytvořit pouze 3 cíle, 3 rutiny (rekurentní úkoly) a 50 aktivit (jednorázových úkolů). Aplikace navíc nedisponuje žádným kanálem příspěvků a možností přidávání přátel. [7]



Obrázek 1.5 Ukázka aplikace Goal Setting Tracker Planner

1.3 Porovnání existujících řešení

V kapitole 1.1 jsou řečeny ideální požadavky takové aplikace. V tabulce 1.1 lze nalézt porovnání existujících řešení vůči právě zmíněným funkcím.

Tabulka 1.1 Porovnání existujících řešení

	Sdílení	Rozdělení cílů	Intuitivní design	Účet v aplikaci	Zdarma
Dreamfora	Ne	Ano	Ano	Ano	Ano
Reach It	Ne	Ano	Ano	Ne	Ne
Higher Goals	Ne	Ano	Ne	Ano	Ne
Fabulous	Ne	Ne	Ne	Ano	Ne
Goal Setting T. P.	Ne	Ano	Ano	Ano	Ne

1.4 Požadavky aplikace

Tato podkapitola se zabývá specifikací požadovaných funkcí a vlastností navrhované aplikace na základě analýzy již existujících řešení. Aplikace se bude skládat z krátkodobých a dlouhodobých osobních cílů. Krátkodobé cíle budou obsahovat jednorázové i opakované úkoly, kterým lze nastavit maximální počet opakování. Dlouhodobé cíle jsou cíle, které nemají žádný termín splnění a jejich úkoly nemají daný žádný počet splnění do jejich dokončení. Skládají se pouze opakujících se úkolů.

1.4.1 Funkční požadavky

F1: Autentizace Před používáním aplikace si uživatel musí založit účet nebo se přihlásit do již existujícího účtu. Uživatel dostane na výběr mezi přihlášením nebo registrací pomocí kombinací emailové adresy a hesla, účtem Facebook nebo účtem Google.

F2: Seznam krátkodobých cílů Uživatel si může zobrazit seznam svých cílů. Tyto cíle může filtrovat podle jejich parametrů. Uživatel může vytvořit nový cíl nebo editovat a mazat stávající.

F3: Seznam dlouhodobých cílů Uživatel si může zobrazit seznam svých dlouhodobých cílů a může jej filtrovat podle jejich parametrů. Cíle lze vytvářet, editovat a mazat.

F4: Seznam úkolů Úkoly jsou rozdělené na rekurentní a jednorázové. Jednorázové úkoly lze přiřadit pouze krátkodobým cílům. Každý úkol musí být přiřazen k právě jednomu cíli. Cíle lze vytvářet, editovat a mazat.

F5: Statistiky Uživatel si může zobrazit své statistiky, jak v posledních dnech plnil své úkoly.

F6: Přátelé Uživatel si může zobrazit seznam svých přátel v aplikaci, může odeslat žádost o přátelství a žádosti přijímat nebo zamítat.

F7: Feed Uživatel si může zobrazit příspěvky ostatních uživatelů, které splnili nějakou aktivitu či cíl. Tyto příspěvky lze lajkovat a lze si u příspěvku zobrazit seznam uživatelů, kteří daný příspěvek lajknuli. Uživatel si může zobrazit i vlastní příspěvky.

F8: Notifikace Pro každý úkol dostane uživatel notifikaci ke splnění daného úkolu v nastaveném čase upozornění úkolu. Uživatel může přímo z notifikace potvrdit, že daný úkol byl splněn nebo může notifikaci odložit na později.

1.4.2 Nefunkční požadavky

N1: Dostupnost aplikace Aplikace bude dostupná ke stažení z oficiálního obchodu Google Play.

N2: Podpora zařízení Aplikace bude kompatibilní s minimálně 98 % zařízeními s operačním systémem Android dostupných na světě.

N3: Jazyk aplikace Aplikace bude dostupná v anglickém jazyce.

N4: Prostředí aplikace Aplikace bude mít 2 prostředí a to produkční, které bude nasazeno do obchodu Google Play a testovací, které bude využíváno k internímu testování a bude nasazováno do „Firebase App Distribution“

Kapitola 2
Návrh

2.1 Případy užití

Případy užití se v softwarovém inženýrství používají pro detailnější specifikování funkčních požadavků. Slouží k zachycení interakcí mezi uživatelem a systémem. Součástí každého případu užití musí být definován aktér (účastník), který daný požadavek využívá. Požadavky lze sepsat na různých úrovních detailu. Stručný popis se používá pro počáteční fáze projektů a obsahuje pouze krátký popis, co daný případ aktérovi umožňuje. V detailním popisu lze poté zaznamenat i scénáře s jednotlivými kroky interakce aktéra se systémem včetně podmínek, které musí nastat před spuštěním a po dokončení případu. Samotné scénáře lze rozdělit na hlavní a vedlejší scénáře, kterými lze dosáhnout stejného výsledku. Kromě textové části lze k modelu přidat i diagram. „*Diagramy případů užití slouží pouze jako doplňková informace, která umožňuje snadno zjistit, který z aktérů/účastníků používá zobrazený případ užití. Pouhé diagramy však v žádném případě nelze považovat za kompletní model případů užití.*“ [8]

UC1: Přihlášení/registrace emailem Na obrazovce přihlášení resp. registrace má uživatel možnost zadat email a heslo a přihlásit se resp. zaregistrovat se.

UC2: Přihlášení/registrace účtem Google Na obrazovce přihlášení resp. registrace se uživatel může autentizovat pomocí účtu Google jedním kliknutím na tlačítko.

UC3: Příhlášení/registrace účtem Facebook Na obrazovce přihlášení resp. registrace se uživatel může autentizovat pomocí účtu Facebook jedním kliknutím na tlačítko.

UC4: Seznam úkolů ke splnění Kliknutím na domovskou obrazovku v navigační liště uživatel vidí seznam nesplněných úkolů pro aktuální den.

UC5: Zobrazení pokroku Kliknutím na domovskou obrazovku v navigační liště uživatel vidí statistiku plnění cílů.

UC6: Seznam cílů Kliknutím na záložku cíle v navigační liště uživatel vidí svůj seznam všech krátkodobých cílů.

UC7: Vytvoření cíle Kliknutím na tlačítko se symbolem „+“ na obrazovce cílů uživatel přejde na obrazovku vytvoření cíle. Po zadání údajů kliknutím na „✓“ uživatel dokončí vytvoření cíle.

UC8: Editace cíle Kliknutím na cíl na obrazovce cílů uživatel přejde na obrazovku editace cíle. Po požadovaných úpravách kliknutím na „✓“ uživatel uloží změny v cíli.

UC9: Filtrování cílů Na záložce cílů může uživatel kliknutím na tlačítko filtrů zobrazit okno s nastavením filtrováním cílů podle jeho parametrů. Po potvrzení se zobrazí cíle dle požadovaných parametrů.

UC10: Smazání cíle Na obrazovce editace cíle kliknutím na ikonu smazání se zobrazí dialogové okno s potvrzením smazání. Uživatel má možnost potvrdit nebo zrušit smazání. Pokud uživatel potvrdí smazání, cíl je trvale odstraněn.

UC11: Seznam dlouhodobých cílů Kliknutím na záložku dlouhodobých cílů v navigační liště uživatel vidí svůj seznam všech dlouhodobých cílů.

UC12: Vytvoření dlouhodobého cíle Kliknutím na tlačítko se symbolem „+“ na obrazovce dlouhodobých cílů uživatel přejde na obrazovku vytvoření cíle. Po zadání údajů kliknutím na „✓“ uživatel dokončí vytvoření dlouhodobého cíle.

UC13: Editace dlouhodobého cíle Kliknutím na dlouhodobých cíl na obrazovce cílů uživatel přejde na obrazovku editace cíle. Po požadovaných úpravách kliknutím na „✓“ uživatel uloží změny v dlouhodobém cíli.

UC14: Filtrování dlouhodobých cílů Na záložce dlouhodobých cílů může uživatel filtrovat dlouhodobé cíle podle jeho parametrů.

UC15: Smazání dlouhodobého cíle Na obrazovce editace dlouhodobého cíle kliknutím na ikonu smazání se zobrazí dialogové okno s potvrzením smazání. Uživatel má možnost potvrdit nebo zrušit smazání. Pokud uživatel potvrdí smazání, cíl je trvale odstraněn.

UC16: Vytvoření rekurentního úkolu Na obrazovce editace cíle může uživatel vytvořit nový rekurentní úkol. Po zadání požadovaných parametrů stisknutím tlačítla „✓“ uživatel dokončí vytvoření úkolu.

UC17: Vytvoření jednorázového úkolu Na obrazovce editace krátkodobého cíle může uživatel vytvořit nový jednorázový úkol. Po zadání požadovaných parametrů stisknutím tlačítla „✓“ uživatel dokončí vytvoření úkolu.

UC18: Editace úkolu Na obrazovce editace cíle kliknutím na daný úkol jej může uživatel editovat. Po dokončení požadovaných změn stisknutím tlačítka „✓“ uživatel uloží změny v úkolu.

UC19: Smazání úkolu Na obrazovce editace úkolu stisknutím ikonky smazání se zobrazí dialogové okno s potvrzením smazání. Uživatel má možnost potvrdit nebo zrušit smazání. Pokud uživatel potvrdí smazání, úkol je trvale odstraněn.

UC20: Žádost o přátelství Na obrazovce profilu kliknutím na tlačítko přídání přítele se zobrazí okno pro zadání emailové adresy přítele. Po zadání emailové adresy a potvrzení je žádost odeslána.

UC21: Potvrzení/zamítnutí žádosti o přátelství Kliknutím na záložku profil v navigační liště se uživateli na obrazovce zobrazí seznam příchozích žádostí o přátelství.

UC22: Zobrazení příspěvků přátel Kliknutím na záložku feed v navigační liště se uživateli zobrazí seznam příspěvků přátel i svých příspěvků (úspěchů).

UC23: Lajknutí příspěvku Na obrazovce feed kliknutím na symbol srdíčka může uživatel lajknout daný příspěvek.

UC24: Notifikace na úkol V nastaveném čase pro daný úkol uživateli přijde notifikace ke splnění úkolu. Notifikace obsahuje tlačítko, kterým uživatel potvrdí splnění úkolu.

UC25: Odložení notifikace V notifikaci ke splnění úkolu je tlačítko, kterým uživatel odloží daný úkol na později.

Vyjma případu užití UC24 platí, že aktérem je uživatel aplikace. Pro UC24 je aktérem časovač operačního systému, který v zadaný čas zobrazí notifikaci. Protože většina případu užití má stejného aktéra, není potřeba vytvářet diagram, který by případům užití nepřidal žádnou další hodnotu.

2.1.1 Pokrytí požadavků případy užití

Protože případy užití detailněji specifikují funkční požadavky, měli by pokrývat všechny zadefinované požadavky. V následující tabulce 2.1 lze vidět, že všechny funkční požadavky jsou pokryté případy užití.

Tabulka 2.1 Tabulka pokrytí požadavků případy užití

	F1	F2	F3	F4	F5	F6	F7	F8
UC1	✓							
UC2	✓							
UC3	✓							
UC4				✓				
UC5					✓			
UC6		✓						
UC7		✓						
UC8		✓						
UC9		✓						
UC10		✓						
UC11			✓					
UC12			✓					
UC13			✓					
UC14			✓					
UC15			✓					
UC16				✓				
UC17				✓				
UC18				✓				
UC19				✓				
UC20						✓		
UC21						✓		
UC22							✓	
UC23							✓	
UC24								✓
UC25								✓

2.2 Architektura

Jedním z důležitých aspektů návrhu softwaru je správné zvolení a používání architektonických vzorů. Architektonické vzory jsou sada pravidel pro řešení často se vyskytujících problémů v návrhu softwarové architektury. Použití architektonických vzorů pomáhá k dosažení

- modulárního a organizovaného kódu, což usnadňuje udržitelnost kódu a zjednodušuje práci vývojářům,

- oddělení zodpovědností kódu jako například obchodní logiku od uživatelského rozhraní,
- zvýšení testovatelnosti kódu. Oddělením zodpovědností kódu lze testovat jednotlivé části softwaru nezávisle na sobě. [9]

2.2.1 MVC

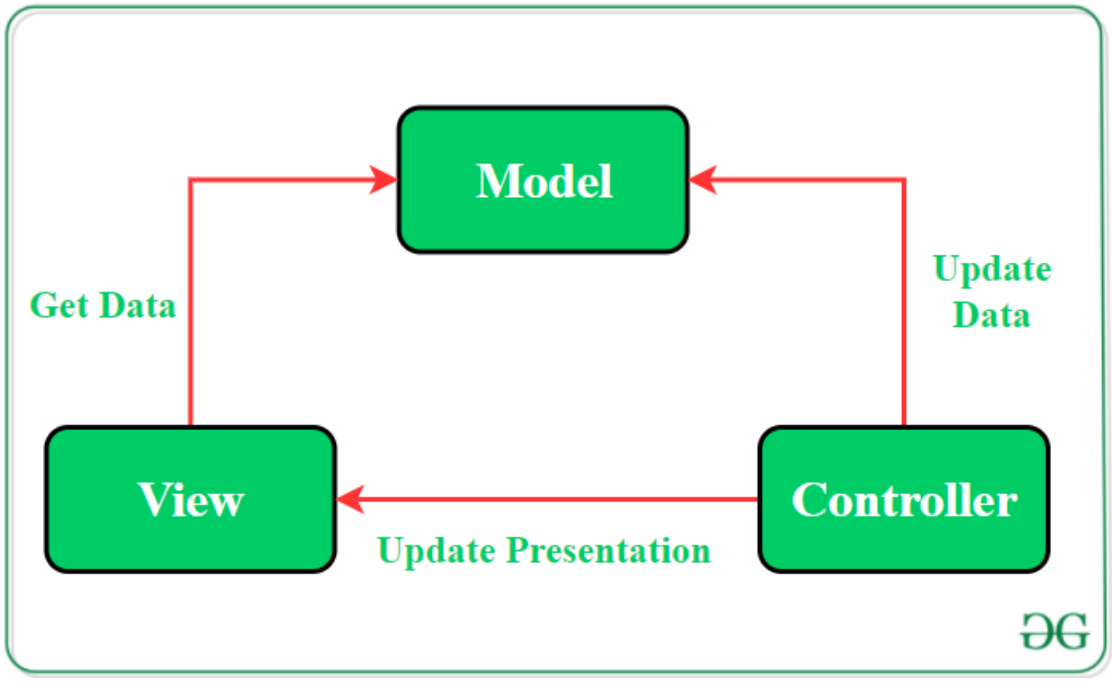
Model-View-Controller architektura se řadí mezi nejstarší a nejpoužívanější architektonické vzory. V MVC architektuře je aplikace rozdělena do tří vrstev a to:

Model Modelová vrstva se stará o uchovávání dat a jejich logiky a dále o komunikaci s perzistentní a síťovou vrstvou.

