UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2024 Jan Franta

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Mobilní aplikace pro notifikaci důležitých termínů

Bakalářská práce

2024 Jan Franta

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jan Franta**
Osobní číslo:	**I21322**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Mobilní aplikace pro notifikaci důležitých termínů**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce je provedení porovnání současných řešení v oblasti mobilních aplikací (zejména pro OS Android) zaměřených na notifikace (možnost definování odložené notifikace pokud úloha není splněna) pro různé kategorie uživatelsky definovaných úloh.
V návaznosti na tuto analýzu dojde k definování požadavků na nově vytvářenou aplikaci a její implementaci pro OS Android, přičemž součástí řešení budou také jednotkové testy.
Předpokladem je využití programovacího jazyka Java nebo Kotlin.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. TAKOORDYAL, Kishan. *Beginning unity Android game development: from beginner to pro*. [United States]: Apress, [2020]. ISBN 978-1-4842-6001-2.
2. ARLOW, Jim a NEUSTADT, Ila. *UML a unifikovaný proces vývoje aplikací: průvodce analýzou a návrhem objektově orientovaného softwaru*. Brno: Computer Press, 2003. ISBN 80-7226-947-X.

Vedoucí bakalářské práce: **doc. Ing. Michael Bažant, Ph.D.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **15. prosince 2023**
Termín odevzdání bakalářské práce: **10. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2024

Prohlašuji:

Práci s názvem *Mobilní aplikace pro notifikaci důležitých termínů* jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 7. 5. 2024

Jan Franta v.r.

PODĚKOVÁNÍ

Chtěl bych poděkovat vedoucímu mé bakalářské práce doc. Ing. Michaelovi Bažantovi, Ph.D. za odborné vedení práce, za jeho ochotu a za poskytnuté rady a konzultace. Také bych rád poděkoval své rodině a přátelům, kteří mě v průběhu celého studia podporovali.

ANOTACE

Bakalářská práce se zabývá analýzou aktuálně dostupných mobilních aplikací, které umožňují uživateli zadávat úlohy s termíny a následně jej na tyto termíny notifikovat. Praktická část práce se zabývá vývojem mobilní aplikace pro OS Android, která tuto funkcionalitu uživateli poskytuje. Aplikace je realizována v programovacím jazyce Kotlin pomocí architektury MVVM. Úlohy jsou ukládány v SQLite databázi Room.

KLÍČOVÁ SLOVA

Mobilní aplikace, Android, notifikace, Kotlin, MVVM

TITLE

Mobile application for notifications of important deadlines

ANNOTATION

The bachelor thesis deals with the analysis of currently available mobile applications that allow the user to enter tasks with deadlines and subsequently notify the user of these deadlines. The practical part of the thesis deals with the development of a mobile application for Android OS that provides this functionality to the user. The application is implemented in Kotlin programming language using MVVM architecture. The tasks are stored in a Room SQLite database.

KEYWORDS

Mobile application, Android, notifications, Kotlin, MVVM

OBSAH

SEZNAM OBRÁZKŮ A TABULEK

SEZNAM ZDROJOVÝCH KÓDŮ

SEZNAM ZKRATEK

AI	Artificial Intelligence
AOSP	Android Open Source Project
APK	Android Application Package
CSS	Cascading Style Sheets
GPS	Global Positioning System
GUI	Graphical User Interface
HTML	HyperText Markup Language
IDE	Integrated Development Environment
JVM	Java Virtual Machine
JSON	JavaScript Object Notation
MVC	Model View Controller
MVP	Model View Presenter
MVVM	Model View ViewModel
OS	Operating System
UI	User Interface
UML	Unified Modeling Language
XML	eXtended Markup Language

ÚVOD

V dnešní době, kde rychlé životní tempo klade stále větší nároky na efektivní organizaci času a plánování úkolů, se mobilní aplikace stávají nenahraditelnými nástroji pro usnadnění každodenního života. Tato bakalářská práce se ve své teoretické části věnuje analýze aktuálně dostupných mobilních aplikací, které slouží k zadávání a správě úkolů s termíny, s důrazem na notifikování uživatele o nadcházejících úkolech.

Praktická část této práce je zaměřena na vývoj mobilní aplikace pro operační systém Android, která uživatelům umožní intuitivní zadávání úkolů s termíny a následně je systematicky notifikuje o blížících se událostech. Aplikace využívá programovacího jazyka Kotlin a implementuje architekturu MVVM k přehledné organizaci jednotlivých částí zdrojového kódu aplikace. Pro persistentní ukládání úkolů je využita databáze Room, zajišťující efektivní uchovávání dat. Aplikace také umožňuje data jednoduše exportovat do formátu JSON, ze kterého lze následně uložené úkoly znovu do aplikace naimportovat.

Cílem této bakalářské práce je nejen poskytnout přehled existujících mobilních aplikací pro správu úkolů, ale i prezentovat konkrétní implementaci nové aplikace, která může efektivně sloužit jako užitečný nástroj pro každodenní organizaci času a splnění stanovených cílů.

1 MOBILNÍ APLIKACE

Mobilní aplikace je takový typ softwaru, který je navržen a přizpůsoben k použití na mobilních zařízeních, jako jsou chytré telefony nebo tablety. Od klasických aplikací, které jsou navrhovány pro desktopové použití, se liší zejména přizpůsobením pro ovládání dotykovými obrazovkami, provozu na obrazovkách menších rozměrů a také optimalizacemi z pohledu nižších výpočetních kapacit, které jsou u mobilních zařízení značně omezené [1].

V současné době je popularita mobilních aplikací na vzestupu a na trhu se vyskytují miliony různorodých aplikací, z nichž každá má svůj specifický účel. Tento trend rostoucí popularity potvrzují údaje z Českého statistického úřadu, kde chytrý telefon dle těchto údajů používá 82 % osob v ČR [2].

1.1 Typy mobilních aplikací

1.1.1 Nativní aplikace

Nativní aplikace jsou vyvíjeny přímo pro operační systém mobilního zařízení. Využívají možností daného systému a benefitují z přístupu k hardwarovým funkcím zařízení, kterými může být například senzor GPS, akcelerometr, fotoaparát či mikrofon.

Tyto aplikace disponují vysokou výkonností díky tomu, že aplikace běží nativně a mohou tak efektivně zpracovávat větší množství dat.

Nevýhodou je nutnost vývoje pro každý operační systém zvlášť. Vývoj probíhá s využitím nativního programovacího jazyka – v případě platformy Android je to Java nebo Kotlin a Objective-C nebo Swift pro iOS aplikaci [1][3].

Existují ovšem také technologie, které se nazývají frameworky a jejich hlavním cílem je umožnit vývojářům vyvíjet nativní aplikace pro více platforem najednou. Mezi nejpopulárnější frameworky patří .NET MAUI, React Native nebo Flutter [4].

1.1.2 Webové aplikace

Webové aplikace nejsou instalovány přímo na zařízení uživatele, místo toho je k nim přistupováno prostřednictvím webového prohlížeče.

V tomto případě je aplikace univerzální, co se týče platformy. To je velká výhoda, protože stačí aplikaci vyvinout pouze jednou, a to použitím webových technologií, kterými jsou JavaScript, HTML a CSS. Často se chovají a vypadají jako nativní aplikace, ale jejich nevýhodou je, že mají omezené možnosti využití hardwaru zařízení [1].

1.1.3 Hybridní aplikace

Hybridní aplikace kombinují technologie nativních a webových aplikací. Aplikace je postavena pomocí webových technologií, ale je následně zabalena do nativního obalu, který aplikaci zpřístupní hardware zařízení.

Jsou zde stejné výhody jednotného vývoje aplikace jako je u webových aplikací, ale hybridní aplikace nedosahují takového výkonu jako je tomu u aplikací nativních – zpracování uživatelských akcí probíhá na vzdáleném serveru, nikoliv lokálně, a tím může docházet k mírným prodlevám v obsluze těchto aplikací [1][3].

1.2 Distribuce mobilních aplikací

K distribuci aplikací se nejčastěji používají distribuční platformy nazývané obvykle jako obchody s aplikacemi (anglicky app stores). Tyto obchody umožňují uživatelům prohledávat dostupné aplikace, které jsou obvykle přehledně kategorizovány dle žánrů, dále aplikace umožňují stahovat, instalovat nebo aktualizovat a případně je také hodnotit a recenzovat.

Dnešní mobilní operační systémy již mají některou z těchto platforem integrovanou do systému. V systému Android je dostupná platforma Google Play a v iOS je to App Store.

Před zpřístupněním aplikace v těchto obchodech musí vývojáři projít registrací k vytvoření vývojářského účtu a následně mohou aplikaci odeslat ke kontrole – každá z aplikací musí splňovat podmínky a zásady použití dané distribuční platformy [5].

V případě systému Android není integrovaná platforma Google Play jedinou možností instalace aplikací. Android umožňuje instalaci aplikací přes alternativní distribuční platformy třetích stran (Amazon AppStore, Samsung Galaxy Store a další) nebo pomocí instalačních balíčků v podobě APK souborů. Tento alternativní přístup však není doporučován, jelikož s sebou nese bezpečnostní rizika. Aplikace z těchto zdrojů nemusí být kontrolovány na přítomnost škodlivého kódu a nelze je tak považovat za bezpečné či důvěryhodné.

Proto také není ve výchozím nastavení systému instalace neznámých aplikací povolena. V systému iOS tato možnost není podporována a distribuce se tak provádí pouze prostřednictvím integrované platformy App Store [6][7].

1.3 Aplikace zaměřené na notifikace

Tato práce se zaměřuje na specifický typ mobilních aplikací, které umožňují uživateli ukládat různé kategorie úloh a následně jej na ně upozornit pomocí notifikace. Cílem těchto aplikací je ulehčit organizaci práce a mít přehled o nadcházejících úkolech.

Obvykle se od aplikací tohoto typu očekává:

- Rychlé přidávání úkolů – zadání nového úkolu či jakákoli jejich jiná organizace by měla být co nejsnazší a nejrychlejší, v co nejmenším počtu kroků.
- Organizace úkolů do skupin – aplikace by měla umožnit úkoly označovat a řadit do různých kategorií či skupin.
- Nastavení upozornění na termín úkolu – jedna z hlavních výhod oproti klasickým papírovým poznámkovým blokům, aplikace uživatele upozorní uživatele na nadcházející termín.

Správná aplikace by tedy měla svému uživateli pomoci zvládnout co nejvíce práce a přitom zajistit, aby dodržel všechny stanovené termíny [8].

2 ANALÝZA DOSTUPNÝCH APLIKACÍ

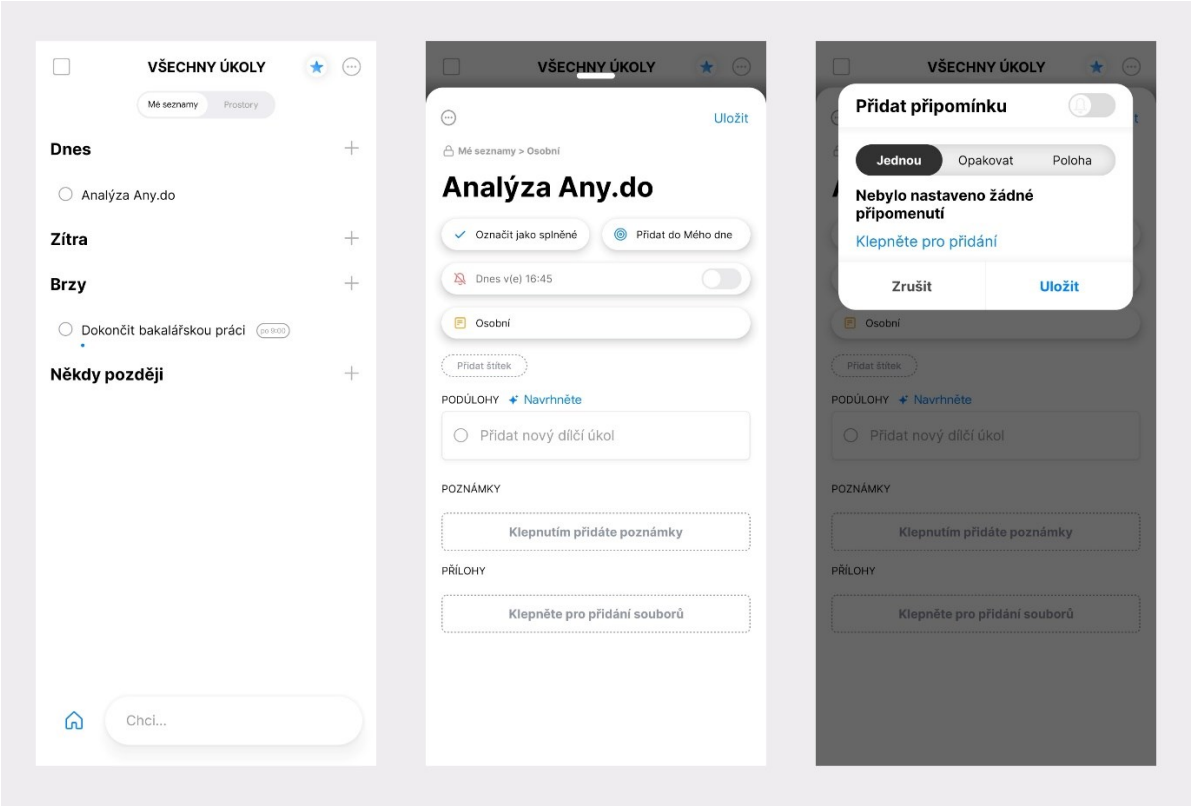
V současnosti existuje široký výběr mobilních aplikací, které slouží k organizaci a správě úkolů, a to ať už v osobním nebo pracovním kontextu. Tato část se věnuje analýze vybraných aplikací zaměřených na notifikace. Všechny níže uvedené aplikace jsou dostupné pro operační systém Android a byly získány z obchodu Google Play. Analýza byla provedena na vlastním zařízení s OS Android ve verzi 13.

Většina těchto aplikací nabízí rozšíření na placenou verzi, která obvykle nabízí více funkcí. V rámci této analýzy byly vyzkoušeny pouze základní bezplatné verze aplikací.

2.1 Any.do

Aplikace Any.do nabízí velmi příjemné uživatelské rozhraní, které je přehledné, intuitivní a umožňuje rychlou orientaci v nadcházejících úlohách.

Umožňuje vytvořit seznamy, do kterých lze úlohy zařadit. To umožňuje například oddělení pracovních úloh od úloh z osobního života. Úlohy lze dále rozlišovat pomocí štítků a případně k nim přiložit poznámku či přílohu ve formě obrázku, zvuku nebo videa. Také umožňuje vložit podúlohy, které pomocí AI dokáže z nadpisu navrhnout.