View View vrstva je zodpovědná za uživatelské rozhraní. Poskytuje vizualizaci modelové vrstvy a zajišťuje interakci aplikace s uživatelem.

Controller Controller vrstva zajišťuje propojení mezi modelovou a view vrstvou. Obsahuje veškerou obchodní logiku aplikace.

Nevýhodou MVC architektury je vysoká provázanost mezi vrstvami. To vede k horší testovatelnosti aplikace neboť je těžší izolovat jednotlivé komponenty a testovat je nezávisle na sobě. Další nevýhodou architektury je příliš velká zodpovědnost controlleru, který obsahuje, kromě obchodní logiky, komunikaci mezi modelovou a view vrstvou. To může způsobit, že controller vrstva obsahuje velké množství kódu, který se může stát nepřehledný. [10, 11]



Obrázek 2.1 Diagram architektonického vzoru MVC. Převzato z [11]

2.2.2 MVP

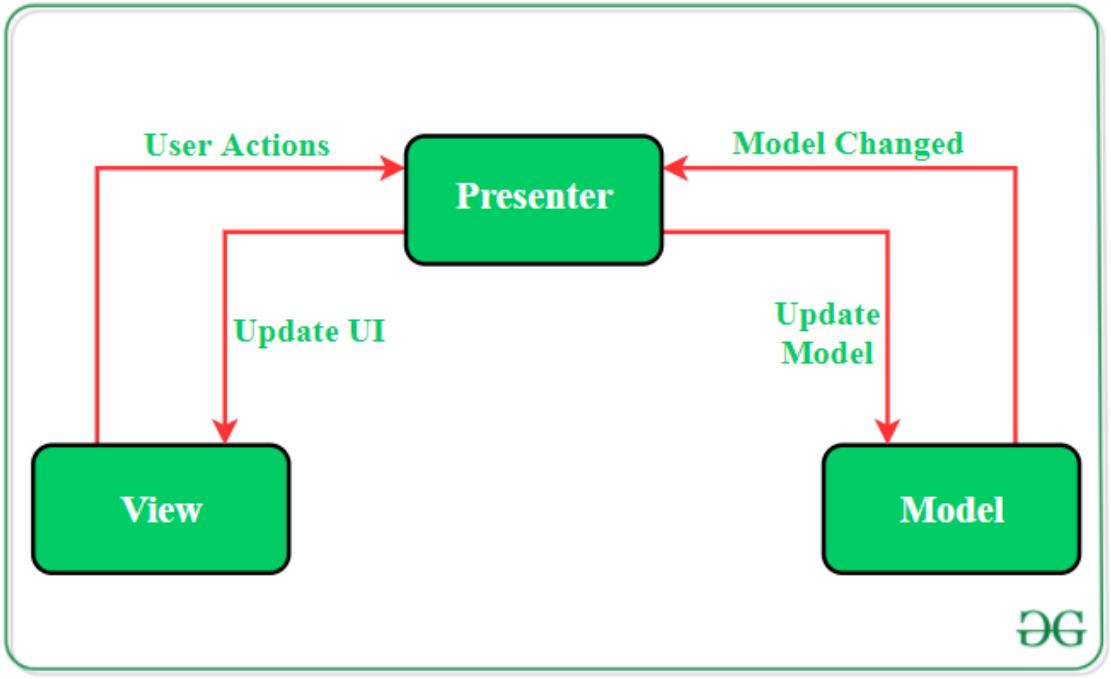
Některé problémy MVC architektury řeší architektura Model-View-Presenter. MVP architektura se také dělí na 3 vrstvy.

Model Stejně jako v MVC architektuře, modelová vrstva se stará o uchovávání dat a jejich logiky a o komunikaci s perzistetní a síťovou vrstvou.

View View vrstva poskytuje vizualizaci dat získanou z presenteru a zaznamenává uživatelské akce pro informování presenteru.

Presenter Presenter vrstva získává data z modelové vrstvy a aplikuje na ni logiku uživatelského rozhraní, spravuje stav view vrstvy a zpracovává uživatelské akce z view vrstvy.

V MVC controller vybírá a vrací nové view pro každou akci. V MVP všechny akce uživatele řeší presenter vrstva a view vrstva pouze notifikuje presenter o akci uživatele. Po provedení změn presenter vrstva aktualizuje view vrstvu skrz view rozhraní. Hlavní výhodou MVP oproti MVC architektuře je snížení provázanosti view vrstvy, což umožňuje snadnější testování. Nevýhodou MVP jsou automatické aktualizace uživatelského rozhraní, kde pro jakoukoliv změnu UI je potřeba podnět ze strany presenteru. [12, 13]



Obrázek 2.2 Diagram architektonického vzoru MVP. Převzato z [11]

2.2.3 MVVM

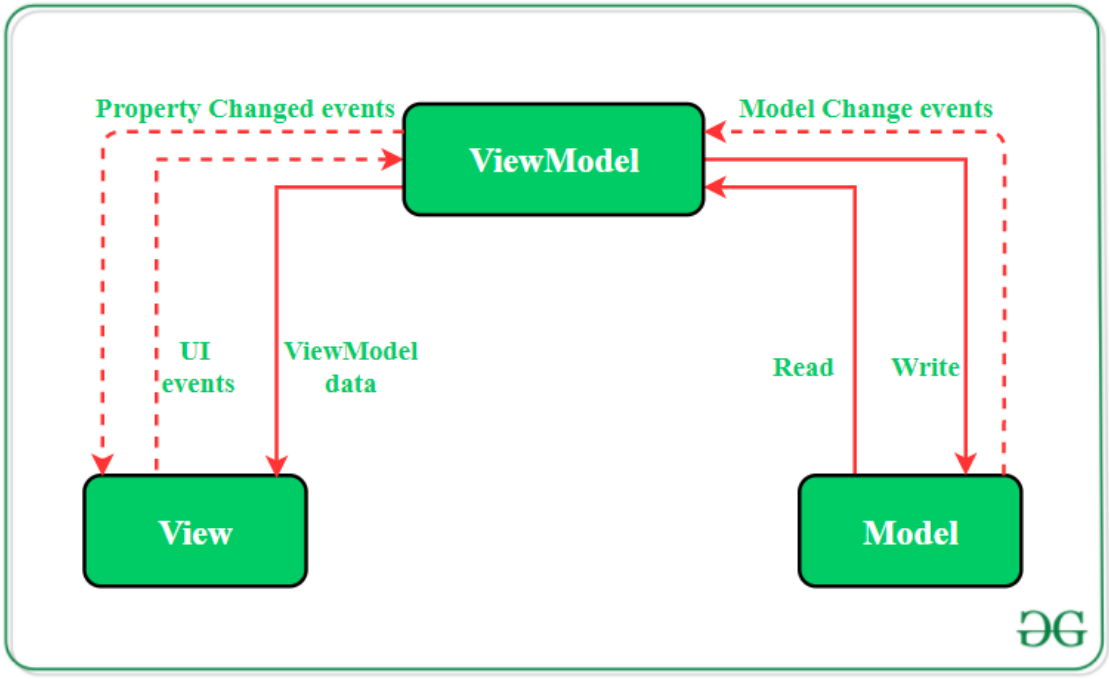
Model-View-ViewModel (MVVM) je architektonický vzor, který se stává stále více populární mezi vývojáři softwaru. Překovává většinu nevýhod MVP a MVC vzorů. MVVM architektura také rozděluje kód do 3 vrstev.

Model Modelová vrstva se stará o uchovávání dat v aplikaci.

View View vrstva se stará o vizualizaci dat a informování ViewModelu o uživatelských akcích. View vrstva pozoruje ViewModel a reaguje na změny stavu. V této vrstvě se nenachází žádná další aplikační logika.

ViewModel Propojuje modelovou vrstvu s view vrstvou. Komunikuje s modelovou vrstvou, ze které získá data, která poté poskytuje view vrstvě.

Protože ViewModel obsahuje logiku prezentace a view vrstva je pasivní a zodpovědná pouze za zobrazení dat, umožňuje tato architektura lepší modularitu a tedy přehlednější a jednodušeji testovatelný kód aplikace. [14, 11]

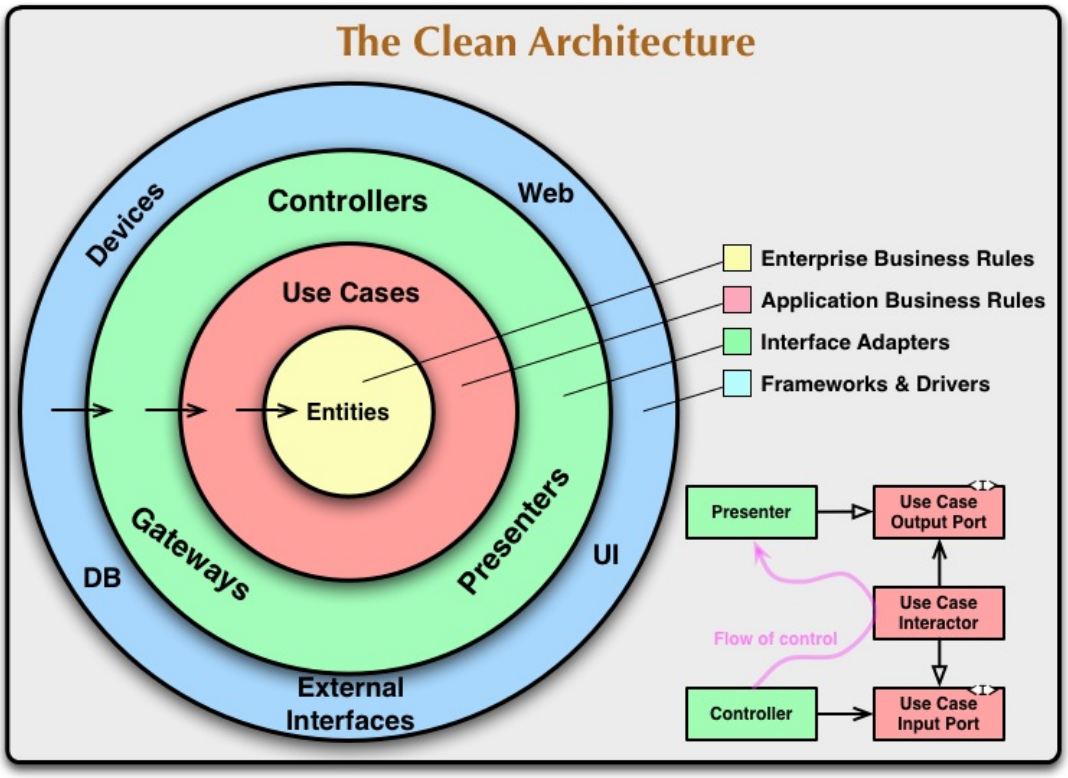


Obrázek 2.3 Diagram architektonického vzoru MVVM. Převzato z [11]

2.2.4 Clean architecture

Jedním z důležitých principů kvalitního kódu softwaru je tzv. oddělení zodpovědnosti (anglicky „Separate of Concerns“ – SoC) který se snaží o minimalizování provázanosti kódu. Clean Architecture je architektura popsaná v roce 2012 Robertem C. Martinem, která poskytuje SoC pomocí rozdělením softwaru do vrstev, které jsou vysoce soudržné a málo provázané mezi sebou. Tyto vrstvy jsou poté uspořádány do kruhů (viz obrázek 2.4). Hlavním principem této architektury je to, že závislosti tříd směřují pouze do středních vrstev zatímco vnitřní vrstvy jsou nezávislé na vnějších. Vnitřní vrstva obsahuje veškerou obchodní logiku specifickou pro danou doménu. Vnější vrstvy jsou zodpovědné za komunikaci s vnějším světem jako třeba UI aplikace, databáze, servery, API, aj. [15, 16]

Díky tomuto rozdělení jsou vnitřní vrstvy aplikace nezávislé na použitých technologiích a frameworcích a tedy jsou jednoduše testovatelné. Je také velmi jednoduché vyměnit implementace vnější vrstvy za jinou (například změna typu databáze, typ UI, API zdrojů aj.) bez zásahu do doménové vrstvy.



Obrázek 2.4 Znázornění Clean architecture. Převzato z [15]

2.2.5 Shrnutí

V této kapitole byly popsány vybrané architektonické vzory. Pro vývoj Android aplikací je doporučeno používat MVVM. Zatímco architektonické vzory MVC, MVP a MVVM se zaměřují na oddělení view vrstvy od bussiness (obchodní) logiky a datové vrstvy, clean architecture staví na principu oddělení zodpovědností mezi různými vrstvami aplikace a proto je možné clean architecture vzor použít v kombinaci s MVVM. Pro implementaci této práce byly vybrány MVVM a clean architecture vzory.