Obrázek 1. Rozhraní aplikace Any.do (zdroj vlastní)

K jednotlivým úkolům je možnost přidání připomínky, kde kromě základního jednorázového připomenutí, které se nastaví na specifický čas, lze úlohu připomínat opakovaně nebo je také možné připomenutí vyvolat dle současné polohy zařízení.

Při přijetí připomínky lze notifikaci odložit (nastavit nový čas připomenutí) nebo úlohu označit jako dokončenou.

K organizaci úloh aplikace nabízí několik režimů zobrazování seznamu:

- *Můj den* přehledně zobrazí nejbližší nadcházející úlohy.
- *Dalších 7 dní* zobrazuje úlohy v rámci týdne.
- *Všechny úkoly* zobrazí všechny úlohy seskupené podle času (Dnes, Zítra, Brzy, Později).
- *Kalendář* zobrazuje úlohy v kalendáři. Umožňuje zobrazit také události z jiných kalendářů na zařízení (vyzkoušen Google Kalendář).

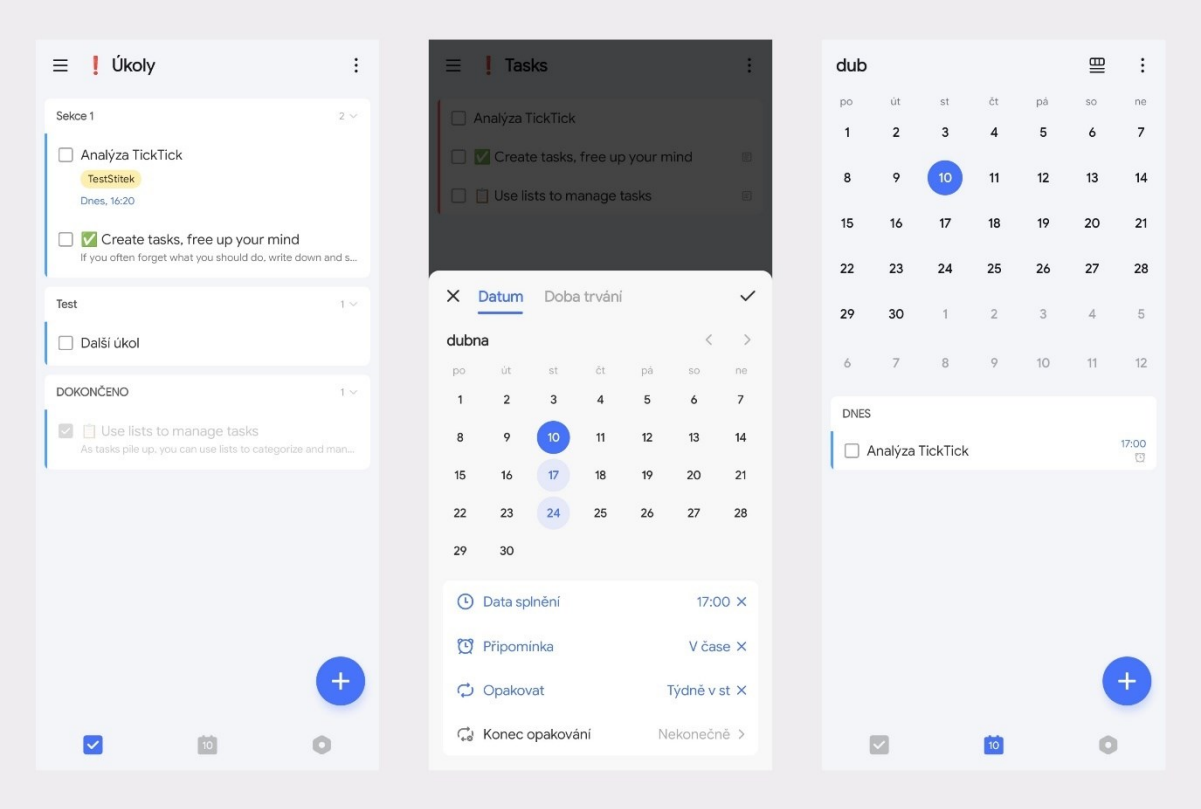
Při prvotním spuštění aplikace je nutné se přihlásit. Tento účet slouží k synchronizaci úloh mezi zařízeními. Any.do se dá také použít v kolaborativním prostředí. Má placenou verzi pro týmy, ve které poskytuje sdílený seznam úloh či propojení s jinými aplikacemi pro produktivitu. Cena je pro jednoho uživatele 5 dolarů za měsíc [9].

Mezi prémiové funkce aplikace patří denní plánovač (Denní Push), který v nastavené dny a čas uživatele pravidelně notifikuje ke zkontrolování nadcházejících úloh. Funkce integrace s aplikací WhatsApp, použití barevných štítků nebo notifikace na základě polohy jsou také součástí pouze placené verze aplikace. Poplatek za prémiové funkce je 109 Kč za měsíc při volbě předplatného na rok [9].

2.2 TickTick

Další analyzovanou aplikací je aplikace TickTick. Tato aplikace umožňuje kromě úloh ukládat také poznámky. Rozhraní této aplikace je intuitivní a přehledné, není zde ovšem na výběr z tolika možností zobrazování úloh. Na druhou stranu má TickTick výrazně větší možnosti přizpůsobení. Umožňuje nastavit výchozí hodnoty při přidávání nové úlohy (výchozí důležitost, seznam, značka a další) a podporuje vytváření a používání šablon pro úlohy.

Úlohy jsou zařazené do seznamů, které si uživatel může vytvořit. Tyto seznamy můžou být dále vloženy do složek. V seznamu je pak možnost úlohy přehledně seskupovat do sekcí a úloha se dá také připnout, přičemž se pak vždy zobrazuje na vrchu seznamu. K úlohám lze přiřadit podúlohy a vkládat obrázky nebo jakoukoli jinou přílohu. Dále lze úlohy organizovat pomocí štítků či nastavení důležitosti.



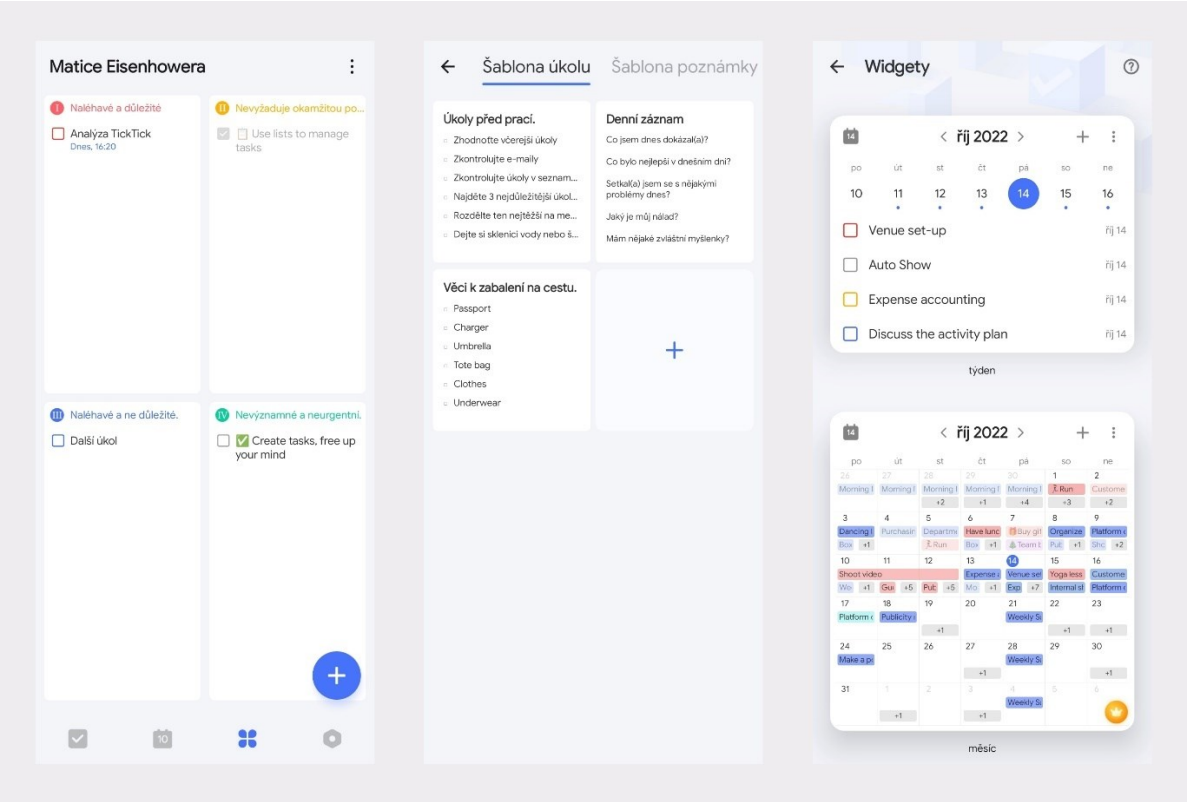
Obrázek 2. Rozhraní aplikace TickTick (zdroj vlastní)

K úloze je v případě uvedení data splnění možnost nastavení připomínky. Je na výběr z několika předvoleb času připomenutí, ale připomínku lze nastavit i na vlastní čas. Připomínku lze také nastavit jako opakovanou, kde kromě intervalu opakování lze nastavit také datum ukončení opakovaného připomínání. Připomínku lze také vyvolat na základě polohy zařízení.

Při přijetí připomínky ji lze z oznámení odložit nebo úlohu označit jako dokončenou.

K zobrazení úloh aplikace umožňuje tyto režimy:

- Klasické zobrazení seznamu, kde je možnost úlohy rozřadit do sekcí a úlohy jsou řazeny dle nejbližšího data splnění.
- Zobrazení úloh v rámci kalendáře, kde jsou přehledně vidět všechny nadcházející úlohy v rámci měsíce (či dne, týdne atd.).
- Na Eisenhowerově matici, která úlohy rozřadí dle jejich naléhavosti a důležitosti



Obrázek 3. Funkce aplikace TickTick (zdroj vlastní)

Aplikace TickTick disponuje celou řadou dostupných widgetů, které umožňují mít o nadcházejících úlohách přehled i bez nutnosti otevření aplikace. Další funkcí aplikace je sledování zvyků, při kterém aplikace napomáhá k udržení zvyků pomocí funkcí jako připomínání a zaznamenávání pokroků v rámci času. Pro ušetření času aplikace obsahuje mnoho již předkonfigurovaných zvyků různých kategorií.

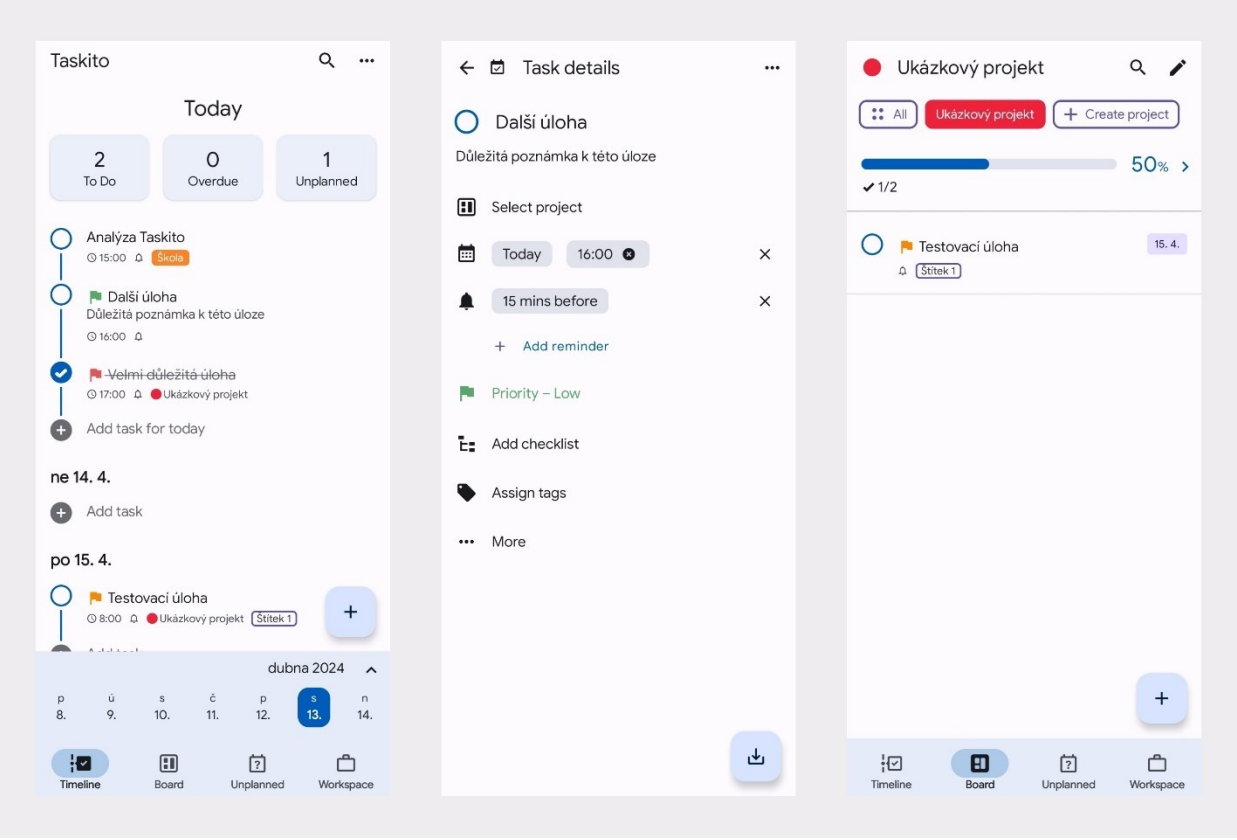
TickTick podporuje synchronizaci úloh mezi více zařízení, aplikaci lze ovšem plně využívat i bez přihlášení. Je zde také podpora sdílení seznamů, kdy seznam může spravovat více uživatelů a úlohy můžou být přiřazovány jednotlivým účastníkům.

Aplikace TickTick má také Premium verzi, která obsahuje funkce navíc, jako více možností zobrazení kalendáře (týdenní, denní, měsíční a další), možnost filtrování úloh ze seznamů (například filtrování pouze důležitých úloh pro nadcházející týden). Další prémiovou funkcí je možnost nastavení trvání úkolu nebo možnost nastavení více připomínek na jednu úlohu. Některé widgety a témata jsou také pouze dostupné s Premium verzí. Cena této verze je 99 Kč na měsíc nebo 990 Kč na rok [10].

2.3 Taskito

Aplikace Taskito má velmi přehledné rozhraní, které ve velké míře používá designový jazyk Material od společnosti Google. Umožňuje správu jak úloh, tak poznámek, a umožňuje tyto položky seskupovat do projektů. V těchto projektech pak lze přehledně sledovat průběh splnění daného cíle projektu. Záložka projektu ukazuje procentuální splnění všech úloh uložených v daném projektu a dále jsou zobrazovány všechny zatím nedokončené úlohy.

Úlohy jsou seskupovány do dnů dle jejich termínů, přičemž aplikace poskytuje 2 režimy zobrazení. Prvním je režim časové osy (Timeline mode), ve kterém jsou úlohy zobrazeny v seznamu dnů, do kterých jsou pak seskupovány. Druhý je denní režim, který zobrazuje pouze úlohy z vybraného dne (výchozím dnem je aktuální datum). Každý z těchto režimů má své výhody, první se hodí především pro udržení přehledu o nadcházejících úlohách v širším měřítku, zatímco denní režim poskytuje přehlednější, méně zahlcené rozhraní pro přehled o úlohách v rámci jednoho dne.



Obrázek 4. Rozhraní aplikace Taskito (zdroj vlastní)

V případě, kdy k úloze není přiřazen termín, je úloha umístěna v samostatném oddílu nenaplánovaných úloh. Podobně, pokud existuje nesplněná úloha, která má již po termínu dokončení, je tato úloha zahrnuta v oddílu zpožděných (overdue) úloh. Taskito pak zobrazuje počet úloh v těchto kategoriích (To Do, Overdue, Unplanned) pro aktuální den.

Kromě základních vlastností úlohy, jako je název a popisek či poznámka, Taskito umožňuje k úlohám přiřadit také prioritu, kde je na výběr z 5 stupňů důležitosti, označit úlohu štítkem nebo vytvořit odškrtávací seznam podúloh. Unikátní funkcí je možnost úlohu označit jako zrušenou, kdy není odstraněna ze seznamu a je pouze označena podobně jako splněná úloha. V Taskito je také, stejně jako v aplikaci TickTick, možnost vytvářet šablony pro úlohy i poznámky.

Taskito samozřejmě také umožňuje na úlohy nastavovat připomínky. Na jednu úlohu je možnost nastavení více připomínek a úlohu lze nastavit jako pravidelně se opakující, kdy se dá nakonfigurovat perioda připomínání, konkrétní dny v týdnu a také rozsah, kdy bude daná úloha opakovaně připomínána.

Při přijetí připomínky ji lze odložit a nastavit nový čas notifikace nebo ji označit jako dokončenou. Oproti ostatním aplikacím, v Taskito není možnost připomínku vyvolat na základě polohy zařízení.

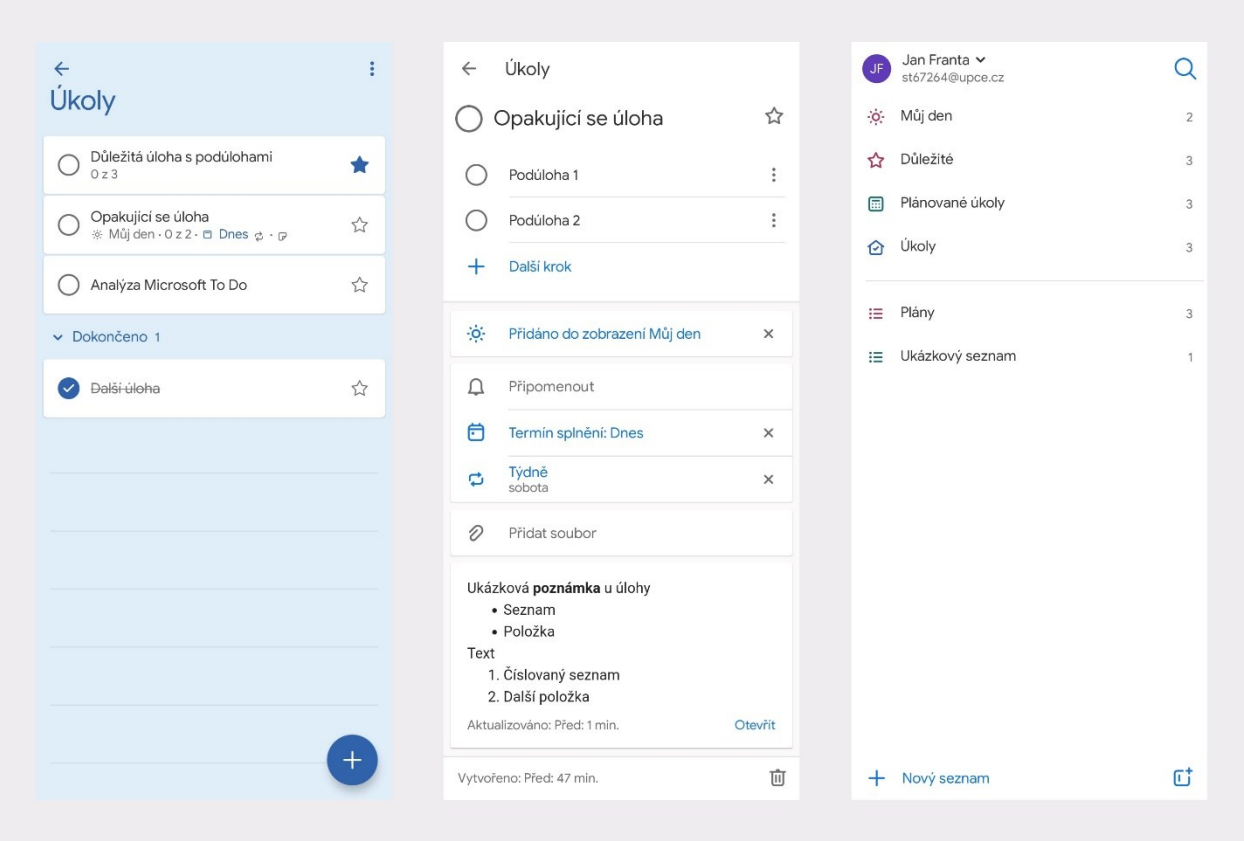
Taskito umožňuje úlohy synchronizovat mezi více zařízení pomocí Taskito účtu. Aplikace přihlašování ovšem nutně nevyžaduje, a kromě absence funkce synchronizace se dá plně využívat offline, bez přihlášení do účtu. Oproti ostatním aplikacím je synchronizace značně omezená. Taskito nedisponuje webovou aplikací a synchronizace je tak možná pouze v rámci nativní mobilní aplikace. Úlohy lze tedy spravovat pouze z mobilního zařízení, což je nevýhoda oproti konkurenčním aplikacím, kde je k úlohám přístup prakticky z jakéhokoli zařízení s webovým prohlížečem.

Volně dostupná verze aplikace Taskito má svá omezení a plnou funkcionalitu poskytuje Premium verze aplikace. Nastavování více připomínek v rámci jedné úlohy, neomezený počet projektů a opakovaných úloh a připomínek, funkce importu z kalendáře a některá témata (vzhledy) aplikace jsou funkce dostupné pouze v Premium verzi. Neplacená verze aplikace umožňuje omezený počet 3 projektů a 3 opakujících se úloh. Předplatné je nabízeno v několika možnostech. Taskito se dá předplatit měsíčně za 49 Kč, ročně za 250 Kč a také je nabízena možnost doživotní licence, která je zpoplatněna ve výši 499 Kč [11].

2.4 Microsoft To Do

Aplikace Microsoft To Do je velmi jednoduchá aplikace na správu úloh. Rozhraní aplikace je přehledné a ovládání je intuitivní. Úlohy lze rozřazovat do seznamů. U těchto seznamů lze pro rozlišení nastavit různé barevné motivy. Seznamy lze dále zařadit do skupin.

V seznamech se úlohy dají řadit podle několika faktorů: termínu, důležitosti, abecedy, data vytvoření, anebo pokud byly přidány do zobrazení Můj den.



Obrázek 5. Rozhraní aplikace Microsoft To Do (zdroj vlastní)

Kromě uživatelsky definovatelných seznamů jsou v aplikaci tyto seznamy:

- *Můj den* – tento seznam poskytuje přehled úloh s blízkým termínem splnění, ovšem úlohy se dají na tento seznam také manuálně přidávat (nebo z něj odebírat)
- *Důležité* – na tomto seznamu se vyskytují pouze úlohy, které byly v kterémkoli seznamu označeny hvězdičkou jako důležité
- *Plánované úkoly* – zobrazuje všechny úlohy s termínem splnění a umožňuje je pomocí nich filtrovat (například úlohy s termínem pouze v tomto týdnu)
- *Úkoly* – výchozí seznam pro přidávání úloh

K úloze se dají uložit podúlohy, termín splnění a připomínka nebo vložit poznámku, u které jsou dostupné pokročilejší možnosti formátování, jako jsou číslované seznamy nebo seznamy s odrážkami. Microsoft To Do nenabízí možnost označení úloh pomocí štítků, a úlohy jsou tak organizovány pouze podle seznamů. Úlohy se dají označit hvězdičkou jako důležité.

Úlohy se dají nastavit jako opakující a k úloze lze nastavit připomínku. Při přijetí oznámení připomínky lze, stejně jako u ostatních aplikací, připomínku odložit nebo úlohu označit jako dokončenou. Aplikace také nabízí připomínaní pro naplánování dne, kdy ve zvolené dny a čas uživatele notifikuje, aby se podíval na nadcházející úlohy.

Aplikace vyžaduje přihlášení do Microsoft účtu, díky kterému jsou úlohy synchronizovány. Úlohy tak pak jsou přístupné z jakéhokoli zařízení, Microsoft To Do totiž funguje také jako webová aplikace. Díky možnosti sdílení se seznamy dají zpřístupnit více uživatelům, kteří ho pak mohou upravovat. Tato aplikace nenabízí žádné předplatné a je dostupná zcela zdarma.

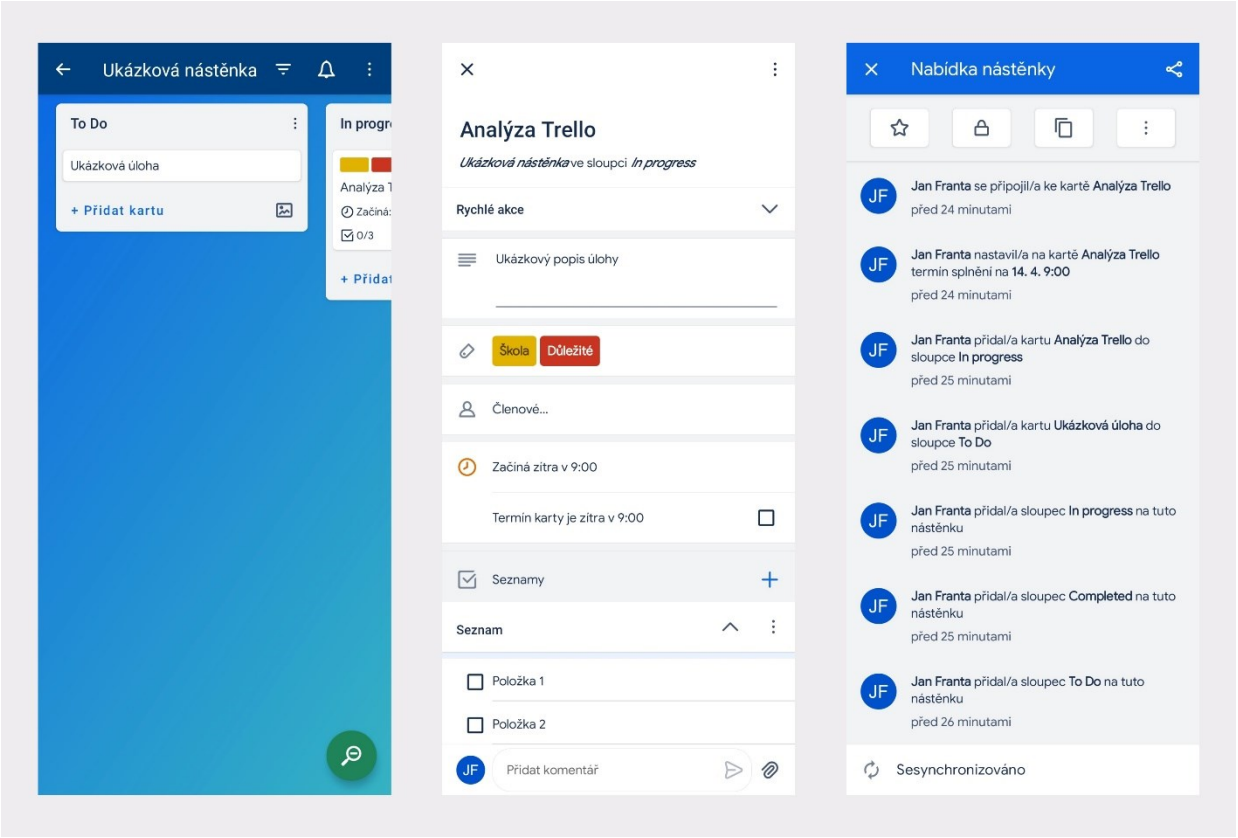
2.5 Trello

Aplikace Trello je vyvíjena společností Atlassian, která se zabývá aplikacemi pro správu projektů v kolaborativním prostředí a stojí například za velmi populární platformou pro řízení vývoje Jira. Platforma Trello je tak spíše zamýšlena pro týmy, a její rozhraní a způsob použití je tak lehce odlišný od ostatních analyzovaných aplikací. Přesto se Trello dá také použít i pro osobní účely jednotlivce [12].

Uživatelé v týmu jsou přiřazeni do takzvaného pracovního prostoru, ve kterém jsou umístěny nástěnky. Tyto nástěnky se dají přirovnat k seznamům v ostatních aplikacích, a mohou tak například sloužit pro ukládání různých kategorií úloh. Nástěnka je pak složena ze sloupců, do kterých se již ukládají samotné úlohy (nazývané karty) a slouží k jejich rozřazení a oddělení. Úlohy pak lze jednoduše mezi sloupci přesouvat.

Ukázkové použití nástěnky by mohlo zahrnovat 3 sloupce: *To Do*, *In progress* a *Completed*. Jednalo by se tak o nástěnku, která slouží jako přehled o úlohách, které je potřeba udělat, které jsou zrovna vykonávány a o těch, které již byly splněny.

Do sloupců se pak přidávají takzvané karty, které reprezentují již samotné úlohy. K úloze se dá uložit titulní obrázek a popis, označit ji štítky a přiřadit k ní členy (například ti, kteří mají danou úlohu za úkol). Dále lze vložit seznamy podúloh a nastavit její termín a připomenutí, které pak dostávají všichni členové této karty.