2.3 Návrh uživatelského rozhraní

Návrh uživatelského rozhraní je nedílnou součástí návrhu každé aplikace. Návrh slouží jako předloha ke vzhledu aplikace pro následnou implementaci. Uživatelské rozhraní by mělo být navrženo tak, aby bylo snadno použitelné a intuitivní pro uživatele. Pro návrh UI aplikace byl použit nástroj Miro [17].

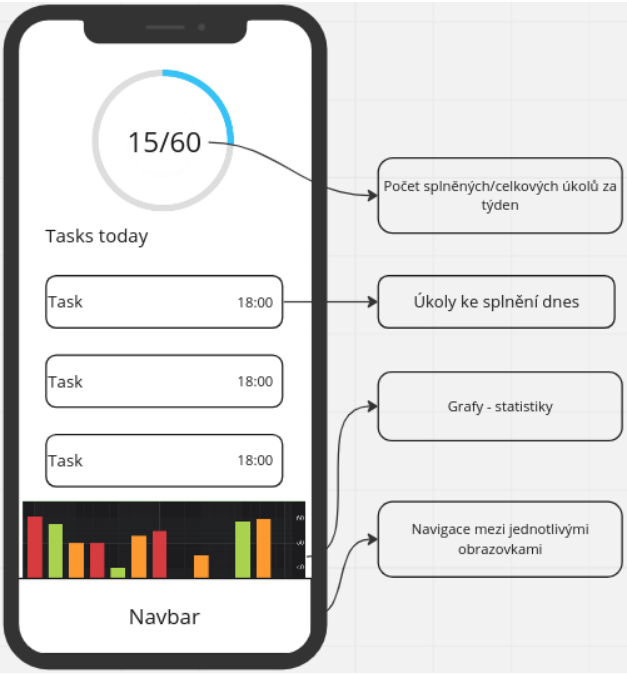
Aplikace bude rozdělena na následující obrazovky:

- domovská obrazovka,
- seznam krátkodobých cílů,
- seznam dlouhodobých cílů,
- editace cíle,

- editace úkolu,
- obrazovka feedu,
- obrazovka profilu,
- přihlášení a registrace.

2.3.1 Domovská obrazovka

Na této obrazovce budou uživateli zobrazeny statistiky plnění jeho osobních cílů spolu s přehledným seznamem úkolů, který zbývá splnit aktuální den. Statistiky budou zobrazovat počet splněných úkolů za poslední týden. Uživatel bude mít možnost potvrdit splnění úkolu přímo na této obrazovce. Na obrázku 2.5 je znázorněn návrh domovské obrazovky.



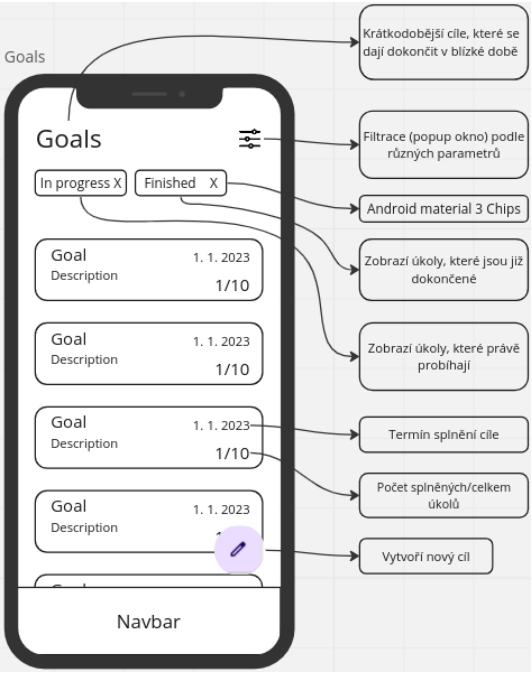
Obrázek 2.5 Návrh domovské obrazovky

2.3.2 Krátkodobé cíle

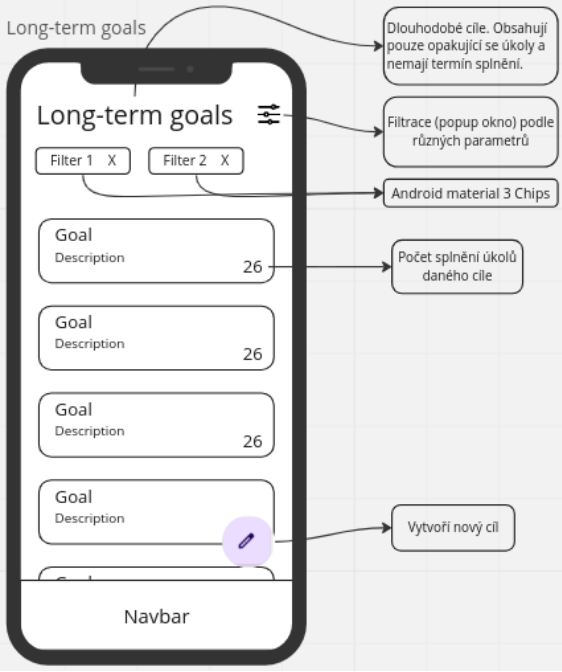
Obrazovka krátkodobých cílů bude uživateli zobrazovat seznam jeho osobních krátkodobých cílů. Uživatel bude mít možnost své cíle filtrovat pomocí vyskakovacího menu, dále bude mít možnost vytvořit nový cíl a editovat stávající kliknutím na cíl. Jednotlivé cíle zobrazují název cíle, popis cíle, termín jeho dokončení a počty splněných úkolů v rámci daného cíle. Na obrázku 2.6 lze najít návrh této obrazovky.

2.3.3 Dlouhodobé cíle

Na obrazovce dlouhodobých cílů si bude moci uživatel zobrazit seznam jeho dlouhodobých cílů. Podobně jako na obrazovce krátkodobých cílů zde může uživatel vytvořit nový cíl, filtrovat cíle a přejít na obrazovku editace cíle kliknutím na cíl. Na každém cíli lze nalézt jeho název, popis a počet splněných úkolů. Návrh obrazovky je ilustrován na obrázku 2.7.



Obrázek 2.6 Návrh obrazovky krátkodobých cílů



Obrázek 2.7 Návrh obrazovky dlouhodobých cílů

2.3.4 Editace cíle

Po kliknutí na daný cíl se uživateli zobrazí obrazovka editace cíle. Zde bude moci uživatel změnit jméno cíle, popis cíle a v případě krátkodobého cíle i jeho termín splnění. Kliknutím na ikonku popelnice může uživatel smazat editovaný cíl. Pod informacemi cíle uživatel nalezne seznam úkolů pro daný cíl. Kliknutím na úkol přejde uživatel na obrazovku editace úkolu. Kliknutím na tlačítko lze vytvořit nový úkol. Pro krátkodobé cíle zde bude na výběr mezi rekurentními úkoly a jednorázovými. Na obrázku 2.8 je ukázka návrhu této obrazovky.

Obrázek 2.8 Návrh obrazovky editace cíle

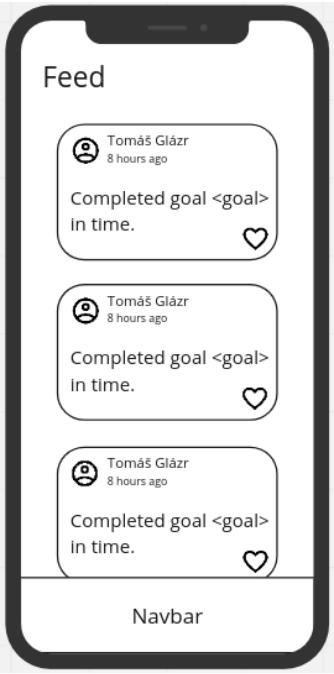
2.3.5 Editace úkolu

Na obrazovce editace úkolu může uživatel změnit jméno úkolu. V případě opakujícího se úkolu lze změnit čas opakování, dny v týdnu a počet opakování do splnění úkolu. Pro jednorázové úkoly lze nastavit pouze datum a čas připomenutí úkolu. Na obrázku 2.9 lze najít návrh této obrazovky.

2.3.6 Feed

Obrazovka Feedu zobrazí uživateli úspěchy přátel a vlastní úspěchy. Uživatel bude moci příspěvky svých přátel lajkovat kliknutím na ikonu srdce nebo v případě vlastního příspěvku zobrazit si seznam lidí, kteří příspěvek lajknuli. Návrh obrazovky lze nalézt na obrázku 2.10.

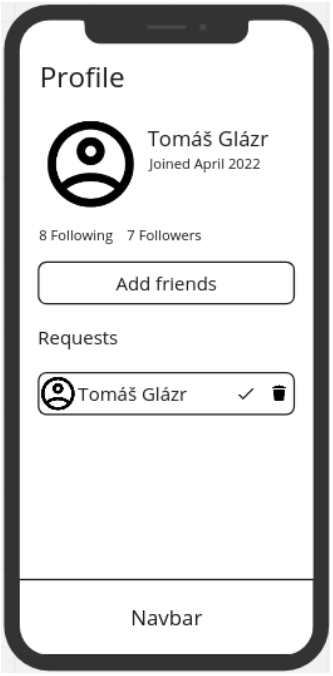
Obrázek 2.9 Návrh obrazovky editace úkolu



Obrázek 2.10 Návrh obrazovky feed

2.3.7 Profil

Obrazovka profilu nabídne uživateli možnost zobrazit si přátele, kteří jej sledují a možnost přidat si nové přátele pomocí emailové adresy. Na této obrazovce se také zobrazí uživatelovi příchozí žádosti o přátelství, které může uživatel přijmout nebo zamítnout. Návrh obrazovky se nachází na obrázku 2.11.



Obrázek 2.11 Návrh obrazovky profilu

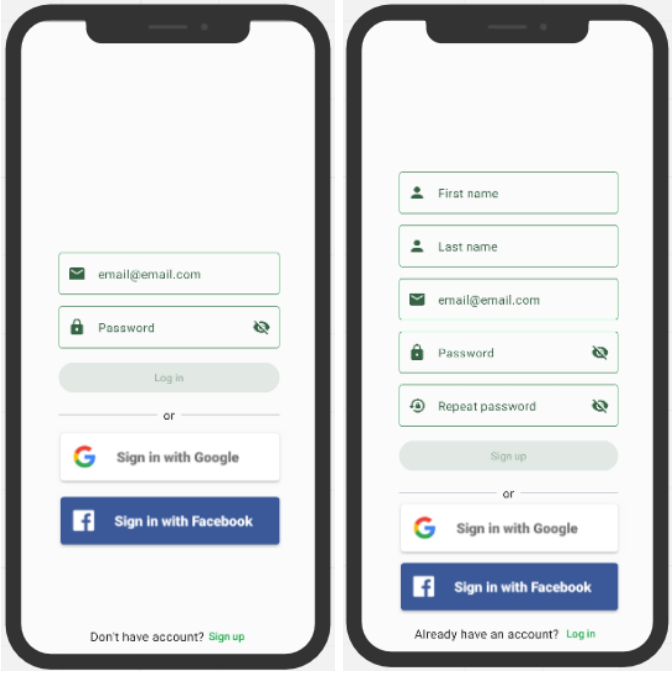
2.3.8 Přihlášení a registrace

Na obrazovce přihlášení resp. registrace uživatel bude moci zadat email a heslo pro přihlášení, V případě registrace zde budou také pole na jméno, příjmení a zopakování hesla. Uživatel zde bude mít možnost se přihlásit resp. zaregistrovat pomocí google účtu nebo facebook účtu. Návrh obrazovky lze nalézt na obrázku 2.12.

2.4 Cloudové služby

Pro usnadnění vývoje aplikace je možné použít existující cloudová řešení, které umožní aplikaci ukládat data na straně serveru, autentizovat uživatele, analyzovat chování aplikace a další nástroje. V současné době je na trhu mnoho různých platforem, které mohou být využity pro vývoj aplikace pro Android. Jednou z možností je použití Firebase, který je oblíbenou volbou mezi vývojáři pro jeho mnohostrannost a snadnou integraci do aplikace.

Firebase je platforma od společnosti Google, která nabízí celou řadu služeb pro vývoj mobilních aplikací, včetně autentizace uživatele, 2 typů databází, hostingu, cloudového úložiště, analýzy aplikace a mnoha dalších. Firebase je oblíbenou volbou mezi Android vývojáři díky své jednoduchosti a rychlosti, s nimiž lze tyto služby integrovat do aplikace. Firebase také nabízí širokou škálu nástrojů pro testování a ověřování výkonu aplikace.



Obrázek 2.12 Návrh obrazovky přihlášení a registrace

Další možností pro vývoj aplikace pro Android je využití služeb od Amazon Web Services (AWS), které nabízejí podobné funkcionality jako Firebase jako například autentizace uživatele, databáze a hosting. AWS je také oblíbenou volbou mezi vývojáři zejména pro svůj rozsáhlý výběr služeb a jeho škálovatelnost.

Další alternativou je použití služeb od Microsoft Azure, které také poskytují podobné funkcionality jako Firebase a AWS. V porovnání s platformou Firebase nebo AWS je však Azure méně rozšířený a má menší komunitu.

Při rozhodování o tom, kterou službu použít, je třeba zohlednit několik faktorů, jako jsou náklady, škálovatelnost, snadnost použití a integrace do aplikace. Firebase se obvykle preferuje pro svou snadnou integraci a nízké náklady pro malé aplikace. AWS a Azure mohou být ale lepší volbou pro větší projekty, kde je třeba vyšší škálovatelnost a robustnost.

Pro implementaci aplikace vyvíjené v této práci byla zvolena platforma Firebase pro svoji jednoduchou integraci a nulovým nákladům pro menší projekt.