Obrázek 6. Rozhraní aplikace Trello (zdroj vlastní)

Aplikace Trello nabízí řešení pro správu zejména většího počtu úloh, které je přehledné díky rozřazení úloh do sloupců a díky možnosti úlohy označit a odlišit štítkem. K úloze lze také nastavit pouze jednu připomínku. Trello také nabízí takzvaná vylepšení, která rozšiřují možnosti a efektivitu nástěnek. Je dostupná celá řada těchto vylepšení, patří mezi ně například rozšířené poznámky nebo tabulky v rámci karet, integrace s poštovními klienty, cloudovými úložišti nebo platformou GitHub.

Do aplikace je nutné se přihlásit pod účtem, který následně zajišťuje online synchronizaci úloh. Trello je dostupné také ve formě webové aplikace, takže přístup k úlohám je možný prakticky z jakéhokoli zařízení.

Trello nabízí několik úrovní předplatného. Je dostupná verze zdarma, která je limitována maximálně 10 nástěnkami s maximálním počtem 10 účastníků, omezenou velikostí nahrávaných souborů a další. Dále je nabízena verze Standard, ve které je již neomezený počet nástěnek, vylepšené seznamy podúloh, které jsou zaměřeny na vyšší produktivitu v kolaborativním prostředí, více možností automatizace v nástěnkách a další. Cena této verze je 5 dolarů za uživatele za měsíc při předplacení na rok. Další dostupnou verzí je Premium, která nabízí vícero možností zobrazení nástěnky – zobrazení v rámci časové osy, kalendáře

a dalších. Také obsahuje prvky Atlassian Inteligence, pro generování obsahu, kontrolu gramatiky a brainstormingu řízeného umělou inteligencí. Tato verze je určena pro týmy, které potřebují pokročilejší nástroje ke správě svých projektů. Cena Premium verze je ve výši 10 dolarů za uživatele za měsíc. Poslední nabízenou verzí je Enterprise, která je mířena na velmi rozsáhlé týmy a organizace a nabízí neomezené množství pracovních prostorů nebo komplexnější nástroje pro administraci a zabezpečení, jako je například možnost nastavení práv v rámci celé organizace. Cena této verze závisí na celkovém množství uživatelů, pro 1000 uživatelů je odhadovaná cena 13 dolarů za uživatele za měsíc [13].

2.6 Shrnutí analýzy

V této sekci je uvedena stručná analýza aplikací, které se zaměřují na notifikace o nadcházejících úlohách. Jsou zde shrnuty klíčové vlastnosti vybraných aplikací a následně jsou mezi sebou porovnány z hlediska jejich obecných funkcí.

Any.do

Hlavní předností aplikace Any.do je její uživatelské rozhraní, které je přehledné a intuitivní, což zahrnuje také výběr z více možností zobrazování úloh dle preferencí uživatele. Unikátní funkcí mezi ostatními analyzovanými aplikacemi je možnost navrhnutí možných podúloh pomocí AI. Nevýhodou je možnost použití štítků, které je možné pouze v placené verzi.

TickTick

Aplikace TickTick je specifická především svými možnostmi přizpůsobení. Nabízí vytváření šablon nebo nastavení výchozích hodnot k úloze. Také nabízí zobrazení úloh na tzv. Eisenhowerové matici, která úlohy přehledně rozřazuje dle naléhavosti a důležitosti. Oproti ostatním aplikacím také TickTick nabízí široký výběr widgetů, které lze umístit na domovskou obrazovku. Také je možnost používat aplikaci i bez přihlášení, pokud uživatel nepotřebuje úlohy synchronizovat.

Taskito

Aplikace Taskito disponuje velmi přehledným uživatelským rozhraním. Umožňuje vytvářet šablony úloh, obdobně, jako tomu je u aplikace TickTick. Slabší stránkou je synchronizace úloh, která je oproti ostatním omezená pouze mezi nativními aplikacemi, jelikož Taskito nedisponuje webovou aplikací. Nicméně je možné aplikaci používat i bez přihlášení.

Microsoft To Do

Aplikace Microsoft To Do je jednou z aplikací v této kategorii, která nenabízí velké množství funkcí, ale za to může být pro některé uživatele jednodušší na použití než ostatní aplikace. Je dostupná pouze v jedné verzi, a veškeré její funkce jsou dostupné zdarma a bez omezení.

Trello

Aplikace Trello je přizpůsobena pro použití v kolaborativním prostředí a nenabízí takové množství funkcionalit, jako tomu je u ostatních aplikací, které jsou určeny spíše pro osobní účely. Hodí se pro správu velkých počtů úloh v rámci větších skupin uživatelů.

V následující tabulce byly aplikace porovnány z hlediska klíčových funkcionalit, které mohou být od aplikací tohoto typu očekávány. Aplikace jsou porovnávány z hlediska nabízených funkcionalit v rámci připomínek, možností organizace úloh, možnosti synchronizace nebo uživatelského přizpůsobení. Porovnání je provedeno také z hlediska ceny předplatného za rozšířenou verzi aplikací.

Funkce aplikace *pouze v premium	**Any.do**	**TickTick**	**Taskito**	**Microsoft To Do**	**Trello**
Opakované připomínky	✓	✓	✓	✓	✕
Možnost odložení připomínky	✓	✓	✓	✓	✕
Více připomínek k úloze	✕	✓ *	✓	✕	✕
Online synchronizace	✓	✓	✓	✓	✓
Oddělení úloh – seznamy	✓	✓	✓	✓	✓
Rozlišení úloh – štítky	✓ *	✓	✓	✕	✓
Přizpůsobení vzhledu aplikace	✓	✓	✓	✓	✓
Cena předplatného/měsíc	109 Kč	99 Kč	49 Kč	-	119 Kč

Tabulka 1. Souhrn a porovnání funkcí vybraných aplikací (zdroj vlastní)

2.7 Motivace pro vytvoření nové aplikace

Nová aplikace by měla poskytovat jednoduchou správu úloh s cílem realizovat rozřazení pomocí štítků a také s možností odkládat připomínky. Vzhled aplikace bude přehledný a intuitivní. Online synchronizace bude substituována možností exportu do souboru, kterou žádná z analyzovaných aplikací neposkytuje.

Vytvoření nové aplikace tohoto typu je především příležitostí z pohledu zlepšení dovedností ve vývoji mobilních aplikací, prohloubení znalostí programovacího jazyka Kotlin a dalších technologií spjatých s vývojem aplikací pro OS Android.

3 NÁVRH NOVÉ APLIKACE

V této sekci jsou uvedeny požadavky na novou aplikaci. Také jsou popsány aktuálně dostupné technologie pro vývoj mobilních aplikací a následně je proveden výběr technologií, které jsou použity k samotné implementaci nové aplikace.

3.1 Požadavky na novou aplikaci

Vývoj nové aplikace se řídí následujícími UML požadavky. UML požadavky specifikují charakteristiky aplikace a dělí se na 2 kategorie. Funkční požadavky definují chování a funkce systému. Nefunkční požadavky se zaměřují na popis obecných vlastností systému **Chyba! Nenalezen zdroj odkazů.**.

3.1.1 Funkční požadavky

FP01 **Hlavní funkce:** Aplikace umožní zadávat úkoly a bude připomínat jejich termíny.

FP02 **Organizace úkolů:** Aplikace uživateli umožní úkoly organizovat pomocí kategorií, které budou uživatelsky definovatelné.

FP03 **Podrobnosti úkolu:** K úkolu bude aplikace umožňovat ukládat nadpis, popis, jeho zařazení do kategorie, mezní termín splnění a čas připomenutí.

FP04 **Základní operace:** Úkol bude možné označit jako dokončený (nebo odznačit zpět na nedokončený) a bude možné ho i odstranit.

FP05 **Editace:** Aplikace bude umožňovat již existující úkoly upravovat.

FP06 **Filtrování:** Úkoly bude možné vyhledávat podle názvů a filtrovat je pomocí kategorií.

FP07 **Odložení notifikace:** Při přijetí připomínky bude uživateli nabídnuta možnost danou připomínku odložit.

FP08 **Perzistence:** Úkoly budou perzistentně uloženy v zařízení.

FP09 **Import a export dat:** Aplikace umožní import/export ze/do souboru.

3.1.2 Nefunkční požadavky

NP01 **Platforma:** Aplikace bude vyvíjena pro OS Android.

NP02 **Programovací jazyk:** Vývoj bude realizován v jazyce Kotlin.

NP03 **Architektura:** Zdrojový kód aplikace bude organizován dle architektury MVVM.

NP04 **Ukládání dat:** Úkoly budou ukládány do databáze Room.

NP05 **Import a export:** Import a export bude využívat soubory formátu JSON.

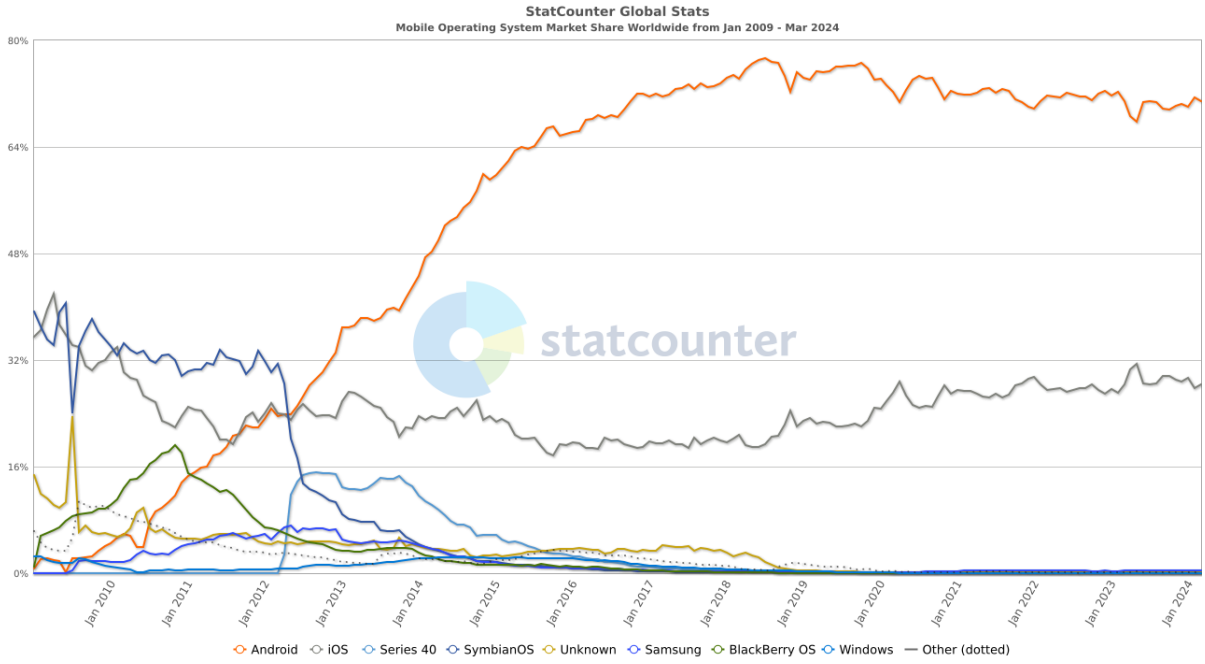
NP06 **Připomínky:** Připomínky budou zasílány ve formě systémových notifikací.

NP07 **Vzhled:** Vzhled aplikace bude používat koncepty designového jazyka Material You.

NP08 **Jednotkové testy:** Zdrojový kód bude pokrytý jednotkovými testy.

3.2 Platforma

První a zásadní rozhodnutí je o platformě, pro kterou bude aplikace vyvíjena. Nejpopulárnějším zastupitelem je systém Android, který má přibližně 71% podíl na celosvětovém trhu, další platformou je systém iOS s 28% podílem [15].



Obrázek 7. Podíl mobilních OS na globálním trhu (zdroj [15])

Nová aplikace bude vyvíjena pro platformu Android. Rozhodnutí bylo provedeno na základě předchozích zkušeností autora s vývojem pro tuto platformu a také kvůli tomu, že se jedná o nejpopulárnější a nejrozšířenější mobilní operační systém.

3.2.1 Android

Android je nejpopulárnější mobilní operační systém, který byl postaven nad modifikovanou verzí jádra operačního systému Linux. Jedná se o systém, který je navržen výhradně pro zařízení s dotykovou obrazovkou, jako jsou chytré telefony, tablety a chytré hodinky. K dispozici je ovšem také verze pro televize. Od roku 2007 je aktivně vyvíjen společností Google jako volně dostupný open-source projekt AOSP (Android Open Source Project) [16].

Většina Android zařízení má kromě samotného systému ještě předinstalovaný dodatečný software. Mezi nejznámější patří kolekce aplikací Google Mobile Services (GMS). Tato kolekce obsahuje různé aplikace, které integrují do systému služby od společnosti Google, mezi které patří například i platforma pro distribuci aplikací Google Play.

Výrobci také často instalují na svá zařízení takzvané nadstavby, které obsahují další předinstalované aplikace nebo služby a ve většině případů také modifikují vzhled a samotné chování systému [16][17].

3.3 Programovací jazyk a organizace kódu

Pro vývoj aplikací pro platformu Android je dostupných několik programovacích jazyků. Pro vývoj nativních Android aplikací se používají jazyky Kotlin, Java nebo C++. Dnes je preferovaným jazykem Kotlin, který byl v roce 2017 společností Google určen jako oficiální jazyk pro vývoj Android aplikací. Nahradil tak dosud používaný jazyk Java, se kterým je ovšem interoperabilní a dá se s ním kombinovat. Jazyk Java je ovšem stále velmi používaným jazykem v celé řadě dostupných aplikací. Jazyk C++ se používá při vývoji aplikací, které vyžadují vysoký výkon, zejména 3D her. Výběr programovacího jazyka pak závisí na výsledném použití aplikace [18][19].

Za programovací jazyk pro novou aplikaci byl zvolen Kotlin. A to z důvodu, že se jedná o doporučovaný a oficiální jazyk pro vývoj Android aplikací, a že charakter nové aplikace nevyžaduje nejvyšší výkon. Jako vývojové prostředí bude použito Android Studio.

Pro organizaci zdrojového kódu pomocí modulárního přístupu se při vývoji aplikací využívá softwarových architektonických vzorů (architektur). Pro vývoj Android aplikací se nejčastěji používají architektury MVC (Model-View-Controller), MVP (Model-View-Presenter) a MVVM (Model-View-ViewModel). Hlavní myšlenkou těchto architektur je oddělit části aplikace pro snadnější testování a údržbu kódu nebo přidávání a odebírání funkcí [20].

Zdrojový kód nové aplikace bude organizován dle architektury MVVM.

3.3.1 Kotlin

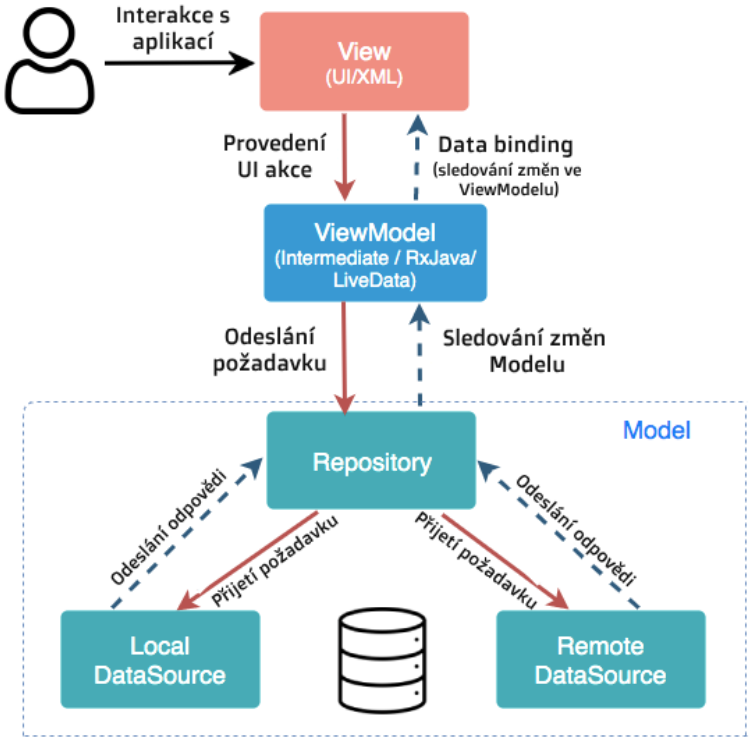
Kotlin je staticky typovaný, objektově orientovaný programovací jazyk, který je interoperabilní s Java knihovnami a s JVM (Java Virtual Machine). Je vyvíjen společností JetBrains a první verze vznikla v roce 2016. Jedná se o programovací jazyk, který je nejvíce využíván při vývoji aplikací pro systém Android.

Jeho myšlenkou bylo vylepšit jazyk Java a zároveň zachovat interoperabilitu mezi těmito jazyky tím, že se kompiluje do stejného bajtkódu. Další myšlenkou bylo zajistit bezpečné programování, proto má Kotlin funkci null safety, která eliminuje výjimky přístupu k null hodnotám. Kotlin také zjednodušuje syntaxi oproti jazyku Java, což má za následek lepší přehlednost kódu a zároveň šetří vývojářům čas [21].

3.3.2 Architektura MVVM

Model-View-ViewModel (MVVM) je softwarová architektura, která odděluje uživatelské rozhraní od samotné logiky aplikace. Tato architektura rozděluje aplikaci do třech odlišných vrstev, a to na Model, View a ViewModel, přičemž každá z těchto vrstev má na starost specifickou část funkcionality aplikace. Tento přístup vývojářům ulehčuje správu a údržbu zdrojového kódu aplikace, což přináší výhody jako je znovupoužitelnost částí kódu nebo snazší testovatelnost a škálovatelnost aplikace [22].

Architektura MVVM přebírá koncepty z architektur MVC (Model-View-Controller) a MVP (Model-View-Presenter) a snaží se překonat jejich nedostatky. Používá techniku data binding, která vytváří přímou vazbu mezi daty ve vrstvě ViewModel a prvky uživatelského rozhraní ve vrstvě View. Binding zaručuje, že je View automaticky notifikován na změny v datech a zobrazovaná data jsou tak dynamicky aktualizována [23].



Obrázek 8. Schéma architektury MVVM (zdroj [25])

Model

Vrstva Model zapouzdřuje veškerá data, se kterými aplikace pracuje. Reprezentuje zdroje dat, jako jsou databáze nebo webové služby. Má na starost správu dat aplikace a provádí operace s ní spojené – načítání, manipulace či ověřování dat. Komunikuje s vrstvou ViewModel, přes kterou následně zprostředkovaně komunikuje s vrstvou View [24].

View

Vrstva View se stará o uživatelské rozhraní aplikace. Reprezentuje jednotlivé obrazovky aplikace (aktivity), formuláře a další UI prvky. Zpracovává uživatelské vstupy a zobrazuje data, která získává z vrstvy ViewModel pomocí techniky data binding. Tato vrstva by zpravidla neměla obsahovat aplikační logiku a měla by být co nejjednodušší [24].

ViewModel

Vrstva ViewModel slouží jako mezičlánek pro komunikaci mezi vrstvami Model a View. Hlavním úkolem této vrstvy je dynamické zprostředkovávání dat vrstvě View. To je umožněno použitím techniky data binding, která v případě změny stavu aplikace vrstvu View notifikuje. ViewModel abstrahuje možnosti vrstvy Model a zpřístupňuje vrstvě View zjednodušené rozhraní pro komunikaci s vrstvou Model [24].

3.4 Ukládání dat

V mobilních aplikacích se dají data ukládat několika způsoby. Aplikace může data ukládat ve formě souborů, a to buď do svého specifického adresáře, který je čitelný pouze pro danou aplikaci, nebo do sdíleného adresáře, kde má přístup jakákoli aplikace. Další možností je využití privátní databáze, což je způsob pro uložení strukturovaných dat. Implementovaná aplikace bude svá data ukládat v rámci interní databáze Room [26].

3.4.1 Databáze Room

Room je knihovna pro perzistentní uložení dat a poskytuje abstraktní vrstvu nad databází SQLite. Je specificky designována pro použití v Android aplikacích. Poskytuje snadný a efektivní způsob ukládání, přístupu a správy dat a obecně zjednodušuje práci s databází SQLite. V online aplikacích je nejčastěji využívána jako cache pro zobrazování dat v případě nedostupnosti sítě [27][28].

SQLite je samostatný bezserverový databázový stroj, který používá standardizovaný dotazovací jazyk SQL pro manipulaci s daty. Jedná se o nenáročné a efektivní úložiště, které je vhodné pro malé až středně velké databáze [27].

Zjednodušení práce s databází SQLite spočívá ve využití ORM (object-relational mapping) přístupu. Tabulky v databázi jsou mapovány na objekty programovacího jazyka, což vývojářům umožňuje pracovat s databází použitím technik z objektově orientovaného programování. Také provádí kontrolu SQL dotazů, což chrání aplikaci před neočekávaným chováním [27][29].

3.5 Vzhled aplikace

Pro tvorbu vzhledu uživatelského rozhraní aplikace byl použit značkovací jazyk XML. Ten je výchozím jazykem v editoru uživatelského rozhraní, který je součástí použitého vývojového prostředí Android Studio. Vzhled aplikace dále používá komponenty a principy designového jazyka Material, konkrétně jeho nejnovější verze, Material 3 (také nazývané Material You).

Aplikace je rozdělena do tzv. aktivit. Aktivita reprezentuje jednotlivé obrazovky aplikace, ve kterých je vykreslováno uživatelské rozhraní. Ke každé aktivitě je tak specifikován XML soubor, který popisuje vzhled a rozložení jejího uživatelského rozhraní [30].

3.5.1 XML

Značkovací jazyk XML (Extensible Markup Language) je používán pro popis strukturovaných dat. XML dokumenty se skládají z tzv. elementů, které jsou označeny tzv. tagem. Uvnitř těchto elementů mohou být další elementy nebo samotná data. Pomocí jazyka XML je možnost uložit strukturovaná data v přehledném a snadno čitelném formátu. Z pohledu formátování je však striktní a v případě chyby je celý dokument neplatný [31].

3.5.2 Material You

Material You (také zvaný Material 3) je designový jazyk od společnosti Google, který byl poprvé představen v systému Android 12. Tento design se především soustředí na dynamické barvy systému, které jsou vybrány na základě nastavené tapety zařízení. Systém vygeneruje celkově 5 různých barevných palet. Uživatel si jednu z palet vybere a barvy z této palety jsou pak aplikovány na různé části uživatelského rozhraní. Dalšími prvky jsou konzistentní animace pohybů po systému, jako je rolování seznamů nebo sjednocení vzhledu widgetů [32].

3.6 Popis implementované aplikace

Nová aplikace slouží jako jednoduchý správce úkolů. Uživateli umožní zadávat různorodé úkoly, u kterých zaznamenává název události, volitelně také popis a následně mezní termín pro splnění. Úkoly je možné rozlišovat pomocí barevných štítků. Také je umožněno nastavení připomenutí daného úkolu před zadaným termínem.

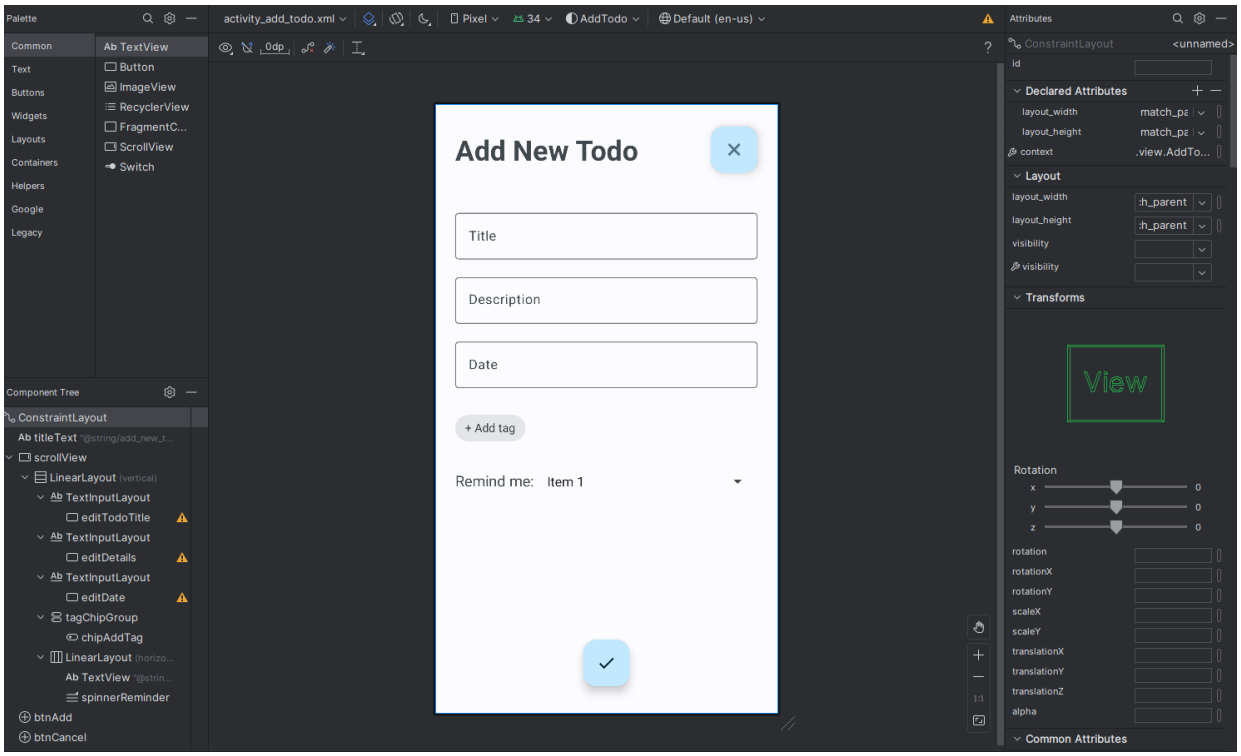
Aplikace je vyvíjena pro operační systém Android a je realizována v programovacím jazyce Kotlin. K organizaci zdrojového kódu je využita architektura MVVM. O perzistentní uložení úkolů se stará databáze Room a aplikace umožňuje data exportovat do souboru. Vzhled aplikace využívá prvky Material You designu a aplikace podporuje zobrazení v tmavém režimu.

4 IMPLEMENTACE NOVÉ APLIKACE

4.1 Vývojové prostředí

K vývoji aplikace je použito vývojové prostředí Android Studio ve verzi Iguana 2023.2.1. Toto IDE nabízí veškeré potřebné nástroje k vývoji aplikací pro operační systém Android, a to buď v programovacím jazyce Kotlin nebo Java.

Jeden z nejužitečnějších nástrojů je vbudovaný emulátor zařízení s OS Android, na kterém se vyvíjená aplikace dá spustit a testovat její chování. Další užitečný nástroj je editor pro vytváření uživatelského rozhraní, který pomocí intuitivního vizuálního prostředí vytváří XML soubor popisující vzhled dané aktivity.