2.5 Návrh databáze

Pro uchovávání dat uživatelů je nutné použít perzistentní úložiště. Protože aplikace bude sdílet data mezi jednotlivými uživateli je potřeba využít online úložiště, které zajišťuje spolehlivost a zabraňuje ztrátě dat uživatele v případě poškození či změně zařízení uživatele.

Pro strukturování dat v online úložišti je nejvhodnější použít databázové systémy, které umožňují efektivní správu a rychlé vyhledávání dat. Nejrozšířenějšími typy databází jsou relační a NoSQL databáze [18].

2.5.1 Relační databáze

Relační databáze jsou úložiště, kde informace jsou uloženy v tabulkách. Mezi daty napříč různými tabulkami je vztah, kterému se říká relace. Informace v tabulkách jsou uložené pomocí sloupců

tabulky. Zpravidla každá tabulka má jeden nebo více sloupců, které tvoří primární klíč (jednoznačný unikátní identifikátor záznamu v tabulce). Nejčastější způsob interakce s relačními databázemi je pomocí jazyka SQL (Structured Query Language), který umožňuje vývojářům nad daty provádět CRUD (zkratka pro Create, Read, Update, Delete) operace. [19]

Relační databáze mají mnoho výhod. Jednou z klíčových výhod je, že dodržují ACID (Atomicity, Consistency, Isolation, Durability) princip, který zajišťuje konzistentnost a spolehlivost dat. Atomicita zajišťuje, že transakce jsou buď provedeny celé nebo vůbec. Konzistence zaručuje, že data jsou vždy v platném stavu a odpovídají definovaným pravidlům. Izolace zajišťuje, že transakce jsou oddělené a navzájem neovlivňují. Trvanlivost znamená, že po dokončení transakce data zůstanou zachována a to i v případě výpadku systému. [20]

Hlavní nevýhodou relačních databází je jejich omezená škálovatelnost. Výkon těchto systémů je přímo závislý na komplexitě a množství uchovávaných dat. Tyto systémy jsou určené k běhu na jednom zařízení a proto, pokud výkon daného systému není dostatečný, musí se systému vylepšit hardwarové prostředky (tzv. vertikální škálování). Vertikální škálování je ale velmi drahé a lze provádět pouze do určité míry. Další nevýhodou relačních databází je jejich omezená flexibilita. Schéma těchto databází je pevně dané a jakékoliv pozdější změny vyžaduje úpravu všech existujících dat. [19]

2.5.2 NoSQL databáze

NoSQL databáze jsou databáze, který nevyužívají metodu ukládání dat do tabulek a sloupců. Jejich hlavní výhodou je jejich lepší flexibilita a škálovatelnost neboť se dají velmi snadno škálovat horizontálně (měnit počet databázových systému). NoSQL databáze se dělí na několik typů:

Dokumentové databáze Tyto databáze ukládají data ve formě dokumentů nejčastěji ve formátu JSON (JavaScript Object Notation). Hodnoty v dokumentu jsou primitivní hodnoty, pole nebo objekty, kterými můžou být i vnořené dokumenty. Dokumenty jsou považovány na sobě nezávislé což znamená, že mohou být distribuovány mezi více serverů.

Databáze typu klíč-hodnota Jedná se o velmi primitivní typ databáze, kde se každá hodnota ukládá podle zadaného unikátního klíče. Tento klíč poté slouží pro opětovné získání hodnoty. Hlavní nevýhodou tohoto typu databáze je nemožnost ukládat složité datové typy.

Grafové databáze Tento typ databáze ukládá data do uzlů. Tyto uzly jsou poté spojované hranami reprezentující relace mezi daty. Výhodou těchto databází je vysoká flexibilita dat a rychlé vyhledávání v datech. Jejich nevýhodou je velmi špatné dotazování dat, která nejsou propojena.

„Wide-column“ databáze Wide-column databáze nebo také sloupcové databáze se velmi podobají relačním databázím způsobem ukládání dat do tabulek a sloupců. Rozdílem však je nepovinnost dodržení formátování sloupců pro každý záznam tabulky. Jednotlivé sloupce dokonce můžou být uloženy napříč různými servery. Tento typ databází je také velmi flexibilní avšak dotazování v těchto databázích je značně pomalé neboť se data shlukují podle stejných sloupců oproti seskupování dat podle daného záznamu. [19]

2.5.3 Shrnutí

V této podkapitole byly popsány základní typy databázových systémů. Pro implementaci této práce byla vybrána dokumentová databáze pro svoji flexibilitu schématu a možnosti škálovatelnosti v cloudovém úložišti.

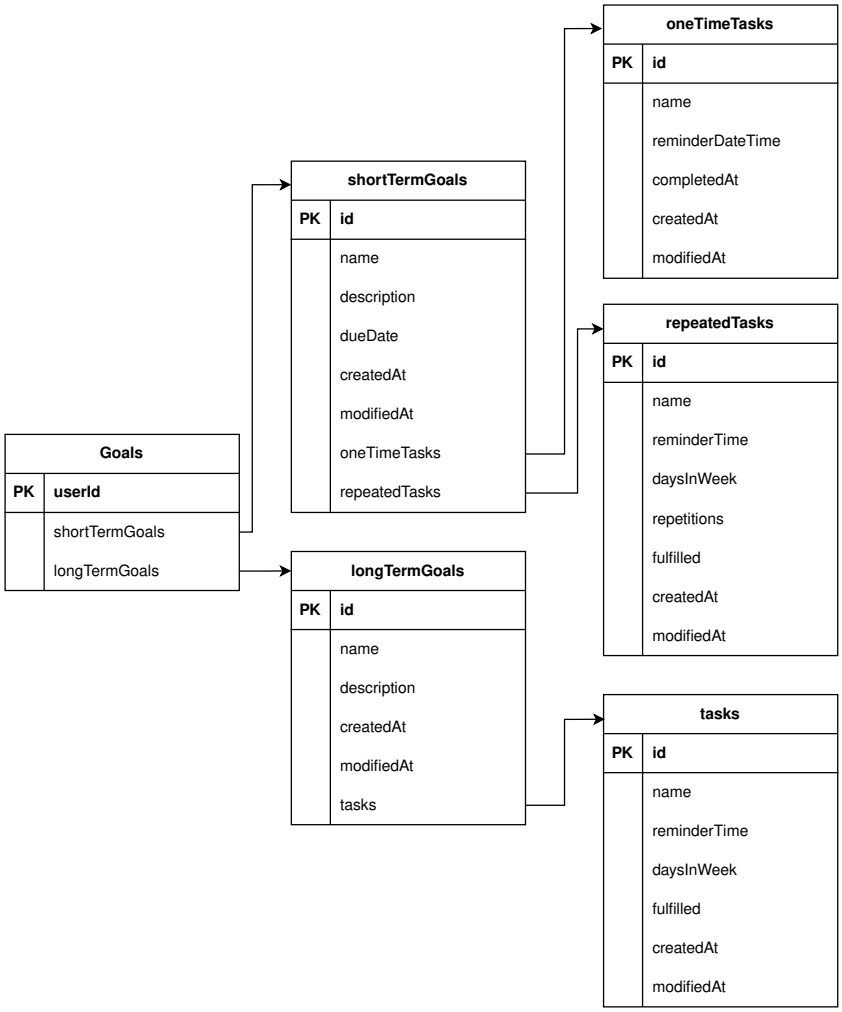
Pro implementaci práce bude použita databáze Firestore. Jedná se o NoSQL dokumentovou cloud databázi, která je součástí cloudových řešení Firebase od společnosti Google. Výhodou

Firestore databáze je, že nevyžaduje žádný middleware a lze jej napojit přímo na aplikaci. Firestore ukládá data v databázi jako dokumenty, které sdružuje do tzv. kolekcí. Dokumenty mohou obsahovat kromě primitivních hodnot a kolekcí také komplexní vnořené objekty a i podkolekce. Díky tomu jsou data ukládána v hierarchii a databáze je tak flexibilní. Mezi další výhody patří předpřipravené SDK přímo určené pro Android vývoj, což usnadňuje implementaci vývojářům, cachování dat pro offline přístup a také snadné škálování databáze. [21]

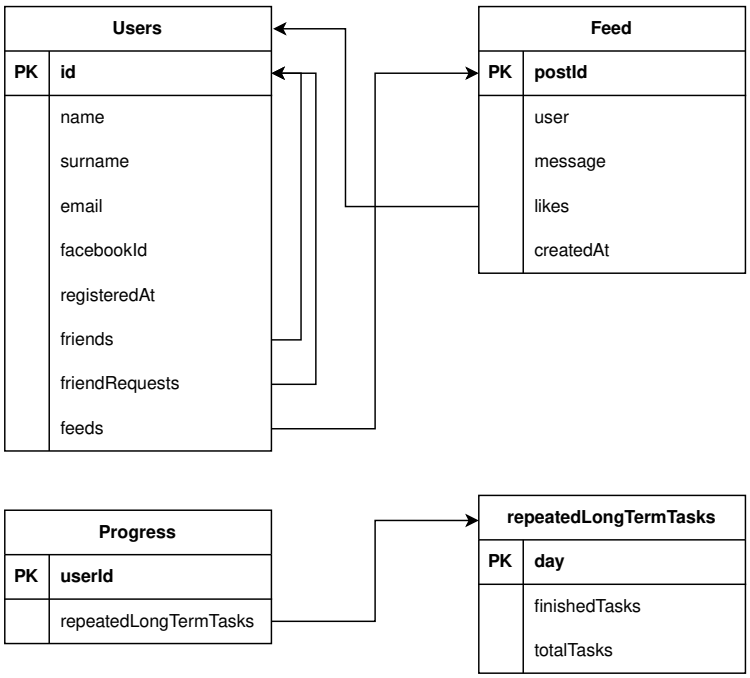
2.6 Databázový model

Ačkoliv nemají NoSQL databáze rigidní schéma je dobré při návrhu aplikace zvážit, jakým způsobem budou data aplikace ukládána. K tomu poslouží vytvoření databázového modelu. Pro nově vytvářenou aplikaci lze nalézt databázový model na obrázcích 2.13 a 2.14.

Databáze se bude skládat ze 4 kolekcí. V kolekci „Users“ budou uloženy informace o uživateli jako jeho jméno a příjmení, email, facebook ID, přátelé a žádosti o přátelství, dostupné feedy a datum registrace. Jménem dokumentu bude ID uživatele poskytované Firebase Authentication službou. Další kolekcí bude kolekce „Goals“ ve které se budou ukládat jednotlivé cíle a úkoly uživatele. Zde také bude jméno dokumentu ID uživatele. Třetí kolekcí bude „Feed“. Zde budou ukládané generované příspěvky od uživatelů. Jménem každého dokumentu bude unikátní identifikátor. Tento identifikátor se poté bude ukládat do pole feeds v kolekci uživatelů. Čtvrtou kolekcí bude „Progress“ ve které se budou průběžně ukládat hodnoty počtu plnění dlouhodobých úkolů. Jménem každého dokumentu je také ID uživatele.



Obrázek 2.13 Model kolekce Goals



Obrázek 2.14 Model kolekce Users, Feed a Stats

Kapitola 3
Implementace

Tato kapitola se věnuje samotné implementaci navržené aplikace. Budou zde popsány jednotlivé použité technologie včetně využitého programovacího jazyku, frameworků a knihoven, které byly použity k vývoji aplikace, a jak byla aplikace strukturována a organizována. Kromě toho se tato kapitola také zaměřuje na samotnou distribuci aplikace a způsob monitorování pádů aplikace.

3.1 Verze Android API

Výběr správné verze Android API je jedním z klíčových aspektů vývoje aplikace. Určuje kompatibilitu a dostupnost funkcí operačního systému Android. Při vývoji aplikace je nutné určit 3 verze:

Kompilační verze SDK (Compile SDK version) určuje jakou verzí SDK bude aplikace zkompilována. Neovlivňuje však chování za běhu aplikace. Stanovuje sadu rozhraní a knihoven, které bude aplikace mít k dispozici během kompilace.

Minimální verze SDK (Minimum SDK version) určuje minimální verzi operačního systému Android, pro kterou bude aplikace spustitelná. Pro nižší verze nebude možné aplikaci nainstalovat. Obchod Google Play podle této hodnoty určuje, pro která zařízení uživatele aplikaci nabídnout.

Cílová verze SDK (Target SDK version) určuje verzi pro kterou je aplikace otestována a systém nemusí aktivovat žádný režim kompatibility. [22]

Cílová verze SDK musí být vždy větší nebo stejná jako minimální verze a menší nebo stejná jako kompilační verze. Dále obchod Google Play požaduje pro nové aplikace, aby jejich target SDK verze byla nejnovější verze Android vydaná v průběhu posledního roku a pro stávající aplikace v průběhu posledních 2 let. [22, 23] Mnoho knihoven také můžou vyžadovat určitou minimální verzi SDK, kterou potřebují pro svoje fungování. Například sada knihoven Jetpack vyžaduje minimální verzi 14 a pro použití knihovny Jetpack Compose (knihovna pro nejnovější způsob tvorby uživatelského rozhraní aplikace na Android) je stanovena minimální verze 21. [24]

V této práci byla pro nově vytvářenou aplikaci zvolena verze 23, která má podporu 98,2 % všech Android zařízení na světě. Volba této nízké verze API neomezuje funkčnost aplikace neboť funkce, které nejsou podporovány nižšími verzemi Androidu jsou označeny speciálními anotacemi, které zabrání jejich použití. Pokud vývojář chce využít dané funkce, musí pro tyto verze napsat speciální kód, který bude určen pro starší verzi API.

3.2 Technologie a knihovny

V této podkapitole jsou popsány použité technologie a knihovny v samotné implementaci práce.

3.2.1 Kotlin

Kotlin je staticky typovaný vysokoúrovňový programovací jazyk vyvíjený společností JetBrains od roku 2011. V roce 2017 na konferenci Google I/O byla oznámena oficiální podpora jazyka Kotlin a v roce 2019 Google oznámil, že pro vývoj Android aplikací bude primárně prosazovat právě Kotlin oproti předtím používané Javě [25]. Programovací jazyk Kotlin má oproti Javě mnoho výhod:

Jednoduchost: Kotlin nabízí mnoho syntaktických zkratek usnadňující psaní kódu. Díky tomu je obvykle kód kratší, přehlednější a zvyšuje produktivitu programování.

Stabilita: Kotlin obsahuje funkce, které zabraňují častým chybám v kódu. Jednou z nich je například „NullPointerException“ ke které za normálních okolností v Kotlinu nemůže dojít. Díky tomu jsou tyto aplikace méně náchylnější k pádům.

Interoperabilita: Protože je Kotlin kompilovaný do bytekódu Javy, je pro něj zajištěna plná interoperabilita. To znamená, že z Kotlinu je možné využívat Java kódy a knihovny bez omezení.

Asynchronní běh: Kotlin obsahuje jednoduchý způsob pro ovládání asynchronního kódu zvaný „Coroutines“. [25]

Jak je zmíněno výše, jednou z výhod jazyka Kotlin je jeho přístup k asynchronnímu programování pomocí Coroutines, který spolu s Kotlin Flows poskytují řešení pro reaktivní programování. Flow poskytuje abstrakci proudu dat, který je generování asynchronně a následně konzumován v jiné části aplikace.

V implementaci navrhnuté aplikace jsou Coroutines a Flows využity k získávání a zpracování dat ze strany serveru. V repozitáři je vytvořen flow, který vysílá data do proudu dat. Zpravidla jako první vyšle stav načítání a po úspěšném provedení úkonu je vyslán úspěšný stav spolu se získanými daty nebo v případě neúspěchu je vyslán chybný stav se zprávou chyby. Tento proud je poté přes „Use Case“ třídu poskytnut ViewModelu, která daná data zpracovává a poskytuje View vrstvě.

3.2.2 Koin

Koin je framework pro řešení správy závislostí (tzv. DI - dependency injection) pro jazyk Kotlin. Dependency injection je technika pro oddělení závislostí tříd. Dosahuje toho tím, že oddělí vytváření závislostí od jejich používání. Tato technika zvyšuje modularitu aplikace a snižuje provázanost jednotlivých komponent. [26]

Třídy je nutné rozdělit do 4 rolí:

Service je třída, která bude využita v jiné části kódu.

Klient je třída, která využívá service třídy.

Interface je rozhraní, které service třída implementuje a klient využívá.

Injector vytváří (instancuje) service třídu a vkládá ji do klienta. [26]

Pro použití Koin frameworku je potřeba vytvořit moduly, které zadefinovávají, jakým způsobem vytvořit jednotlivé třídy. Na příkladu 3.1 je ukázka použití Koin modulu, kde pro každý use case, který je reprezentován pomocí rozhraní, je vytvořen singleton[1] (pomocí příkazu single). Ve složených závorkách se poté nachází vytvářená implementace daného use case a pomocí příkazu „get“ jsou do implementace dosazeny požadované závislosti třídy.

Všechny moduly jsou seskupeny v objektu „Module“. Pro použití těchto modulů v aplikaci je nezbytné je aplikovat při spuštění frameworku Koin (viz ukázka kódu 3.2).

```
val shortTermGoalsModule = module {
    single<GetGoalsListUseCase> { GetGoalsListUseCaseImpl(get()) }
    single<CreateShortTermGoalUseCase> { CreateShortTermGoalUseCaseImpl(get())
    ↪ }
    single<ReadShortTermGoalUseCase> { ReadShortTermGoalUseCaseImpl(get()) }
    single<UpdateShortTermGoalUseCase> { UpdateShortTermGoalUseCaseImpl(get())
    ↪ }
    single<DeleteShortTermGoalUseCase> { DeleteShortTermGoalUseCaseImpl(get(),
    ↪ get()) }

    single<GetRepeatedTaskListUseCase> { GetRepeatedTaskListUseCaseImpl(get())
    ↪ }
    single<CreateRepeatedTaskUseCase> { CreateRepeatedTaskUseCaseImpl(get(),
    ↪ get()) }
    single<ReadRepeatedTaskUseCase> { ReadRepeatedTaskUseCaseImpl(get()) }
    single<UpdateRepeatedTaskUseCase> { UpdateRepeatedTaskUseCaseImpl(get(),
    ↪ get()) }
    single<DeleteRepeatedTaskUseCase> { DeleteRepeatedTaskUseCaseImpl(get(),
    ↪ get()) }
}
```

Výpis kódu 3.1 Ukázka vytvoření Koin modulu

3.2.3 Jetpack

Jetpack knihovna je sada komponent, knihoven a nástrojů usnadňující vývoj aplikací na Android vyvíjená společností Google od roku 2018. Obsahuje mnoho užitečných funkcí pro vývojáře jako například řešení kompatibility mezi různými verzemi Androidu, zjednodušení práce s uživatelským rozhraním, navigace mezi obrazovkami, správa úkolů běžících na pozadí aplikace aj. [27]

Knihovna Jetpack je rozdělena do několika částí (viz obrázek 3.1):

Foundation komponenty Obsahuje sadu knihoven pro kompabilitu starších verzí operačního systému Android. Dále poskytuje rozšíření jazyka Kotlin (Android KTX) pro snadnější práci s Android API, testovací framework Expresso UI a multidex podporu.

Architektonické komponenty Zaměřuje se na architekturu aplikace a ukládáním dat. Usnadňuje komunikaci mezi jednotlivými vrstvami a specifikuje doporučený návrh Android aplikace.

Komponenty chováni Poskytují integraci s Android službami jako například manažer stahování dat, spouštění medií na pozadí aplikace, notifikace, oprávnění aplikace aj.

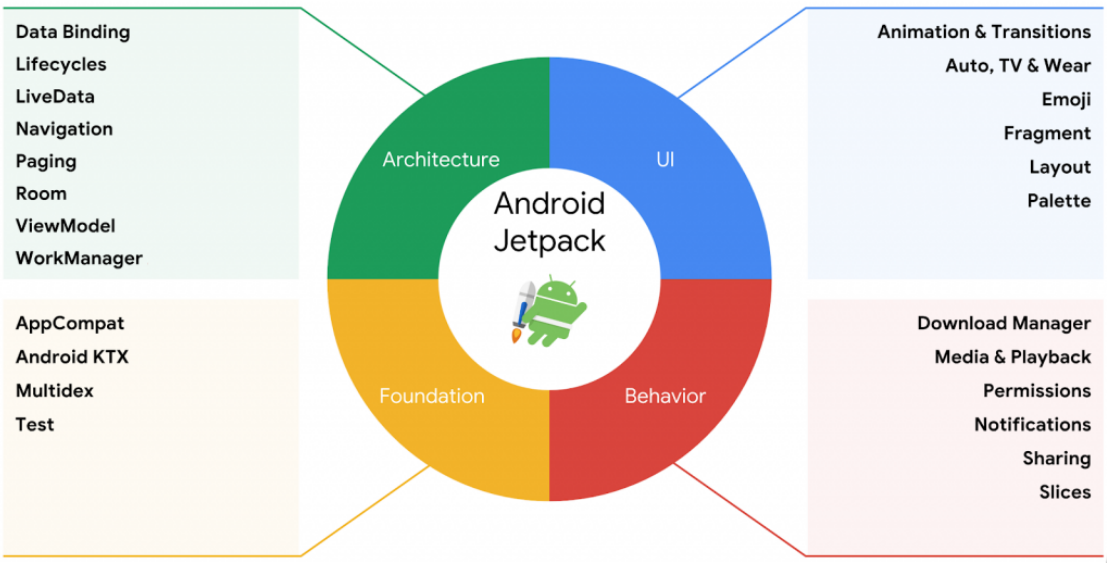
[1] Návrhový vzor, kde instance třídy v celé aplikaci je vytvořena právě jednou.

UI komponenty Obsahují předpřipravné UI komponenty a styly pro vytváření uživatelského rozhraní. [28]

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()

        startKoin {
            androidContext(this@App)
            modules(
                Module.authenticationModule,
                Module.progressModule,
                Module.shortTermGoalsModule,
                Module.longTermGoalsModule,
                Module.profileModule,
                Module.feedModule,
                Module.notificationModule
            )
        }
    }
}
```

Výpis kódu 3.2 Ukázka spuštění Koin frameworku se zadefinovanými moduly



Obrázek 3.1 Moduly knihovny Jetpack. Převzato z [29]

3.2.4 Jetpack Compose

Jetpack Compose je moderní způsob pro vytváření nativního uživatelského rozhraní. Framework byl poprvé oznámen na konferenci Google I/O 2019 a jeho stabilní verze byla vydána v roce 2021 [30]. Narozdíl od tradičního způsobu vytváření UI aplikace pomocí XML layoutů, vývojáři mohou psát UI pomocí deklarativního zápisu v jazyce Kotlin. To umožňuje snadnější a rychlejší vytváření UI a také lepší čitelnost kódu.

Framework využívá reaktivního programování pomocí stavů view vrstvy. Každá změna stavu způsobí rekompozici, tedy aktualizování UI aplikace. Tyto stavy jsou zpravidla poskytovány z ViewModelu. ViewModel tyto stavy aktualizuje podle požadovaných akcí. Například při načítání dat ViewModel zavolá Use Case, který pro danou akci poskytne Flow. Tento Flow následně ViewModel zpracuje a zapíše do stavů. Změnou těchto stavů se spustí rekompozice, která aktualizuje UI aplikace.

Jetpack Compose lze velmi jednoduše použít v kombinaci s MVVM architekturou. V tomto případě Composable funkce si vyžádá instanci ViewModelu, který poskytuje neměnitelné stavy obrazovky. Změny jsou prováděny pomocí metod ViewModelu, které se propagují Compose funkcemi pomocí lambda funkcí. Samotný ViewModel neobsahuje žádnou referenci na View vrstvu a je na ni tedy absolutně nezávislý.

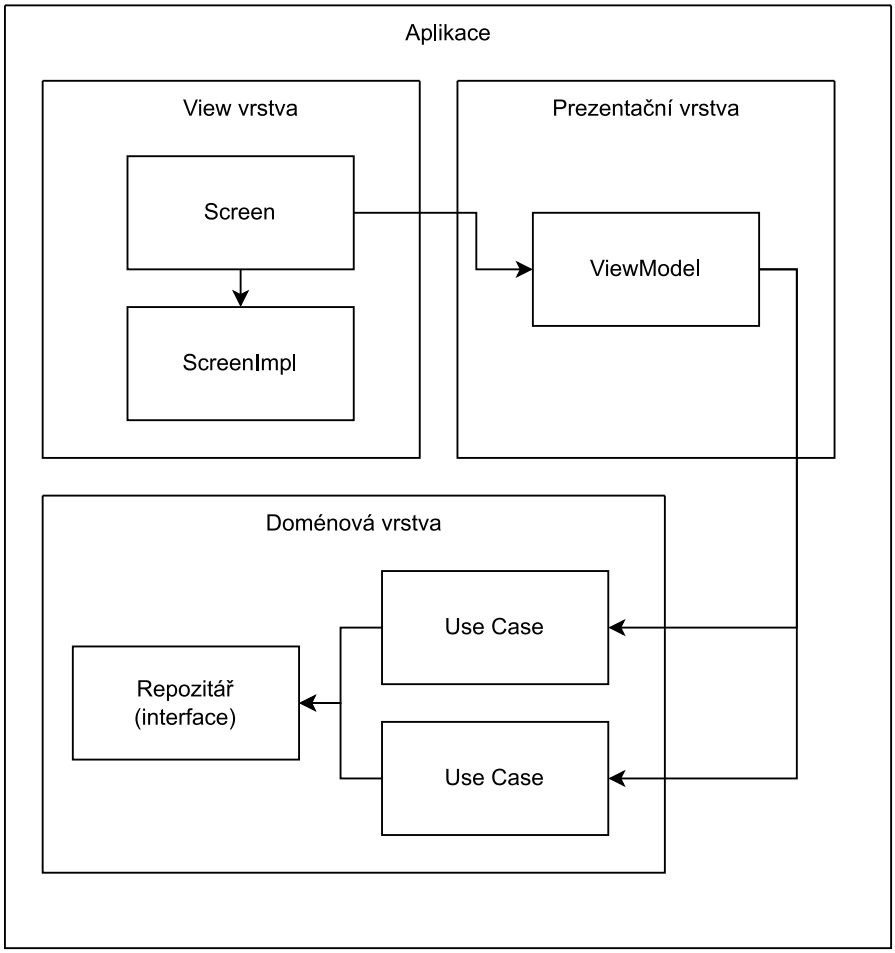
Téměř každá obrazovka aplikace potřebuje získat ViewModel a tzv. NavController[2] a případně další závislosti. Tyto třídy představují komplikaci, pokud je potřeba Composable funkci zobrazit v náhledu Android Studia nebo testovat uživatelské rozhraní, protože tyto třídy nemusí být jednoduše vytvořitelné mimo samotný běh aplikace. Z tohoto důvodu je každá obrazovka rozdělena na 2 Composable funkce a to na obrazovku (Screen) a její implementaci (ScreenImpl). Funkce Screen zajišťují získání ViewModelu a případných dalších závislostí a volají funkci ScreenImpl, které obsahují samotný vzhled dané obrazovky. Funkce ScreenImpl by jako argument měli dostat pouze data (v podobě doménových modelů) a lambda funkce pro provádění akcí a neměli by být nijak závislé na třídách doménové vrstvy. To zajistí, že samotný vzhled obrazovky bude možné zobrazit přímo ve vývojovém prostředí Android Studio a obrazovka bude případně testovatelná UI testy. Na obrázku 3.2 lze nalézt diagram propojení vrstev View vrstvy a ostatních vrstev aplikace.

3.2.5 Firebase Firestore

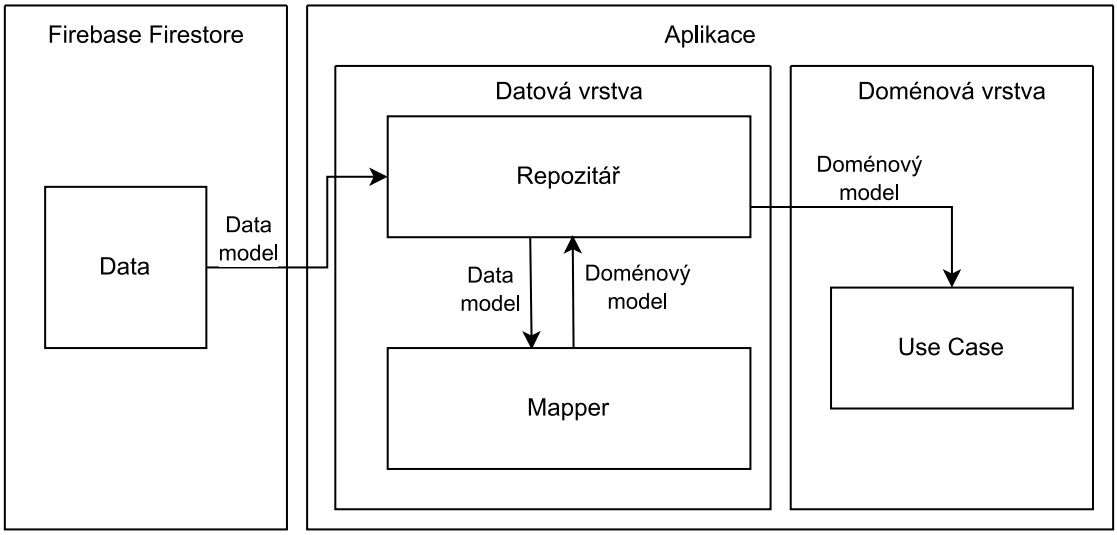
Jako úložiště pro nově vyvíjenou aplikaci byla zvolena dokumentová NoSQL databáze Firestore. Veškerá logika komunikace s databází se nachází v repozitářích v datové vrstvě aplikace. Pro komunikaci s API databáze je využito Firebase SDK, které obsahuje intuitivní rozhraní k ovládání databáze. Ve výpisu kódu 3.3 lze vidět přidání závislostí pro platformu Firebase.

Protože Firestore nepodporuje všechny typy dat používané v aplikaci jako například kolekci Set nebo datové a časové typy, jsou modely rozděleny do 2 vrstev – datového modelu a doménového modelu. Datový model reprezentuje podobu dat v cloudovém úložišti Firestore a repozitář jej využívá pro získávání a odesílání dat. Doménový model je využíván napříč aplikací. Tyto dva modely jsou propojeny pomocí mapovací funkce, která umožňuje vzájemné převádění mezi nimi. V těchto mapovacích funkcí je zadefinován převod nepodporovaných dat na jiný podporovaný typ dat. Například zmíněná kolekce Set je v datovém modelu převedena na kolekci List a časové údaje jsou převáděny na Unixový čas[3] včetně převodu z aktuální časové zóny zařízení na UTC čas. Na obrázku 3.3 je znázorněna komunikace s databází Firestore pro získání dat.

[2] Třída pro ovládání navigace obrazovek poskytována knihovnou Jetpack.
[3] Číslo udávající počet sekund od uplynutí 1. 1. 1970 00:00.



Obrázek 3.2 Diagram propojení View vrstvy, ViewModelu a doménové vrstvy



Obrázek 3.3 Diagram získání dat z databáze Firestore