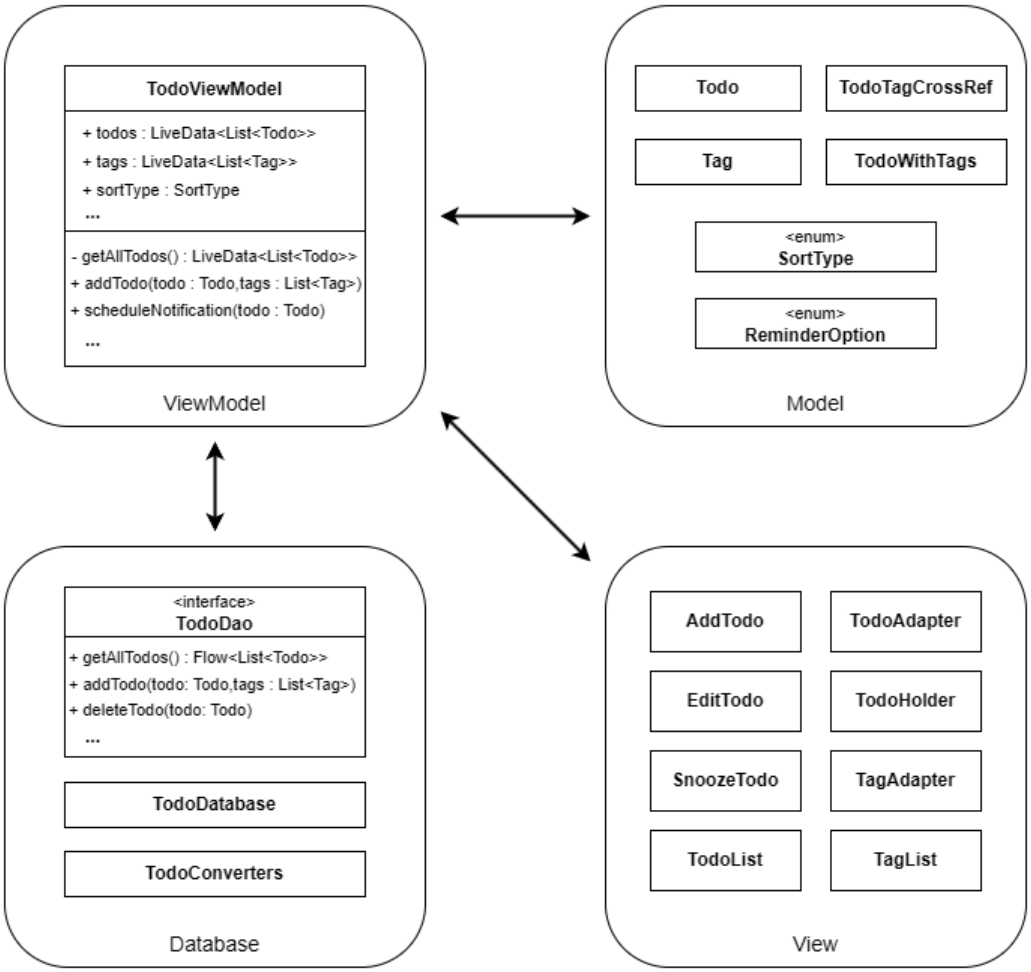


Obrázek 9. Editor GUI aktivity v prostředí Android Studio (zdroj vlastní)

4.2 Organizace zdrojového kódu

Zdrojový kód aplikace je organizován podle architektury MVVM. Je tedy rozdělen do následujících balíčků:

- **Model** – obsahuje třídy a výčty (enums) definující data použité v aplikaci
- **ViewModel** – obsahuje třídu, která obstarává práci s databází a ostatní logiku aplikace
- **View** – obsahuje třídy, které specifikují funkcionalitu jednotlivých aktivit
- **Database** – obsahuje třídy a rozhraní, které zpřístupňuje metody k práci s databází



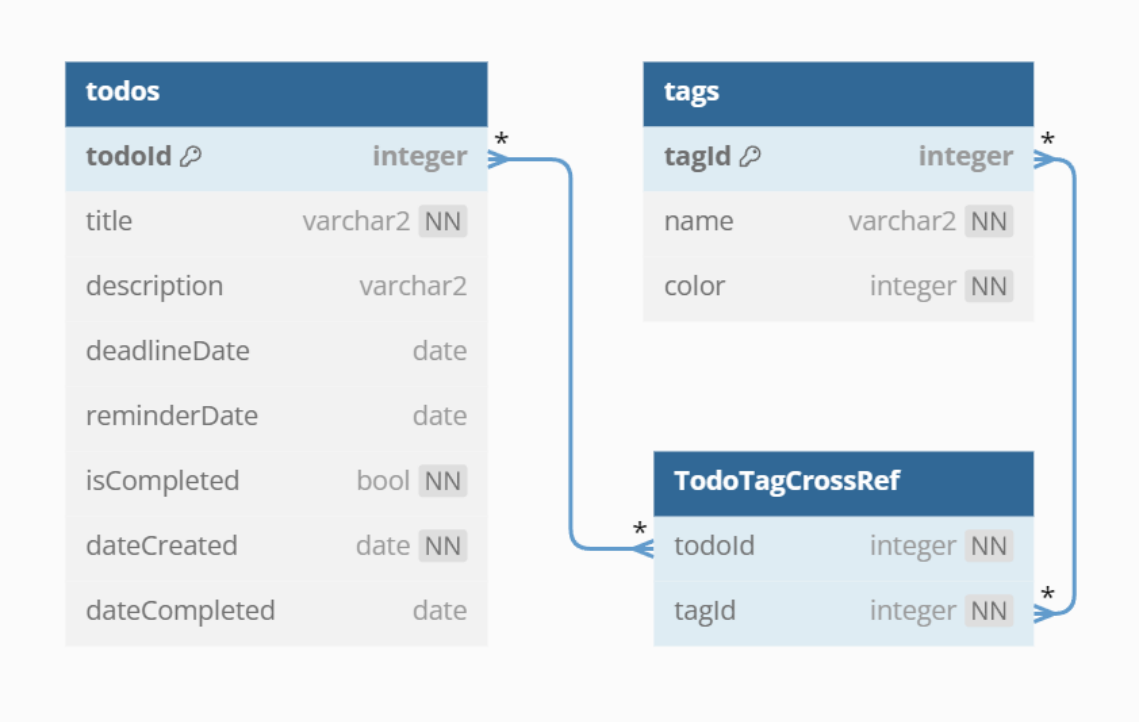
Obrázek 10. Zjednodušený diagram balíčků aplikace (zdroj vlastní)

4.3 Schéma databáze

Databáze je v aplikaci využita pro perzistentní uložení dat o úlohách a štítcích. Přístup k databázi je třídě ViewModel zprostředkován pomocí DAO (Data Access Object) třídy. Tato třída poskytuje abstraktní metody pro přístup a práci s daty uloženými v databázi. Při kompilaci je pak automaticky vygenerována samotná implementace této třídy [33]. Přístup k TodoDao je zprostředkován pomocí třídy TodoDatabase, která databázi inicializuje a zpřístupní automaticky implementovanou třídu TodoDao, která je předána ViewModelu.

Databáze obsahuje celkem 3 tabulky:

- Tabulka *todos* obsahuje informace o úloze, jako je název, popisek, stav splnění, data termínu, připomínky, vytvoření a splnění.
- Tabulka *tags* obsahuje informace o štítku, jako je jeho název a barva.
- Tabulka *TodoTagCrossRef* je propojovací tabulkou pro zachycení reference mezi úlohou a jejími štítky.



Obrázek 11. Schéma databáze (zdroj vlastní)

Mezi tabulkami *todos*, *tags* a tabulkou *TodoTagCrossRef* je relace typu M:N, jelikož každá úloha (todo) může mít více štítků (tag) a zároveň každý štítek může být přiřazen vícero úlohám.

Z důvodu odlišností ve formátech ukládání datumů mezi databází a jazykem Kotlin jsou ve třídě TodoConverters zavedeny převodníky, které umožní objekty *Date* ukládat ve formátu používaném databází a při načtení je převádět zpět na objekt *Date*.

4.3.1 Použití anotací

Při použití databáze Room je vhodné ukládané objekty definovat jako tzv. entity. Definice entity se provádí použitím anotace *@Entity*, díky které je pak automaticky definováno také schéma tabulky tím, že jsou atributy entity namapovány na sloupce tabulky.

Při definici pak mohou být využívány další anotace, jako je *@PrimaryKey*, která z daného atributu vytvoří primární klíč výsledné tabulky. Mohou být také definovány sloupce pro vytvořené indexy a další vlastnosti [34].

V případě entity *TodoTagCrossRef* jsou zde definovány také cizí klíče do tabulek *todos* a *tags*, včetně strategie při odstraňování záznamů, kde je definováno kaskádové mazání. To znamená, že při odstranění štítku jsou všechny vazby štítku s úlohami odstraněny (úlohy pak již nemají daný štítek přiřazen). A při odstranění úlohy jsou všechny její vazby na štítky také z průnikové tabulky odstraněny.

```
@Entity (
    tableName = "todos",
    indices = [Index(value = ["title"], unique = false)]
)
data class Todo(
    @PrimaryKey(autoGenerate = true) val todoId: Int = 0,
    var title : String,
    var description : String?,
    var deadlineDate : Date?,
    var reminderDate : Date?,
    var isCompleted : Boolean,
    val dateCreated : Date,
    var dateCompleted : Date?
)
```

Zdrojový kód 1. Anotace třídy Todo jako entity (zdroj vlastní)

Anotace jsou také použity v abstraktní třídě TodoDao. Anotace zde slouží pro označení typů metod, pomocí kterých může pak být automaticky vygenerovaná implementace SQL dotazů. Pro jednoduché operace lze použít anotace *@Insert*, *@Update*, *@Delete* a pro složitější lze k metodě přiřadit SQL dotaz pomocí anotace *@Query*. Pro zajištění konzistence databáze při vykonávání více příkazů v metodě je lze provést v transakci, pomocí anotace *@Transaction*.

```
@Dao
interface TodoDao {
    @Insert(onConflict = ABORT)
    fun addTodo(todo : Todo)

    @Transaction
    suspend fun addTodoWithTags(todo: Todo, tags: List<Tag>) {
        val todoId = addTodoAndGetId(todo).toInt()

        tags.forEach { tag ->
            val crossRef = TodoTagCrossRef(todoId = todoId, tagId =
tag.tagId)
            addTodoTagCrossRef(crossRef)
        }
    }

    @Update(onConflict = ABORT)
    fun updateTodo(todo: Todo)

    @Delete
    fun deleteTodo(todo: Todo)

    @Query("SELECT * FROM todos WHERE todoId = :todoId")
    fun getTodoById(todoId: Int): Flow<Todo?>

    @Query("SELECT * FROM todos")
    fun getAllTodos() : Flow<List<Todo>>

    @Insert(onConflict = ABORT)
    fun addTag(tag: Tag)
```

Zdrojový kód 2. Možnosti anotace DAO třídy a metod (zdroj vlastní)

4.4 Implementace funkcí

4.4.1 Zobrazování úloh

Úlohy jsou zobrazovány v hlavní aktivitě aplikace TodoList. Pro zobrazování je využita komponenta RecyclerView, která poskytuje efektivní zobrazení větších datových sad. Pro prvky seznamu je nutné definovat vzhled, pomocí tzv. adaptéru a holderu, a RecyclerView pak tyto prvky dynamicky vytváří, když je třeba. Výhodou je, že jsou zpracovávány pouze ty prvky, které jsou zrovna viditelné na obrazovce. Pro nové prvky stejného typu je použit již vytvořený adaptér a holder, což vede ke zlepšení výkonu a responzivity aplikace [35].

Úlohy jsou seznamu RecyclerView předávány ze třídy TodoViewModel. Ta úlohy získává pomocí třídy TodoDao jako datový typ *LiveData*, což je typ dat, který umožňuje sledovat změny (observe) ve zdrojovém datasetu a upozorňovat na tyto změny své tzv. observery.

```
// TodoList
private val todoDao by lazy { TodoDatabase.getDatabase(this).todoDao() }
todoViewModel = TodoViewModel(todoDao, this) // předání TodoDao

binding.todosRecyclerView.layoutManager = LinearLayoutManager(this)
todoAdapter = TodoAdapter(todoViewModel)
binding.todosRecyclerView.adapter = todoAdapter

todoViewModel.todos.observe(this) { todos ->
    todoAdapter.submitList(todos)
}

// TodoViewModel
val todos: LiveData<List<TodoWithTags>> get() = _todos

_todos.addSource(getAllTodos()) {
    val sortedTodos = it.sortedWith(getTodoComparator())
    _todos.value = sortedTodos
}
```

Zdrojový kód 3. Předávání úloh do aktivity TodoList pomocí datového typu LiveData (zdroj vlastní)

Jak již bylo zmíněno, pro RecyclerView jsou důležité třídy adapter a holder. Ve třídě TodoAdapter je specifikován tzv. DiffCallback, což je objekt, který seznamu pomáhá určit, které prvky seznamu byly změněny. Aby byl tento proces účinný, je nutné, aby měl objekt seznamu implementované metody *equals()* a *hashCode()*. Při aktualizaci seznamu z observeru je proveden výpočet rozdílů původního a nového seznamu, a jsou aktualizovány pouze změněné prvky. Je zde také realizován filtr, který umožňuje úlohy filtrovat dle štítků či v nich vyhledávat.

Třída TodoHolder slouží k realizaci vzhledu a funkcionality dané položky v seznamu. Jsou zde nastavována zobrazovaná pole textů, aby reflektovala údaje zobrazované úlohy a také jsou přiřazeny akce tlačítek pro označení dokončené, smazání či dialogu detailu úlohy.

```
class TodoAdapter(private val todoViewModel: TodoViewModel) :
ListAdapter<TodoWithTags, TodoHolder>(DiffCallback),Filterable {
    companion object {
        private val DiffCallback = object :
DiffUtil.ItemCallback<TodoWithTags>() {
            override fun areItemsTheSame(oldItem: TodoWithTags, newItem:
TodoWithTags): Boolean {
                return oldItem.todo.todoId == newItem.todo.todoId
            }

            override fun areContentsTheSame(oldItem: TodoWithTags, newItem:
TodoWithTags): Boolean {
                return oldItem == newItem
            }
        }
    }
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
TodoHolder {
        return
TodoHolder(TodoTileBinding.inflate(LayoutInflater.from(parent.context),
parent, false), todoViewModel, parent.context)
    }
    override fun onBindViewHolder(holder: TodoHolder, position: Int) {
        holder.bind(getItem(position))
    }
// TodoHolder
fun bind(todoWithTags: TodoWithTags) {
    val todo = todoWithTags.todo
    binding.root.setOnClickListener {
        showTodoDetailsDialog(todo)
    }

    binding.checkBoxCompleted.isChecked = todo.isCompleted
    updateTileStyle(todo.isCompleted)

    binding.todoTitleText.text = todo.title
```

Zdrojový kód 4. Třídy TodoAdapter (objekt DiffCallback) a TodoHolder

4.4.2 Přidávání a editace úloh

Pro přidávání i editaci úloh jsou vytvořeny prakticky totožné aktivity. Pro přidání je vytvořena aktivita AddTodo, která obsahuje textová pole pro název, popis, výběr termínu, přidání štítků a možnost nastavit připomínku. Rozdíl v aktivitě pro úpravu, EditTodo, je, že se při inicializaci aktivity do polí načtou informace k úloze. To je vyřešeno předáním todoId při spouštění této aktivity pomocí příkazu *putExtra()*. Aktivita pak má k tomuto údaji přístup pomocí příkazu *getIntExtra()*, a může tak podle ID z databáze údaje úlohy načíst.