```
dependencies {
    // BoM - Bill of Materials - pro jednotné verze firebase knihoven
    implementation(platform("com.google.firebase:firebase-bom:31.2.2"))

    // Kotlin rozšíření pro Firebase SDK
    implementation("com.google.firebase:firebase-firestore-ktx")

    // Crashlytics
    implementation("com.google.firebase:firebase-crashlytics-ktx")

    // Firebase Analytics
    implementation("com.google.firebase:firebase-analytics-ktx")
}
```

Výpis kódu 3.3 Ukázka přidání Gradle závislostí pro Firebase

3.3 Notifikace

Aplikace vyžaduje posílání notifikací uživateli na plnění úkolů. Protože aplikace nemusí být nutně spuštěna v čase plnění úkolu, je potřeba zajistit, aby operační systém Android v nastavený čas probudil aplikaci k zobrazení notifikace. K tomuto účelu slouží služba „AlarmManager“, která v zadaný čas pošle tzv. „Broadcast message“, kterou může aplikace odchytit a spustit kód. [31]

Od API verze 19 Google uvedl 2 typy alarmů. Nepřesný typ alarmu (inexact alarm) je navržen tak, aby optimalizoval spotřebu baterie zařízení tím, že jej systém shlukuje s ostatními nepřesnými alarmy a spouští je v jeden čas. Přesnost těchto alarmů je ovlivněna mnoha faktory jako například strategií spotřeby energie zařízení, počet dalších nepřesných alarmů aj. Tato přesnost se může pohybovat v řádech minut až hodin. [31] Protože si uživatel nastavuje přesný čas, kdy chce daný úkol splnit, je potřeba uživatele upozornit v právě zadaný čas úkolu. K tomuto slouží druhý typ alarmů.

Přesný typ alarmů (exact alarms) spustí požadovaný kód přesně v nastaveném čase. Pro použití přesných alarmů je však potřeba mít oprávnění. Pro zařízení do verze API 32 si stačí vyžádat o oprávnění „SCHEDULE_EXACT_ALARM“, které je při instalaci automaticky schváleno bez nutnosti potvrzení uživatele. Od verze API 33 je však nutné o toto potvrzení požádat uživatele a při pouštění kódu zkontrolovat, zda aplikace může nastavovat přesné alarmy. Od této verze API 33 však může vývojář zažádat o jiný typ oprávnění „USE_EXACT_ALARM“, které nevyžaduje potvrzení uživatele. Pro toto oprávnění však musí mít vývojář řádný důvod, jinak aplikace nebude schválena v obchodu Google Play. [31, 32]

Tyto oprávnění se nastavují v manifestu aplikace. Pro implementaci této aplikace bylo využito oprávnění „USE_EXACT_ALARM“, neboť tato aplikace má důvod ke spuštění přesných alarmů. V případě api nižší než 33 je využito oprávnění „SCHEDULE_EXACT_ALARM“. Ukázku těchto oprávnění lze nalézt v ukázce kódu 3.4.

Pro nastavení alarmu je nutné Androidu předat „BroadcastReceiver“, který bude zavolán v nastavený čas. V implementaci aplikace k tomuto slouží třída „ScheduledTaskNotificationReceiver“, která po zavolání vytvoří notifikaci a naplánuje další spuštění alarmu pro daný úkol. Samotná notifikace poté obsahuje 2 akce, pro které v aplikaci existují 2 „BroadcastReceivers“ a to pro splnění úkolu, kde se úkol označí za dokončený v daný den a pro odložení úkolu, který nastaví nový čas alarmu pro notifikaci daného úkolu. Tato nová notifikace přemaže stávající nastavený alarm, neboť je pro ně zadané stejné ID.

Pro každý úkol je v době vytvoření nebo úpravy nastaven alarm pro jeho notifikaci. Avšak tyto alarmy nejsou zachovány při vypnutí či restartování zařízení. Proto je potřeba při spuštění

operačního systému Android nastavit alarmy pro všechny úkoly uživatele. K tomuto slouží broadcast message „ACTION_BOOT_COMPLETED“, který po spuštění OS spustí nastavený kód v aplikaci. K tomuto chování stačí nastavit manifest aplikace (viz ukázka kódu 3.5). Třída TaskNotificationSchedulerReceiver se dotáže na všechny úkoly uživatele a nastaví pro ně patřičné alarmy.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
    <uses-permission android:name="android.permission.USE_EXACT_ALARM" />
    <uses-permission android:name="android.permission.SCHEDULE_EXACT_ALARM"
    ↪  android:maxSdkVersion="32" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"
    ↪  />

    <application>
        ...
    </application>
</manifest>
```

Výpis kódu 3.4 Ukázka oprávnění aplikace v manifestu

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application>
        <receiver android:name=
            ".repository.notification.ScheduledTaskNotificationReceiver"
            />
        <receiver android:name=
            ".repository.notification.TaskNotificationActionDoneReceiver"
            />
        <receiver android:name=
            ".repository.notification.TaskNotificationActionLaterReceiver"
            />
        <receiver
            android:name=
                ".repository.notification.TaskNotificationSchedulerReceiver"
            android:exported="false">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

Výpis kódu 3.5 Ukázka nastavení Broadcast receivers v manifestu

3.4 Struktura zdrojového kódu projektu

Zdrojový kód je členěn do 6 balíčků. Těmito balíčky jsou:

Feature: V tomto balíčku se nachází View vrstva (Composable funkce) a prezentační vrstva (ViewModely). Kód v tomto balíčku je rozdělen na moduly podle funkčních požadavků aplikace.

Repository: V tomto balíčku je kód také rozdělen podle jednotlivých modulů. V každém modulu se nachází balíčky Data, Domain a Model. Balíček Data obsahuje datovou vrstvu aplikace – tj. implementaci repozitářů, datové modely a mapovací funkce mezi datovým modelem a doménovým modelem. Balíček Domain obsahuje případy užití (Use Cases) a to jak jejich rozhraní, tak jejich samotnou implementaci a dále obsahuje rozhraní repozitáře, který doménová vrstva vyžaduje. Balíček Model obsahuje doménové modely daného modulu.

DI: Zde se nachází moduly frameworku Koin pro řešení závislostí tříd. Je zde umístěn jeden objekt „Module“, který jako jeho vlastnosti (property) obsahuje jednotlivé moduly obsahující definice závislostí tříd.

Navigation: Tento balíček obsahuje definice navigace mezi obrazovkami. Skládá se ze dvou tříd - NavigationScreens, která obsahuje seznam všech obrazovek, jejich cest a argumentů a Navigation, která tyto obrazovky zadefinovává s využitím Screen Composable funkcí z balíčku Feature.

UI: Obsahuje společné funkce a hodnoty uživatelského rozhraní.

App: V tomto balíčku se nachází hlavní kód aplikace, který se spouští se startem aplikace. Nachází se zde App třída, která spouští logovací framework Timber, framework Koin a nastavuje kanály pro notifikace aplikace. Dále se zde nachází jediná aktivita aplikace „MainActivity“, která spouští navigaci obrazovek. V modulu App se také nachází společné pomocné funkce a hodnoty.

Pro správné spuštění aplikace a aktivity je nutné tyto parametry zadefinovat v manifestu. Příklad takové definice lze nalézt v ukázkovém kódu 3.6.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:name=".app.system.App">
        <activity
            android:name=".app.system.MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</mainfest>
```

Výpis kódu 3.6 Ukázka nastavení aplikace a aktivity v manifestu

3.5 Distribuce

Po dokončení implementace aplikace je potřeba dostat aplikaci ke koncovým uživatelům nebo alespoň k testovacím uživatelům. Tento proces zahrnuje vytvoření instalovatelného balíčku a jeho poskytnutí uživatelům. Avšak sdílení těchto balíčků pomocí webových stránek či přímého posílání apk balíčku od vývojáře by nebylo praktické ani bezpečné, lze proto využít služby, které se postarají o poskytování a instalaci těchto balíčků.

3.5.1 Firebase App Distribution

App Distribution je nástroj pro distribuování aplikací bez nutnosti použití externích služeb a jejich složitých způsobů schvalování aplikace. Do App Distribution konsole lze přímo nahrát soubor .apk a snadno sdílet odkazy ke stažení aplikace mezi svými testery a uživateli. Tento nástroj také umožňuje vytvářet různé skupiny uživatelů a těmto skupinám lze zpřístupňovat konkrétní verze aplikace. Po nasdílení aplikace můžou uživatelé poslat zpětnou vazbu dané verze což usnadňuje vývoj aplikace. Služba Firebase App Distribution byla využita pro interní testování aplikace. Do této služby byla použita build varianta „debug“, která obsahuje debugovací symboly a není minimalizovaná. Díky tomu při pádu aplikace testera je snazší dohledat, ve které části kódu došlo k chybě.

3.5.2 Obchod Google Play

Obchod Google Play je oficiálním obchodem, kde lze stahovat Android aplikace a hry. Obchod je spravován společností Google. Na aplikace v tomto obchodu jsou kladeny přísné požadavky na bezpečnost aplikace, sdílení dat uživatelů, dostupnosti pro různé věkové skupiny aj.

Od srpna 2021 každá nová aplikace musí být publikována pomocí nového formátu „Android App Bundle“. Tento formát obsahuje veškerý kompilovaný kód a zdroje, ale odkládá samotné generování apk souboru a jeho podpis. Obchod Google Play použije bundle pro vygenerování optimalizovaného apk sobouru podle konfigurace zařízení, pro které je instalace určena. [33]

Pro publikování aplikace v obchodu Google Play je třeba si založit účet v Google Play Console a po založení účtu již stačí založit novou aplikaci, vyplnit požadované údaje, nahrát aplikaci a odeslat ji ke kontrole. Tato kontrola trvá obvykle několik dní až týdnů. Aplikaci lze publikovat do různých distribučních kanálů jako například interní testování, uzavřené testování, otevřené testování a produkce. Protože aplikace pro testovací účely využívá Firebase App Distribution, je pro obchod Google Play využit pouze produkční kanál. Pro distribuci do obchodu Google Play je využita build varianta „release“, která obsahuje obfuskovaný[4] a minimalizovaný kód aplikace. Nově vytvořenou aplikaci lze nalézt v obchodě Google Play pod názvem „Better Everyday“ na adrese `https://play.google.com/store/apps/details?id=cz.glazr.tomas.better_everyday`

3.6 Monitorování aplikace

Pro vylepšování aplikace je nutné sledovat a porozumět chování uživatelů v aplikaci. Tyto informace můžou být použity k vylepšení uživatelského rozhraní, optimalizaci výkonu, vylepšení uživatelské zkušeností a dalších aspektů aplikace. Analytické nástroje můžou také pomoci vývojářům identifikovat trendy a preference uživatelů a použít tyto informace k vytvoření lepších a úspěšnějších aplikací. Takovéto analytické nástroje poskytuje již v aplikaci integrovaná cloudová služba Firebase.

[4] Proces, při kterém je kód zamaskován tak, aby byl těžší na pochopení pro útočníky. Cílem je ztížení reverzní analýzy aplikace.

3.6.1 Firebase Analytics

Firebase Analytics je analytický nástroj cloudové platformy Firebase. Firebase platforma slouží pouze jako prostředník a veškerá analytická data jsou ukládána v Google Analytics. V těchto datech lze nalézt informace o používání aplikace jako například počty aktivních uživatelů v čase, distribuce uživatelů napříč verzemi aplikace, rozložení uživatelů podle zemí světa časy, typy používaných zařízení, způsoby interakce uživatele s aplikací, poměry uživatelů, které provedli platební transakci v aplikaci a mnoho dalších nástrojů.

Integrace aplikace se službou je velmi jednoduchá. Přidáním jedné Gradle závislosti (viz ukázka kódu 3.3) je služba nastavená a spuštěná.

Pro nově vyvinutou aplikaci budou tyto statistiky využity pro sledování počtu uživatelů a typy zařízení, které uživatelé používají. Díky tomu se při dalším vývoji aplikace může vývojář zaměřit na cílené testování na nejpoužívanějších zařízení.

3.6.2 Crashlytics

Firebase Crashlytics je nástroj pro monitorování pádů aplikace v reálném čase. Tento nástroj umožňuje rychle zjistit, kdy a proč došlo k pádu aplikace, což usnadňuje vývojářům opravu těchto problémů. Nástroj také k pádům zaznamenává typ, stav a informace o zařízení, verze aplikace, počet samotných pádů, počet ovlivněných uživatelů a výpis chyby z kódu (stacktrace), ze kterého lze poznat, kde chyba vznikla. Integrace Crashlytics s aplikací je také velmi jednoduchá. Stačí přidat 1 Gradle závislost (viz ukázka kódu 3.3) a aplikace automaticky začne shromažďovat záznamy pádu aplikace.

3.7 Shrnutí implementace

V této kapitole byla vytvořena aplikace Better Everyday podle návrhu z předchozích kapitol. V tabulce 1.1, která je součástí první kapitoly práce, byly porovnány existující řešení vůči požadovaným funkcím a parametrům. V tabulce 3.1 lze nalézt původní tabulku rozšířenou o nově implementovanou aplikaci.

Tabulka 3.1 Porovnání existujících řešení z tabulky 1.1, rozšířená o nově implementovanou aplikaci

	Sdílení	Rozdělení cílů	Intuitivní design	Účet v aplikaci	Zdarma
Dreamfora	Ne	Ano	Ano	Ano	Ano
Reach It	Ne	Ano	Ano	Ne	Ne
Higher Goals	Ne	Ano	Ne	Ano	Ne
Fabulous	Ne	Ne	Ne	Ano	Ne
Goal Setting T. P.	Ne	Ano	Ano	Ano	Ne
Better Everyday	Ano	Ano	Ano	Ano	Ano

Kapitola 4
Testování

Již během implementace byla aplikace průběžně testována a laděna vývojářem. Toto testování probíhalo na dvou emulovaných zařízení Pixel 4 s API 30 (Android 11) a zařízení Pixel XL API 33 (Android 13) a dále na soukromém fyzickém zařízení vývojáře Xiaomi MI 10T s API 31 (Android 12).

Po dokončení implementace bylo provedeno uživatelské testování. Pro uživatelské testování byl vytvořen scénář, podle kterého se uživatel řídil při jejím testování. Tento scénář se skládal z následujících bodů:

1. Zaregistruj se do aplikace.
2. Přidej si nového přítele v aplikaci pomocí emailové adresy.
3. Vytvoř krátkodobý cíl a přiřaď mu termín splnění.
4. Přidej k cíli opakovatelný a jednorázový úkol.
5. Vytvoř další libovolné cíle s několika úkoly (pro následující test filtrace).
6. Otestuj filtraci – aplikuj několik filtrů a zkontroluj, zda zobrazovaná data odpovídají zadaným filtrům.
7. Odstraň zadané filtry.
8. Změň způsob řazení cílů a zkontroluj, zda jsou cíle správně seřazeny.
9. Vytvoř dlouhodobý cíl a úkol k němu.
10. Pro libovolný úkol jej označ jako dnes splněný.
11. Lajkni libovolný příspěvek svého kamaráda.
12. Zobraz si u svého libovolného příspěvku počet lajků.
13. V notifikaci na splnění úkolu jej označ za splněný.
14. Další notifikaci ke splnění úkolu odlož na později.
15. Odhlas se z aplikace.

Testování se zúčastnili 3 uživatelé. Před zahájením testování bylo uživateli stručně vysvětlen koncept aplikace k čemu aplikace slouží. Toto vysvětlení mělo za úkol uživatele seznámit s tím, co od aplikace může očekávat a jaké jsou základní funkce, které aplikace nabízí. V průběhu testování mohl uživatel klást otázky a vyjadřovat své názory na daný úkol a na aplikaci samotnou.

4.1 Uživatel 1

Uživatel 1 disponoval zařízením Xiaomi Redmi Note 9 Pro s API verzí 29 (Android 10) na kterém aplikaci testoval. Po nainstalování aplikace z obchodu Google Play se uživatel zaregistroval do aplikace pomocí účtu Google, přičemž uživatel ocenil rychlost procesu registrace, které trvalo velmi krátce. Uživatel dále v bodu 2 přidal nového přítele. Zde uživatel zmínil (po obdržení žádosti od jiného přítele), že by bylo vhodné obdržet notifikaci s žádostí o přátelství. Dále uživatel pokračoval v bodech 3, 4. a 5, které proběhli bez jakýchkoliv obtíží. Při testování filtrace cílů v bodě 6 uživatel narazil na chybné chování, kde po vytvoření úkolu v cíli a následném vrácení se zpět se obrazovka s cíli neaktualizuje a tedy filtrace nepočítá se změněnými daty. Po překliknutí mezi záložkami se však již nová data načtou. Dále uživatel měl problém s filtrací počtu úkolů, neboť maximální počet úkolů v cílech byl 1 a tedy posuvník je potřeba posunout úplně do krajní polohy. Nakonec uživatel ještě zmínil kombinaci tlačítka filtrace v pravém horním rohu a následné záložky objevující se vespod aplikace, která se mu nezamlouvala. Další body 7 a 8 týkající se odstranění filtrů a řazení cílů proběhli v pořádku a uživatel k nim neměl žádné připomínky. S bodem 9 také uživatel neměl problém. V bodě 10 měl uživatel problém nalézt, kde provést danou akci. Po delším hledání ji na domovské obrazovce však našel. Zde bylo zmíněno, že by bylo vhodné tuto akci také nabízet přímo u úkolu. S prací s lajky příspěvků v bodech 11 a 12 nenastal žádný problém, nicméně uživatel zmínil, že by bylo uživatelsky přívětivé zobrazit počet lajků přímo na kartě příspěvku. V bodech 13 a 14 nenastala žádná potíž. S posledním bodem aplikace 15 neměl uživatel žádný problém.

Při testování byla objevena 1 vážnější chyba, kde ačkoliv aplikace nedisponuje tmavým režimem, byly některé texty zobrazeny bílým písmem na bílém pozadí v případě, že mobilní zařízení bylo přepnuto do tmavého režimu. Tato chyba se však na zařízeních vývojáře ani na ostatních zařízení testovacích uživatelů neprojevila. Po dokončení testování byl udělen prostor uživateli k celkovému hodnocení aplikace a dalších případných poznámek. Uživatel zmínil možné vylepšení aplikace mezi které patří:

- možnost změnit profilovou fotku účtu,
- možnost komentovat příspěvky,
- možnost filtrovat příspěvky podle uživatele a podle typu příspěvku (zahájení cíle, dokončení úkolu a dokončení cíle).

4.2 Uživatel 2

Uživatel 2 testoval aplikaci v emulátoru Bluestacks s emulovaným zařízení OnePlus 3T s API verzí 25 (Android 7). Podle bodu 1 začal uživatel s registrací. Zde bylo zmíněno, že by bylo vhodné napřímo napsat požadavky na heslo místo zobrazení chybové hlášky při zadání nedostatečného hesla. Následně podle bodu 2 uživatel úspěšně odeslal žádost o přátelství. Zde uživatel upozornil na 2 drobné nedostatky. Žádost o přátelství lze odeslat i uživateli, který v aplikaci vůbec neexistuje. Na druhou stranu se toto může vnímat jako bezpečnostní prvek proti hledání, jaké emailové adresy jsou v aplikaci zaregistrovány. Druhým zmíněným nedostatkem bylo, že po potvrzení příchozí žádosti není uživateli zobrazena žádná potvrzující zpráva. Dále uživatel pokračoval body 3, 4 a 5. Zde uživatel byl lehce zmaten, který typ úkolu uživatel vytváří, ale po chvíli uživatelské rozhraní pochopil. Uživatel zmínil menší nedostatky, kde po vytvoření jednorázového úkolu je přesměrován na obrazovku editace cíle s vybranou záložkou opakovatelných úkolů namísto jednorázových. Dále bylo zjištěno, že cíl i úkol lze vytvořit beze jména. Uživatel pokračoval body 6, 7 a 8, které proběhli bez problémů. Uživatel akorát zmínil nemožnost zrušení filtru data splnění cíle přímo ve vyskakovacím okně filtrů. Filtr ale dokázal zrušit v hlavičce obrazovky. Bod 9 proběhl bez problémů. S bodem 10 měl uživatel problémy a muselo mu být prozrazeno, kde tuto