Výběr termínu úlohy je realizován pomocí komponenty *MaterialDatePicker* pro výběr data a pomocí komponenty *MaterialTimePicker* pro výběr konkrétního času. Vybraný termín je ukládán v objektu *Calendar*, který umožňuje komplexnější práci s časovými údaji a je využit například pro nastavení výchozího času termínu o 5 minut později, než je aktuální čas.

```
private fun showDatePicker() {
    val constraintsBuilder = CalendarConstraints.Builder()
        .setValidator(DateValidatorPointForward.now())
    val builder = MaterialDatePicker.Builder.datePicker()
        .setTitleText("Select Date")
        .setSelection(calendar.timeInMillis)
        .setCalendarConstraints(constraintsBuilder.build())
    val datePicker = builder.build()
    datePicker.addOnPositiveButtonClickListener { dateInMillis ->
        calendar.timeInMillis = dateInMillis
        showTimePicker()
    }
    datePicker.show(supportFragmentManager, "datePicker")
}

private fun saveTodo() {
    val title = binding.editTodoTitle.text.toString()
    var description : String? = binding.editDetails.text.toString().trim()
    if (description.isNullOrBlank()) description = null
    val deadlineDate = calendar.time
    val dateCreated = Date()
    val todo = Todo(title = title,description = description,deadlineDate =
deadlineDate,reminderDate = reminderTime,isCompleted = false,dateCreated =
dateCreated,dateCompleted = null)
    todoViewModel.addTodo(todo, selectedTagsList)
    todo.reminderDate?.let { todoViewModel.scheduleNotification(todo) }
}
```

Zdrojový kód 5. DialogMaterialDatePicker a ukládání úlohy v aktivitě AddTodo (zdroj vlastní)

4.4.3 Štítky

Pro správu štítků slouží aktivita TagList, ve které je možné štítky přidávat, upravovat a mazat. Pro zobrazování štítků, jak v této aktivitě, tak i v ostatních aktivitách, se používá komponenta *ChipGroup* a jednotlivé štítky jsou pak reprezentovány komponenty *Chip*. V dialozích pro výběr štítků jsou štítky zobrazeny v RecyclerView a je pro ně vytvořen TagAdapter. Pro výběr barvy štítků je použita knihovna ColorPicker (zdroj [36]), která poskytuje dialogy pro výběr z předdefinovaných barev. Konkrétně je použita komponenta *MaterialColorPickerDialog*.

```
val chip = LayoutInflater.from(this).inflate(R.layout.chip_layout_normal,
binding.tagChipGroup, false) as Chip
chip.text = tag.name
chip.chipBackgroundColor = ColorStateList.valueOf(tag.color)
val textColor = if (todoViewModel.isColorDark(tag.color)) Color.WHITE else
Color.BLACK
chip.setTextColor(textColor)
chip.isClickable = true
chip.isCloseIconVisible = true
chip.setOnCloseIconClickListener {
    selectedTagsList.remove(tag)
    binding.tagChipGroup.removeView(chip)
}
binding.tagChipGroup.addView(chip)
// ColorPicker knihovna (build.gradle.kts - dependencies)
implementation ("com.github.Dhaval2404:ColorPicker:2.3")
```

Zdrojový kód 6. Použití komponenty Chip pro reprezentaci štítku, knihovna ColorPicker (zdroj vlastní)

4.4.4 Nastavování připomínek

Připomínky jsou realizovány pomocí systémových služeb *Notification Service* a *Alarm Service*. Služba *Notification Service* slouží k samotnému odesílání systémových oznámení a služba *Alarm Service* provádí plánování notifikací a odesílá je třídě pro zpracování, NotificationReceiver, která při přijetí zprávy vytvoří samotný vzhled notifikace a odešle ji. Každá notifikace musí být přiřazena k notifikačnímu kanálu, jinak se nezobrazí [37].

```
private fun createNotificationChannel() {
    val channel = NotificationChannel("todo_reminders","Todo Reminders",
        NotificationManager.IMPORTANCE_DEFAULT)
    val notificationManager = getSystemService(NOTIFICATION_SERVICE) as
NotificationManager
    notificationManager.createNotificationChannel(channel)
}
fun scheduleNotification(todo: Todo) {
    val context = getContext() ?: return
    todo.reminderDate ?: return
    if (todo.isCompleted) return
    val alarmManager = context.getSystemService(Context.ALARM_SERVICE) as
AlarmManager
    val alarmIntent = Intent(context,
NotificationReceiver::class.java).apply {
        action = "TODO_REMINDER"
        putExtra("TODO_ID", todo.todoId)
        putExtra("TODO_TITLE", todo.title)
    }.let {
PendingIntent.getBroadcast(context,todo.todoId,it,PendingIntent.FLAG_IMMUTA
BLE) }
    val reminderTime = todo.reminderDate!!.time
    alarmManager.set(AlarmManager.RTC_WAKEUP, reminderTime, alarmIntent)
}
// NotificationReceiver
val todoId = intent.getIntExtra("TODO_ID", -1)
val todoTitle = intent.getStringExtra("TODO_TITLE")
val builder = NotificationCompat.Builder(context, "todo_reminders")
    .setSmallIcon(R.drawable.baseline_check_24)
    .setContentTitle("Todo Reminder")
    .setContentText("Don't forget: $todoTitle")
    .setPriority(NotificationCompat.PRIORITY_DEFAULT)
    .setCategory(NotificationCompat.CATEGORY_REMINDER)
    .addAction(0, "Done", markCompletePendingIntent)
    .addAction(0, "Snooze", snoozePendingIntent)
    .setContentIntent(openPendingIntent)
    .setAutoCancel(true)

notificationManager.notify(todoId, builder.build())
```

Zdrojový kód 7. Vytvoření kanálu oznámení aplikace, vytvoření a zpracování notifikace (zdroj vlastní)

4.4.5 Import a export

Aplikace umožňuje uložená data exportovat do souboru formátu JSON. Formát JSON má syntaxi podobnou jazyku JavaScript a umožňuje serializovat data jednoduchých datových typů, jako jsou řetězce a čísla, ale také složitějších, jako jsou objekty a pole hodnot [38].

Pro samotnou serializaci dat je použita knihovna Gson od společnosti Google, která umožňuje serializovat Java/Kotlin objekty do formátu JSON a zpět [39].

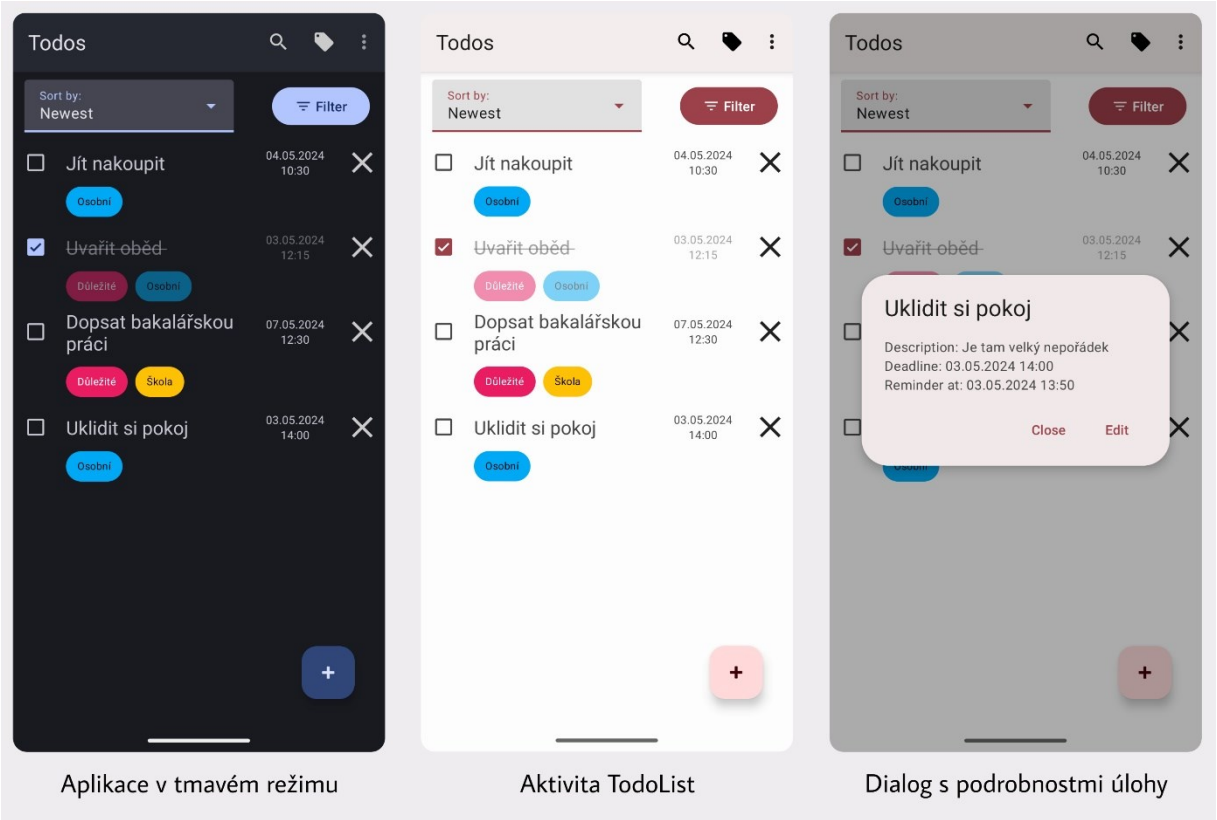
```
// Export
contentResolver.openFileDescriptor(uri, "w")?.use { parcelFileDescriptor ->
    FileWriter(parcelFileDescriptor.fileDescriptor).use { writer ->
        val exportData = hashMapOf<String, Any>()

        exportData["todos"] = todos
        exportData["tags"] = tags
        exportData["todoTagRelations"] = todoTags

        val jsonExportData = Gson().toJson(exportData)
        writer.write(jsonExportData)
    }
}
// Import
contentResolver.openFileDescriptor(uri, "r")?.use { parcelFileDescriptor ->
    BufferedReader(FileReader(parcelFileDescriptor.fileDescriptor)).use {
reader ->
        val jsonString = StringBuilder()
        var line: String?
        while (reader.readLine().also { line = it } != null) {
            jsonString.append(line)
        }
        val gson = Gson()
        val importData: Map<String, Any> =
gson.fromJson(jsonString.toString(), object : TypeToken<Map<String, Any>>()
{}.type)
        val todos: List<Todo> =
gson.fromJson(gson.toJsonTree(importData["todos"]).asJsonArray, object :
TypeToken<List<Todo>>() {}.type)
        addTodos(todos)
        val tags: List<Tag> =
gson.fromJson(gson.toJsonTree(importData["tags"]).asJsonArray, object :
TypeToken<List<Tag>>() {}.type)
        addTags(tags)
        val todoTags: List<TodoTagCrossRef> =
gson.fromJson(gson.toJsonTree(importData["todoTagRelations"]).asJsonArray,
object : TypeToken<List<TodoTagCrossRef>>() {}.type)
        addTodoTags(todoTags)
    }
}
```

Zdrojový kód 8. Import a export dat aplikace do souboru JSON (zdroj vlastní)

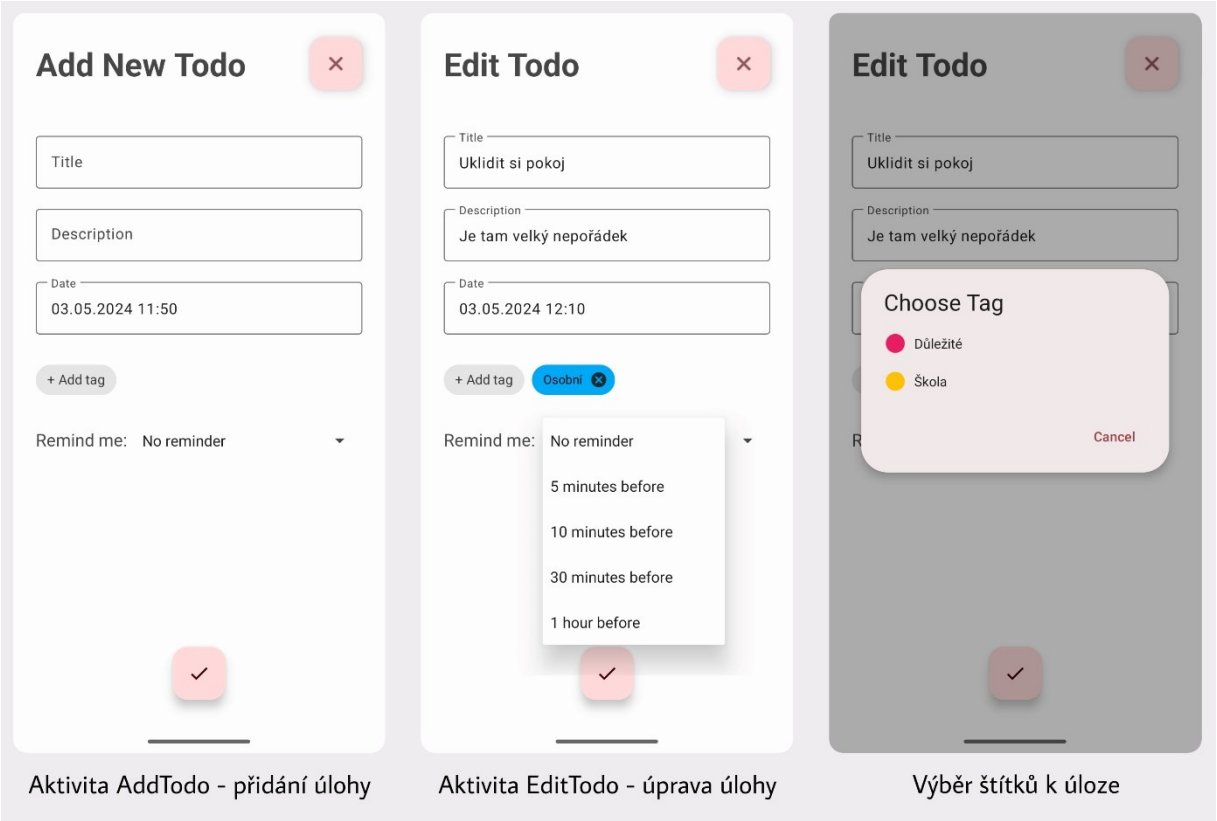
4.5 Uživatelské prostředí

V této části jsou předvedeny jednotlivé obrazovky výsledné aplikace:

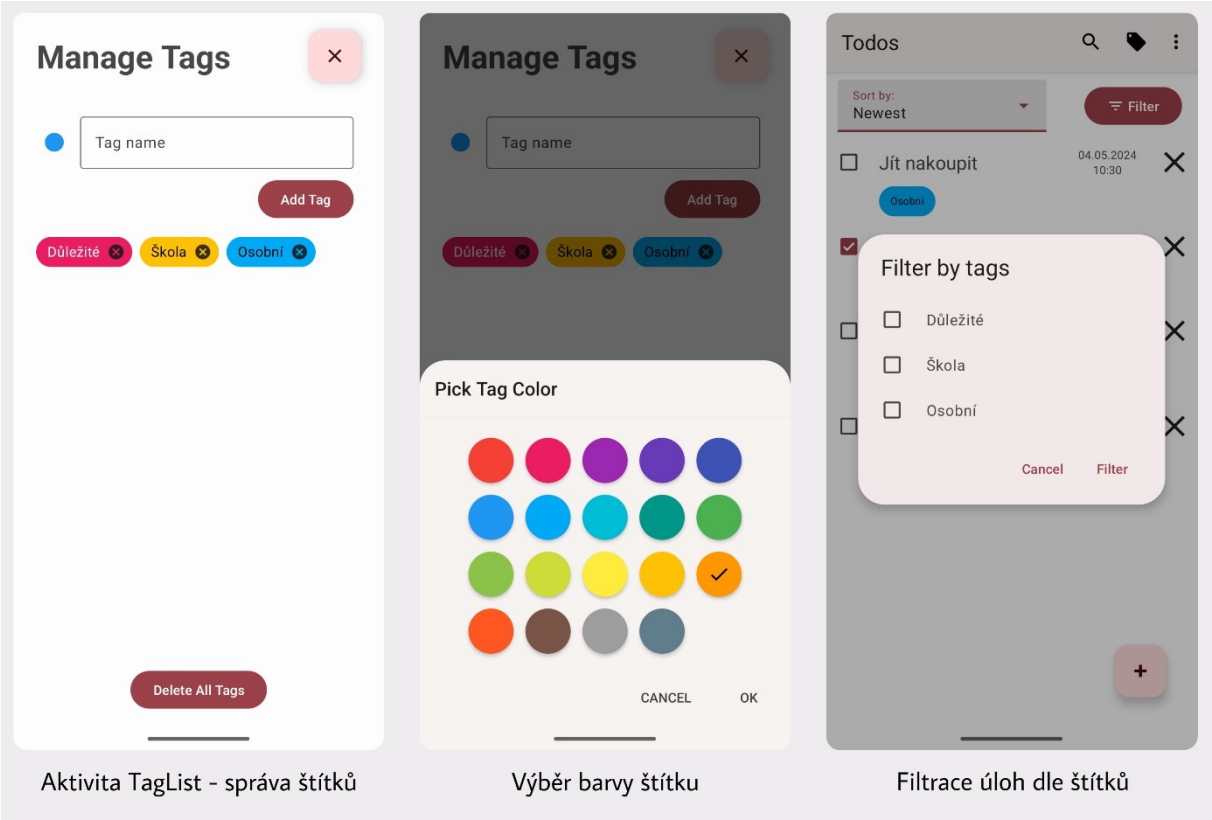
- Aktivita TodoList, která zobrazuje seznam úloh a umožňuje jejich filtraci.
- Aktivity AddTodo a EditTodo, které slouží k přidávání a úpravě úloh.
- Aktivita TagList, která slouží ke správě štítků.
- Přijetí připomínky ve formě notifikace a její odložení.



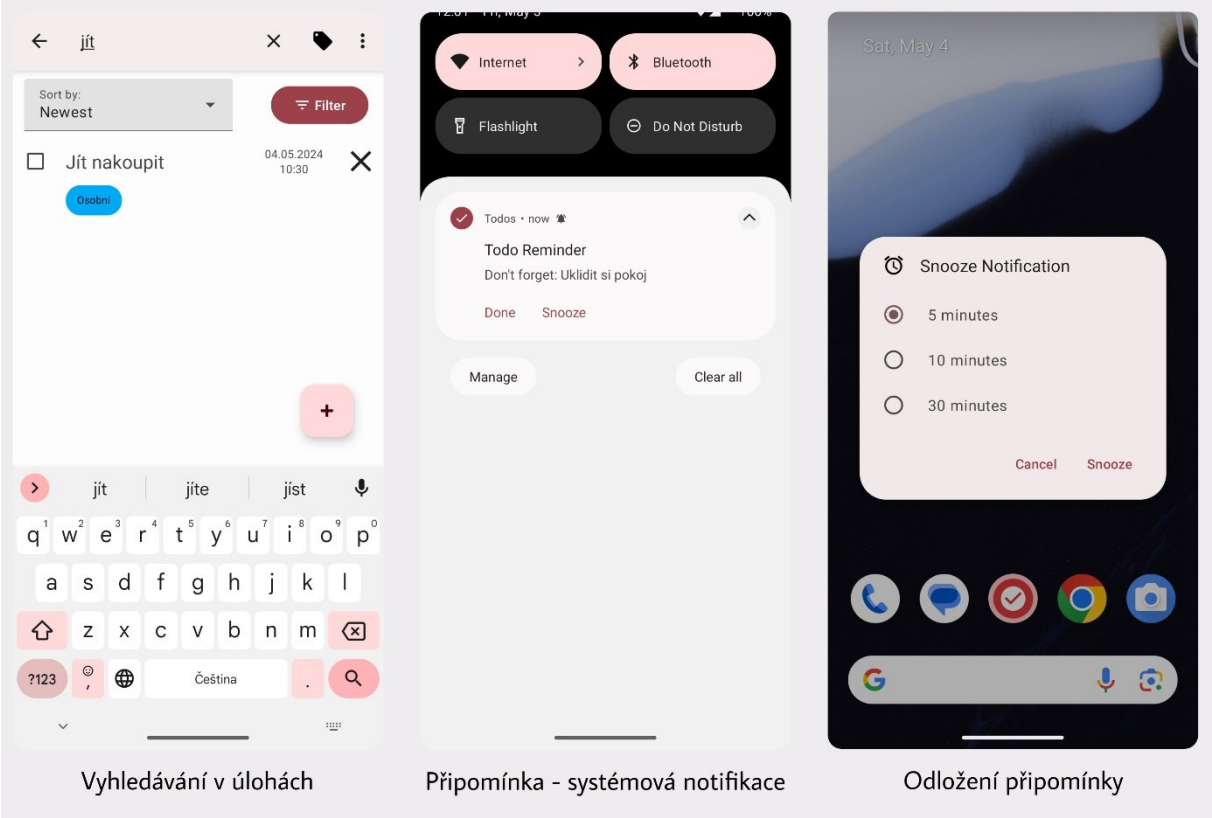
Obrázek 12. Obrazovky vlastní aplikace – tmavý režim, aktivita TodoList, dialog podrobností úlohy (zdroj vlastní)



Obrázek 13. Obrazovky vlastní aplikace – aktivity AddTodo a EditTodo, výběr štítků (zdroj vlastní)



Obrázek 14. Obrazovky vlastní aplikace – použití štítků v aplikaci (zdroj vlastní)



Obrázek 15. Obrazovky vlastní aplikace – vyhledávání, notifikace a odložení připomínky (zdroj vlastní)

ZÁVĚR

Cílem této bakalářské práce bylo analyzovat aktuálně dostupné mobilní aplikace pro OS Android, jejichž úkolem je uživateli umožnit správu úloh s důležitými termíny a následně jej na tyto termíny notifikovat. Vybrané aplikace byly v rámci analýzy popsány a následně byly mezi sebou srovnány na základě jejich klíčových funkcí a vlastností.

Další část práce byla zaměřena na implementaci vlastní aplikace pro notifikaci důležitých termínů. V této části byly objasněny použité technologie, jako je architektura MVVM, databáze Room a designový jazyk Material You. Také byly popsány principy realizace určitých částí a funkcionalit aplikace.

Výsledná mobilní aplikace slouží jako jednoduchý správce nadcházejících úloh, který umožňuje uživatele notifikovat na důležité termíny, úlohy vyhledávat podle jejich názvu nebo je třídit pomocí štítků. Aplikace také umožňuje veškerá data pohodlně zálohovat a obnovovat pomocí JSON souborů.

POUŽITÁ LITERATURA

[1] TERRELL HANNA, Katie a Ivy WIGMORE. Mobile App. In: *Techtarget* [online]. 2023 [cit. 2024-03-14]. Dostupné z: https://www.techtarget.com/whatis/definition/mobile-app

[2] CIESLAR, Jan. V Česku používá chytré telefony již 82 % osob. In: *Český statistický úřad* [online]. 2023 [cit. 2024-03-14]. Dostupné z: https://www.czso.cz/csu/czso/v-cesku-pouziva-chytre-telefony-jiz-82-osob

[3] Druhy mobilních aplikací. In: *Creative Handles* [online]. [cit. 2024-03-14]. Dostupné z: https://creativehandles.com/cs/blogove-prispevky/125/druhy-mobilnich-aplikaci-vyhody-nevyhody-a-jak-vybrat

[4] SCHMITT, Jacob. Native vs cross-platform mobile app development. In: *Circleci* [online]. 2023 [cit. 2024-03-15]. Dostupné z: https://circleci.com/blog/native-vs-cross-platform-mobile-dev/

[5] Distribution of mobile applications. In: *LinkedIn* [online]. [cit. 2024-03-14]. Dostupné z: https://www.linkedin.com/advice/0/how-do-you-distribute-update-mobile-applications

[6] SNYDER, Joel. What are the risks of sideloaded Android apps. In: *Samsung Insights* [online]. 2021 [cit. 2024-03-14]. Dostupné z: https://insights.samsung.com/2021/07/14/what-are-the-risks-of-sideloaded-android-apps/

[7] Alternative distribution options. In: *Android Developer* [online]. 2020 [cit. 2024-03-14]. Dostupné z: https://developer.android.com/distribute/marketing-tools/alternative-distribution

[8] GUINNESS, Harry. The 7 best to do list apps in 2024. In: *Zapier* [online]. 2023 [cit. 2024-04-04]. Dostupné z: https://zapier.com/blog/best-todo-list-apps/

[9] Any.do Pricing. Any.do [online]. [cit. 2024-04-04]. Dostupné z: https://www.any.do/pricing

[10] TickTick Premium. In: *TickTick* [online]. [cit. 2024-04-10]. Dostupné z: https://ticktick.com/upgrade

[11] Taskito: Press kit. In: *Taskito* [online]. [cit. 2024-04-13]. Dostupné z: https://taskito.io/press-kit

[12] Atlassian Products. In: *Atlassian* [online]. [cit. 2024-04-15]. Dostupné z: https://www.atlassian.com/software

[13] Trello Pricing. In: *Trello* [online]. [cit. 2024-04-15]. Dostupné z: https://trello.com/pricing

[14] Functional and Nonfunctional Requirements: Specification and Types. In: *Altexsoft* [online]. 2023 [cit. 2024-04-10]. Dostupné z: https://www.altexsoft.com/blog/functional-and-non-functional-requirements-specification-and-types/

[15] Mobile Operating System Market Share Worldwide. *Statcounter* [online]. 2024 [cit. 2024-03-15]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide

[16] Android Operating System. In: *JavaTpoint* [online]. [cit. 2024-03-16]. Dostupné z: https://www.javatpoint.com/android-operating-system

[17] Android - Google Mobile Services. In: *Android* [online]. [cit. 2024-03-16]. Dostupné z: https://www.android.com/gms/

[18] PETRECOLLA, Diego. Android App Development: Which Language to Choose. In: *Codemotion* [online]. 2023 [cit. 2024-03-16]. Dostupné z: https://www.codemotion.com/magazine/frontend/mobile-dev/android-app-development-which-language-to-choose/

[19] Android App Development: Which Language to Choose. In: *GeeksforGeeks* [online]. 2022 [cit. 2024-03-16]. Dostupné z: https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/

[20] Android Architecture Patterns. In: *GeeksforGeeks* [online]. 2021 [cit. 2024-03-16]. Dostupné z: https://www.geeksforgeeks.org/android-architecture-patterns/

[21] LUTKEVICH, Ben. Kotlin. In: *TechTarget* [online]. 2022 [cit. 2024-03-16]. Dostupné z: https://www.techtarget.com/whatis/definition/Kotlin

[22] MVVM Architecture Pattern in Android. In: *GeeksforGeeks* [online]. 2023 [cit. 2024-03-21]. Dostupné z: https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/

[23] AUNG, Phyo Thinzar. How does data binding work in MVVM and what are its advantages? In: *Medium* [online]. 2024 [cit. 2024-03-21]. Dostupné z: https://medium.com/@phyothinzaraung/how-does-data-binding-work-in-mvvm-and-what-are-its-advantages-c74d960429f0

[24] IŞIK, Onur Cem. Introduction to MVVM Architecture. In: *Medium* [online]. 2023 [cit. 2024-03-21]. Dostupné z: https://medium.com/@onurcem.isik/introduction-to-mvvm-architecture-5c5558c3679

[25] MVVM Architecture. In: *GitHub* [online]. 2022 [cit. 2024-03-21]. Dostupné z: https://github.com/amitshekhariitbhu/MVVM-Architecture-Android

[26] Data and file storage overview. In: *Android Developer* [online]. 2024 [cit. 2024-03-21]. Dostupné z: https://developer.android.com/training/data-storage

[27] ZINCIRCIOĞLU, Sena. RoomDB vs SQLite : Exploring Database Options for Android Development. In: *Medium* [online]. 2023 [cit. 2024-03-21]. Dostupné z: https://medium.com/huawei-developers/roomdb-vs-sqlite-exploring-database-options-for-android-development-1120151e6737

[28] Save data in a local database using Room. In: *Android Developers* [online]. 2024 [cit. 2024-03-21]. Dostupné z: https://developer.android.com/training/data-storage/room

[29] RAITHATHA, Kathank. Room Database In Android. In: *Medium* [online]. 2023 [cit. 2024-03-21]. Dostupné z: https://medium.com/@kathankraithatha/room-database-in-android-294442467bc8

[30] Introduction to activities. In: *Android Developers* [online]. 2024 [cit. 2024-03-24]. Dostupné z: https://developer.android.com/guide/components/activities/intro-activities

[31] LOSHIN, Peter, David LINTHICUM a Maxine GIZA. XML (Extensible Markup Language). In: *TechTarget* [online]. 2021 [cit. 2024-03-24]. Dostupné z: https://www.techtarget.com/whatis/definition/XML-Extensible-Markup-Language

[32] Material You design. In: *Android source* [online]. [cit. 2024-03-24]. Dostupné z: https://source.android.com/docs/core/display/material

[33] Accessing data using Room DAOs. In: *Android Developers* [online]. 2024 [cit. 2024-05-02]. Dostupné z: https://developer.android.com/training/data-storage/room/accessing-data

[34] Define data using Room entities. In: *Android Developers* [online]. 2024 [cit. 2024-05-02]. Dostupné z: https://developer.android.com/training/data-storage/room/defining-data

[35] RecyclerView. In: *Android Developers* [online]. 2024 [cit. 2024-05-03]. Dostupné z: https://developer.android.com/develop/ui/views/layout/recyclerview

[36] Color Picker Library for Android. In: *GitHub* [online]. 2021 [cit. 2024-05-03]. Dostupné z: https://github.com/Dhaval2404/ColorPicker

[37] Notifications overview. In: *Android Developers* [online]. 2024 [cit. 2024-05-03]. Dostupné z: https://developer.android.com/develop/ui/views/notifications

[38] JSON. In: *MDN Web Docs* [online]. 2023 [cit. 2024-05-03]. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/JSON

[39] Gson. In: *GitHub* [online]. 2023 [cit. 2024-05-03]. Dostupné z: https://github.com/google/gson