funkcionalitu nalezne. Uživatel zmínil, že by tuto funkci očekával také přímo v editaci úkolu. Na domovské obrazovce by také uživatel ocenil jiné uspořádání prvků spolu s vysvětlivkami, nicméně po zamyšlení všechny prvky domovské obrazovky a jejich zobrazovaná data pochopil. V bodu 12 si uživatel měl zobrazit počet lajků. Tento úkol uživateli chvíli trval, než přišel na to, kde tuto informaci nalezne, nicméně zmínil, že toto řešení není příliš intuitivní a navrhnul oddělení tlačítka pro zobrazení seznamu lajků a samotného lajknutí příspěvku. Nakonec body 11, 13, 14 a 15 proběhly bez potíží.

Uživatel po dokončení testování dostal prostor k celkovému vyjádření. Aplikace se uživateli celkem líbila, zejména vyzdvihnul použití čistého jednoduchého designu, který nabývá v dnešní době popularity.

4.3 Uživatel 3

Uživatel 3 pro testování použil zařízení Xiaomi Mi 9 Lite s API verze 29 (Android 10). V bodu 1 uživatel vyzdvihnul možnost přihlášení pomocí účtu Google, kterou použil pro samotnou registraci. V bodu 2 uživatel odeslal úspěšně žádost o přátelství. S žádostí o přátelství však uživatel objevil chybu, kde uživatel jednak může odeslat žádost sám sobě (a stát se tedy přítel sám sebe) a jednak může odeslat žádost již přidanému příteli. Dále uživatel pokračoval ve vytváření krátkodobých cílů a úkolů z bodů 3, 4 a 5. Ty proběhli v pořádku, nicméně uživatel měl poznámku k opakovatelným úkolům, kde by výchozí hodnotou měli být zaškrtnuty všechny dny a tedy i pozitivní číslo počtu splnění úkolu do jeho dokončení. Navíc ačkoliv tento počet povoluje na vstupu pouze čísla pomocí vynucené numerické klávesnice, uživateli se pomocí zkopírování textu povedlo do vstupu dostat záporné číslo, které poté bylo i aplikací přijato. Testování pokračovalo body 6, 7, 8 a 9, které proběhli bez jakýchkoliv potíží. V bodě 10 stejně jako uživatel 2 upozornil, že by tato funkce měla být také přímo v editaci úkolu. Dále také upozornil na desetinná čísla objevující se v popisku svislé osy v grafu dlouhodobých cílů. Uživatel poté pokračoval na bod 12, ve kterém stejně jako uživatel 2 upozornil na neintuitivnost tlačítek příspěvků. Zbylé body 11, 13, 14 a 15 uživatel zvládnul bez potíží. Na obrazovce Feed stejně jako uživatel 1 navrhnul možné vylepšení pomocí filtrace uživatelů a typů příspěvku.

Po dokončení testování dostal uživatel prostor k vyjádření. Zde kromě již zmíněných chyb vyzdvihnul design aplikace.

4.4 Shrnutí

Většina problémů, na které uživatelé upozornili, byly v souvislosti s uživatelským rozhraním, jako například umístění tlačítka splnění úkolu, tlačítko lajku u příspěvků či výchozí hodnoty opakovatelného úkolu. Během testování se objevily i vážnější chyby. Například chyba s tmavý režimem zařízení, kde aplikace u jednoho uživatele špatně zobrazovala barvu textů, nicméně po testování byla aplikace spuštěna na několika dalších emulátorech stejné nebo podobné verze Androidu a tuto chybu se nepodařilo nasimulovat. Dalším problémem, který je potřeba opravit v budoucích verzích aplikace, je špatné ošetření žádostí o přátelství, kde je možné odesílat žádosti již přidaným přátelům či sobě samotnému. Testování také přineslo nápady na vylepšení aplikace jako například možnost komentářů k příspěvkům či filtry příspěvků. Tyto nápady na vylepšení mohou být zohledněny v budoucím vývoji aplikace.

Závěr

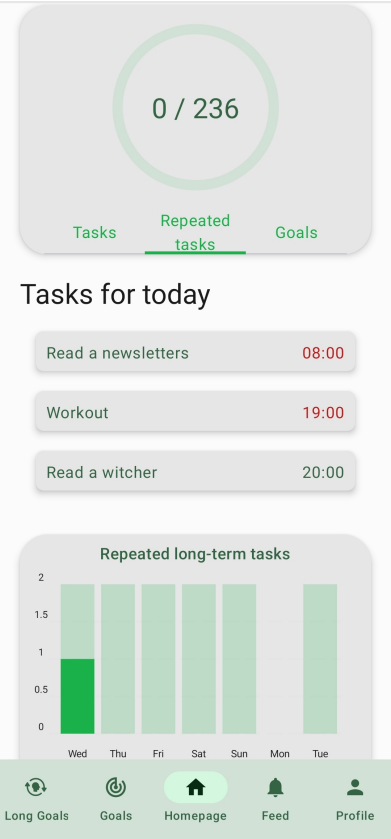
V této práci byla provedena analýza existujících aplikací na Android určených ke sledování osobních cílů na základě které byly stanoveny funkční a nefunkční požadavky takové aplikace. Poté byl proveden návrh nové aplikace ve kterém byly stanoveny případy užití, popsána architektura, návrh uživatelského rozhraní a návrh databáze spolu s cloudovými službami. Následně byla provedena implementace aplikace, kde byl popsán výběr verze Android API, výběr technologií a knihoven a byla popsána struktura zdrojového kódu. Po dokončení implementace byla aplikace zpřístupněna v obchodě Google Play a současně byla podrobena uživatelskému testování. Během tohoto testování byly zjištěny větší i menší nedostatky a nápady na vylepšení, které jsou popsány v kapitole 4.4 a měli by být zohledněny v budoucím vývoji aplikace. Dalšími nápady na vylepšení aplikace jsou například:

- vytvoření bodovacího systému, který by motivoval uživatele k plnění úkolů,
- vylepšení grafického rozhraní aplikace pomocí animací,
- přidání tmavého režimu aplikace,
- přidáním různých textů pro vytváření příspěvků ve feedu,
- přidání lokalizace aplikace do dalších jazyků.

Výsledkem práce je aplikace na Android splňující všechny zadefinované funkční a nefunkční požadavky a případy užití. V budoucnu je plánováno aplikaci dále rozvíjet a udržovat, aby mohla i nadále plnit potřeby uživatelů a přizpůsobovat se novým požadavkům a technologickým trendům.

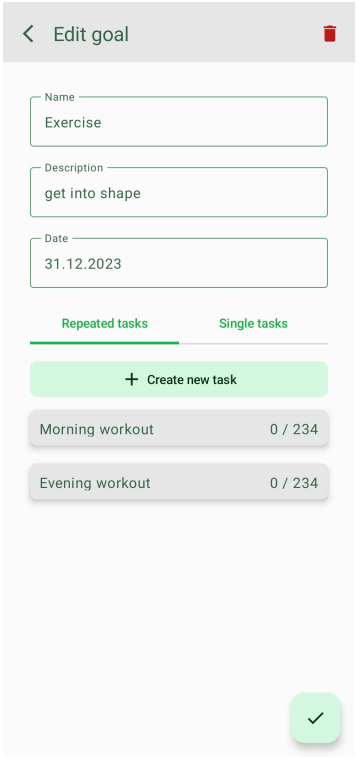
Příloha A

Ukázka aplikace



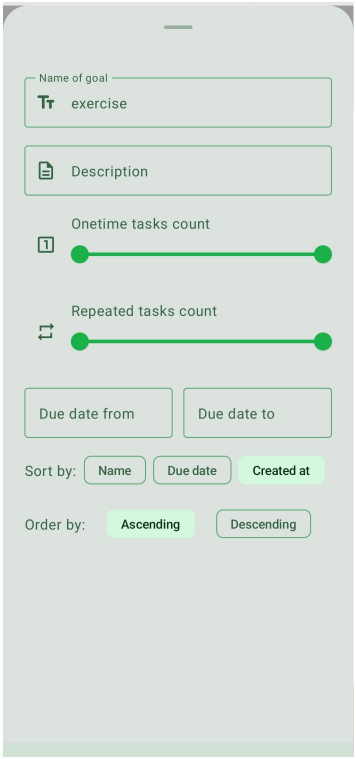
Obrázek A.1 Domovská obrazovka aplikace

Obrázek A.2 Obrazovka seznamu krátkodobých osobních cílů

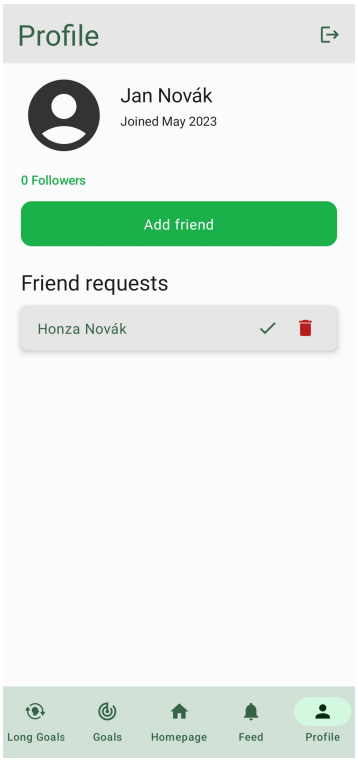


Obrázek A.3 Obrazovka editace krátkodobého cíle

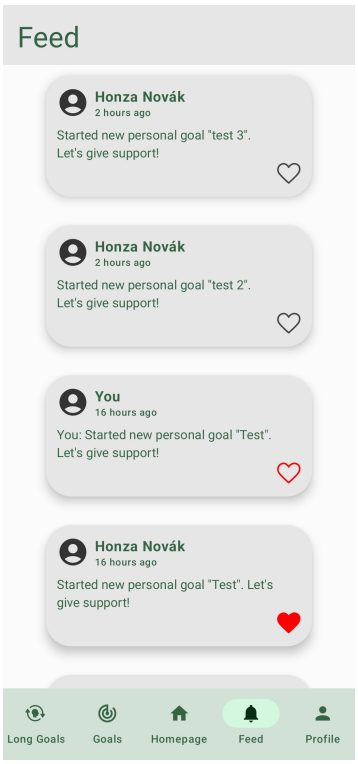
Obrázek A.4 Obrazovka seznamu dlouhodobých cílů



Obrázek A.5 Vyskakovací okno filtrace krátkodobých cílů



Obrázek A.6 Obrazovka profilu uživatele



Obrázek A.7 Obrazovka příspěvků

Bibliografie

1. GRABMEIER, Jeff. *Share your goals – but be careful whom you tell* [online]. 2019. [cit. 2023-04-13]. Dostupné z: https://news.osu.edu/share-your-goals--but-be-careful-whom-you-tell/.
2. CRAWFORD, Steven. *Four Keys to Accountability* [online]. 2020. [cit. 2023-04-13]. Dostupné z: https://www.mcgheepro.com/blog-post/accountability-what-it-is-and-how-you-can-get-it-2/.
3. DREAMFORA. *Dreamfora: Daily Goal Setting* [online]. 2023. [cit. 2023-04-03]. Dostupné z: https://play.google.com/store/apps/details?id=com.dreamfora.dreamfora.
4. REACH IT APPS. *Reach it: Goals, Habit Tracker* [online]. 2023. [cit. 2023-04-03]. Dostupné z: https://play.google.com/store/apps/details?id=com.reachApp.reach_it.
5. MINDFUL SUITE. *Higher Goals: Inspiring Habits* [online]. 2023. [cit. 2023-04-03]. Dostupné z: https://play.google.com/store/apps/details?id=com.mindfulsuite.goals.
6. THEFABULOUS. *Fabulous Daily Routine Planner* [online]. 2023. [cit. 2023-04-03]. Dostupné z: https://play.google.com/store/apps/details?id=co.thefabulous.app.
7. SUCCESS WIZARD. *Goal Setting Tracker Planner* [online]. 2023. [cit. 2023-04-03]. Dostupné z: https://play.google.com/store/apps/details?id=com.way4app.goalswizard.
8. MLEJNEK, Jiří. *Diagram případu užití* [online]. 2019. [cit. 2023-04-14]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/532098/mod_resource/content/1/Use%20Case%20Model.pdf.
9. NGUYEN, Quang. *Architecture patterns in Android — Android architecture design* [online]. 2017. Dostupné také z: https://medium.com/android-news/architecture-patterns-in-android-abf99f2b6f70.
10. GEEKSFORGEEKS. *MVC Design Pattern* [online]. 2023. [cit. 2023-04-16]. Dostupné z: https://www.geeksforgeeks.org/mvc-design-pattern/.
11. RISHU_MISHRA. *Difference Between MVC, MVP and MVVM Architecture Pattern in Android* [online]. 2022. [cit. 2023-04-16]. Dostupné z: https://www.geeksforgeeks.org/difference-between-mvc-mvp-and-mvvm-architecture-pattern-in-android/.
12. BLOCK, Glenn. *What are MVP and MVC and what is the difference?* [online]. 2021. [cit. 2023-04-16]. Dostupné z: https://stackoverflow.com/questions/2056/what-are-mvp-and-mvc-and-what-is-the-difference.

13. LEOPOLDOVIĆ, Luka. *MVP VS MVVM - CHOOSING THE RIGHT ANDROID ARCHITECTURE* [online]. 2020. [cit. 2023-04-16]. Dostupné z: https://www.bornfight.com/blog/mvp-vs-mvvm-choosing-the-right-android-architecture/.
14. RISHU_MISHRA. *MVVM (Model View ViewModel) Architecture Pattern in Android* [online]. 2022. [cit. 2023-04-16]. Dostupné z: https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/.
15. MARTIN, Robert C. *The Clean Architecture* [online]. 2012. [cit. 2023-04-16]. Dostupné z: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html.
16. GOOGLE. *Guide to app architecture* [online]. 2023. [cit. 2023-04-16]. Dostupné z: https://developer.android.com/topic/architecture.
17. REALTIMEBOARD. *Miro* [online]. 2023. [cit. 2023-04-16]. Dostupné z: https://miro.com/.
18. MARK DRAKE, Ostezer. *A Comparison of NoSQL Database Management Systems and Models* [online]. 2014. [cit. 2023-05-04]. Dostupné z: https://www.digitalocean.com/community/tutorials/a-comparison-of-nosql-database-management-systems-and-models.
19. MONGODB. *Relational vs. Non-Relational Databases* [online]. 2023. [cit. 2023-04-25]. Dostupné z: https://www.mongodb.com/compare/relational-vs-non-relational-databases.
20. MONGODB. *What are ACID Properties in Database Management Systems?* [online]. 2023. [cit. 2023-04-25]. Dostupné z: https://www.mongodb.com/basics/acid-transactions.
21. GOOGLE. *Cloud Firestore* [online]. 2023. [cit. 2023-04-25]. Dostupné z: https://firebase.google.com/docs/firestore.
22. LAKE, Ian. *Picking your compileSdkVersion, minSdkVersion, and targetSdkVersion* [online]. 2016. [cit. 2023-04-09]. Dostupné z: https://medium.com/androiddevelopers/picking-your-compilesdkversion-minsdkversion-targetsdkversion-a098a0341ebd.
23. GOOGLE. *¡uses-sdk¿* [online]. 2023. [cit. 2023-04-09]. Dostupné z: https://developer.android.com/guide/topics/manifest/uses-sdk-element.
24. EUGENE. *Android API Levels* [online]. 2023. [cit. 2023-04-09]. Dostupné z: https://apilevels.com/.
25. GOOGLE. *Android's Kotlin-first approach* [online]. 2023. [cit. 2023-05-02]. Dostupné z: https://developer.android.com/kotlin/first.
26. THORBEN. *Design Patterns Explained – Dependency Injection with Code Examples* [online]. 2023. [cit. 2023-04-30]. Dostupné z: https://stackify.com/dependency-injection/.
27. GOOGLE. *Android Jetpack* [online]. 2023. [cit. 2023-04-28]. Dostupné z: https://developer.android.com/jetpack.
28. RISHU_MISHRA. *Introduction to Android Jetpack* [online]. 2022. [cit. 2023-04-28]. Dostupné z: https://www.geeksforgeeks.org/introduction-to-android-jetpack/.
29. JAIN, Vikas. *Getting started with Android Jetpack Architecture Components* [online]. 2020. [cit. 2023-04-28]. Dostupné z: https://codetoart.com/blog/getting-started-with-android-jetpack-architecture-components.
30. BELLINI, Anna-Chiara. *Jetpack Compose is now 1.0: announcing Android's modern toolkit for building native UI* [online]. 2021. [cit. 2023-04-28]. Dostupné z: https://android-developers.googleblog.com/2021/07/jetpack-compose-announcement.html.
31. GOOGLE. *AlarmManager* [online]. 2023. [cit. 2023-05-05]. Dostupné z: https://developer.android.com/reference/android/app/AlarmManager.

32. GOOGLE. *Manifest.permission* [online]. 2023. [cit. 2023-05-05]. Dostupné z: `https://developer.android.com/reference/android/Manifest.permission`.

33. GOOGLE. *About Android App Bundles* [online]. 2023. [cit. 2023-05-05]. Dostupné z: `https://developer.android.com/guide/app-bundle`.

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `better_everyday.apk` ..... instalační balíček aplikace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF