Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Android aplikace nákupního seznamu

Matouš Vanča

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Matouš Vanča

Informatika

Název práce

Android aplikace nákupního seznamu

Název anglicky

Android application shopping list

Cíle práce

Cílem práce je vytvořit aplikaci, která bude fungovat jako nákupní seznam. Aplikace bude umět vytvářet, mazat, duplikovat a přejmenovávat seznamy. Seznamy budou tvořeny položkami. Položky půjde přidávat a označovat za nakoupené. Položky budou dělené do kategorií podle typu zboží a podle kategorií bude možné položky filtrovat. Nenakoupené položky půjde převádět mezi seznamy. Součástí aplikace bude historie předchozích nákupů a možnost vytvořit nový seznam z předchozích nákupů. Položky půjde označovat za oblíbené.

Metodika

Budou dodrženy standardy softwarového inženýrství, především UML a WebML. Při tvorbě aplikace bude využito vývojové prostředí Android studio. Aplikace bude napsána v jazyce Kotlin a uživatelské prostředí bude navrženo v jazyce XML. Data aplikace se budou ukládat do databáze SQLite.

Doporučený rozsah práce

30-40 stran

Klíčová slova

Android aplikace, Android Studio, SDK, Kotlin, XML, MVVM, SQLite

---

Doporučené zdroje informací

ALLEN, Grant. Android 4: průvodce programováním mobilních aplikací. 1. vyd. Brno: Computer Press, 2013, ISBN 8025137821; ISBN 9788025137826

Android Developers. Android Developers [online]. [cit. 2018-11-13]. Dostupné z: https://developer.android.com/

JOHNSON, Paul. Using MVVM Light with Your Xamarin Apps [online]. Berkeley, CA: Apress L. P, 2017;2018; ISBN 9781484224748; ISBN 1484224744

Předběžný termín obhajoby

2018/19 ZS – PEF (únor 2019)

Vedoucí práce

doc. Ing. Vojtěch Merunka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 23. 11. 2018

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 23. 11. 2018

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 08. 03. 2019

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Android aplikace nákupního seznamu" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 13.3.2019 _____

Matouš Vanča

Poděkování

Rád bych poděkoval doc. Ing. Vojtěchu Merunkovi, Ph.D. za vedení bakalářské práce, cenné připomínky a rady.

Android aplikace nákupního seznamu

Abstrakt

Bakalářská práce se věnuje návrhu aplikace Nákupní seznam pro operační systém Android s využitím návrhového vzoru Model-View-ViewModel. Nejprve představuje operační systém Android, jeho architekturu a historii vývoje. Následně se zabývá Android aplikacemi a jejich strukturou. Největší pozornost je věnována architektuře aplikace a návrhovým vzorům. Podrobně je také vysvětlena technika data bindingu. Další část práce se zaměřuje na uživatelská rozhraní a interakci s uživatelem. V závěru teoretické části práce jsou popsány nástroje použité k implementaci. V praktické části jsou vytyčeny požadavky a podle nich vypracovány návrhy podoby aplikace. Podle nejvhodnějšího návrhu vybraného pomocí vícekriteriální analýzy variant je zpracován drátový a datový model aplikace. Tyto modely následně sloužily k implementaci výsledné aplikace. Na závěr jsou popsány knihovny využité při implementaci.

Klíčová slova: Android aplikace, Android Studio, SDK, Kotlin, XML, MVVM, SQLite

Android application shopping list

Abstract

The bachelor thesis suggests an application Shopping List for the Android operation system using the Model-View-ViewModel design pattern. Firstly, the paper introduces the Android operation system, its architecture and history of its development. It then focuses on Android applications and their structures. The greatest attention is paid to the architecture of the application and to design patterns. The technique of data binding is explained in detail as well. Next part of the paper is devoted to user interface and to user interaction. The final section of the theoretical part describes the means used for implementation. In the practical part, the requirements are detailed and designs of the forms of the application are suggested. Based on the most suitable design selected via multi-criteria analysis of variants the wireframe and data models of the application are created. These models further serve for the implementation of the final application. In the end, the libraries used for the implementation are described.

Keywords: Android application, Android Studio, SDK, Kotlin, XML, MVVM, SQLite

Obsah

1 Úvod

Chytré mobilní telefony se stávají čím dál tím víc součástí každodenního života běžného člověka. Praktické aplikace nám mohou usnadnit život, protože nosíme telefon všude s sebou. Může nám tak být užitečná i aplikace, která usnadní naše pravidelné aktivity, jakou je například nakupování. Aby taková aplikace byla opravdu užitečná, musí splňovat určité požadavky. V případě nakupování, kdy nahradíme papírové nákupní seznamy za aplikaci, je důležité, aby byla aplikace přehledná, jednoduchá a dobře ovladatelná. Aplikace by měla uživateli umožnit co nejsnáze a nejrychleji najít v seznamech a položkách, co potřebuje. Ovládání, zejména funkce jako označení položky za nakoupenou, by mělo být přizpůsobené pro ovládání jednou rukou.

Kromě praktické stránky musí být aplikace také stabilní a dobře odladěná. K vytvoření kvalitní aplikace nám pomůže zvolení vhodné architektury. Obecně pro mobilní aplikace se hodí vícevrstvá architektura, konkrétně vzory MVVM, MVC, MVP.

Ve své práci bych rád čtenáři představil některé moderní technologie, které nejsou zatím při tvorbě Android aplikací úplně běžné. Jedná se zejména o technologii data bindingu, programovací jazyk Kotlin a knihovny Android Jetpack.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem této práce je vytvořit aplikaci pro operační systém Android, která bude fungovat jako nákupní seznam. Při její tvorbě využít návrhový vzor Model-View-ViewModel (MVVM) a dodržet principy vícevrstvé architektury. Aplikace bude umět vytvářet, mazat, duplikovat a přejmenovávat seznamy. Seznamy budou tvořeny položkami. Položky bude možné přidávat a označovat za nakoupené. Položky budou dělené do kategorií podle typu zboží a podle kategorií bude možné položky filtrovat. Nenakoupené položky půjde převádět mezi seznamy. Součástí aplikace bude historie předchozích nákupů a možnost vytvořit nový seznam z předchozích nákupů. Položky bude možné označovat za oblíbené.

Dalším cílem je použít technologii data bindingu a sadu moderních knihoven Android Jetpack. Tyto knihovny jsem si zvolil, protože umožňují snadnou implementaci vzoru MVVM a zároveň jeho spojení s data bindingem. Při výběru těchto knihoven jsem také zohlednil, že zjednodušují kód a usnadňují práci s životními cykly komponent Android Frameworku. Cílem je také využít jazyk Kotlin, jelikož se jedná o moderní jazyk, který nahradil Javu a stal od roku 2017 oficiálním jazykem pro vývoj Android aplikací.

2.2 Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýzách odborných informačních zdrojů. Teoretická část se věnuje operačnímu systému Android, dále architektuře aplikace, uživatelským rozhraním a vývojovému prostředí Android Studio, sadě nástrojů Android SDK. Nedílnou součástí teoretické části jsou také programovací jazyky a technologie.

Praktická část práce je zaměřena na tvorbu aplikace. V bakalářské práci jsou dodrženy standardy softwarového inženýrství, především UML a WebML. Při tvorbě aplikace je využito vývojové prostředí Android Studio. Aplikace je napsána v jazyce Kotlin a uživatelské prostředí je navrženo v jazyce XML. Data aplikace se ukládají do databáze SQLite.

Na základě teoretických poznatků a výsledků praktické části práce budou formulovány závěry bakalářské práce.

3 Teoretická východiska

3.1 Operační systém Android

Operační systém Android je otevřený software založený na Linuxu. Byl vytvořen pro velkou škálu zařízení, a to s sebou přináší jak výhody, tak nevýhody (1). Výhoda spočívá ve velkém množství výrobců zařízení, což jednak vede k velké rozmanitosti zařízení, jednak díky konkurenci i k nízké pořizovací ceně. Dále pro operační systém Android je dostupné velkém množství aplikací v obchodě Google Play. Hlavní nevýhodu pak představuje především obtížnost optimalizovat aplikace a samotný systém pro takto velké množství zařízení.

3.1.1 Historie

Společnost Android Inc byla založena v říjnu 2003 Richem Minerem, Nickem Searsem, Chrisem Whitem a Andy Rubinem (55). Jejich původní myšlenkou bylo vytvořit operační systém pro digitální fotoaparáty, brzy na to však svůj záměr změnili a přeorientovali se na mobilní telefony (2). V roce 2005 koupil Google společnost Android Inc (55) a došlo k rozhodnutí založit systém na Linuxovém jádře (2). V roce 2007 vzniklo společenství Open Handset Alliance, které sdružuje společnosti zabývající se vývojem technologií pro mobilní telefony. Patří do něj mnoho technologický lídrů, např. Google, Intel, nVidia, Samsung, Qualcomm, Texas Instruments. Cílem společenství je vytvářet otevřené standardy pro mobilní zařízení. (57). V roce 2008 byl představen první telefon s operačním systémem Android (2).

První verze Android 1.0 podporovala systémové notifikace, WI-FI, Bluetooth, přehrávání multimédií, fotoaparát, synchronizaci se službami Googlu, komunikaci SMS a MMS, widgety na domovské obrazovce a další (55).

Verze 1.1 přišla především s opravami chyb v původní verzi a několika novými funkcemi, jako třeba možnost uložit přílohy ze zpráv nebo možnost skrytí číselníku (55).

Android 1.5 přišel s dalšími novinkami, byla přidána funkce klávesnice na obrazovce, schopnost nahrávat video a přibyly animované přechody obrazovek (55).

S Android verze 1.6 přichází podpora různých rozlišení obrazovek a podpora CDMA mobilních sítí. Novinkou je panel pro rychlé vyhledávání a plná integrace fotoaparátu, kamery a galerie (55).

Android 2.0 přináší podporu více účtů, režim pro ovládání v autě, integraci s Facebookem a novou odemykací obrazovku (55).

Ve verzi 2.2 došlo k osvěžení designu a ke zrychlení systému. Verze byla rozšířena o podporu sdílení internetového připojení pomocí USB a WIFI. Přibyla také podpora push notifikací, bluetooth handsfree a adobe flash. S touto verzí také začíná řada telefonů Nexus, což jsou telefony s čistým operačním systémem a jsou podporovány přímo Googlem (55).

Mezi nejdůležitější změny, které přišly s verzí 2.3, patří podpora NFC, různých senzorů, nových video a audio formátů a čelní kamery (55).

Android 3.0 vznikl především pro tablety a soustředil se na změnu designu. Mezi nové funkce lze zařadit podporu USB OTG, FLAC audio formátu a šifrování uživatelských dat (55).

Brzy po Androidu 3.0 vydal Google verzi 4.0, která už však cílila na všechna zařízení a stala se poměrně úspěšnou. Kromě dalších změn v designu
byla doplněna o nové funkce, mezi jež lze zařadit možnost pořídit screenshot obrazovky, rozpoznávání tváře, podpora VPN a WIFI Direct a podpora formátu obrázků WebP (55).

V následujících verzích 4.1, 4.2 a 4.3 se Google zaměřil na zrychlení systému. Novými funkcemi byly chytré karty Google Now, rozšířené notifikace, podporu USB audio a vícekanálového zvuku (55).

S verzí 4.4 přichází změny v designu ale i další významné změny. Novinkou je Android Runtime ART, podpora nositelné elektroniky, bezdrátového tisku, senzoru pro měření kroků a možnost změnit výchozí aplikaci pro SMS a domovskou obrazovku (55).

S verzí 5.0 se systém proměnil do Material Designu, mezi důležité změny patří podpora 64bitových procesorů, funkce chytrého zámku, integrovaná funkce ovládání LED fotoaparátu a chytřejší notifikace (55).

S Androidem 6.0 se mění systém oprávnění. Aplikace nyní vyžadují jednotlivá povolení přístupu k určitým částem systému nebo hardwaru až při jejich běhu. Uživatel se může rozhodnout, zda je povolí, nebo ne, a svoji volbu pak může v budoucnu změnit v nastaveních (12).

Mezi nejdůležitější změny ve verzi 7.0 patří podpora rozdělení obrazovky na dvě okna, ve kterých můžou odděleně běžet dvě různé aplikace. Změnou procházejí také notifikace, které nyní umí víc funkcí jako například rychlé odpovědi nebo vícero akčních tlačítek (60).

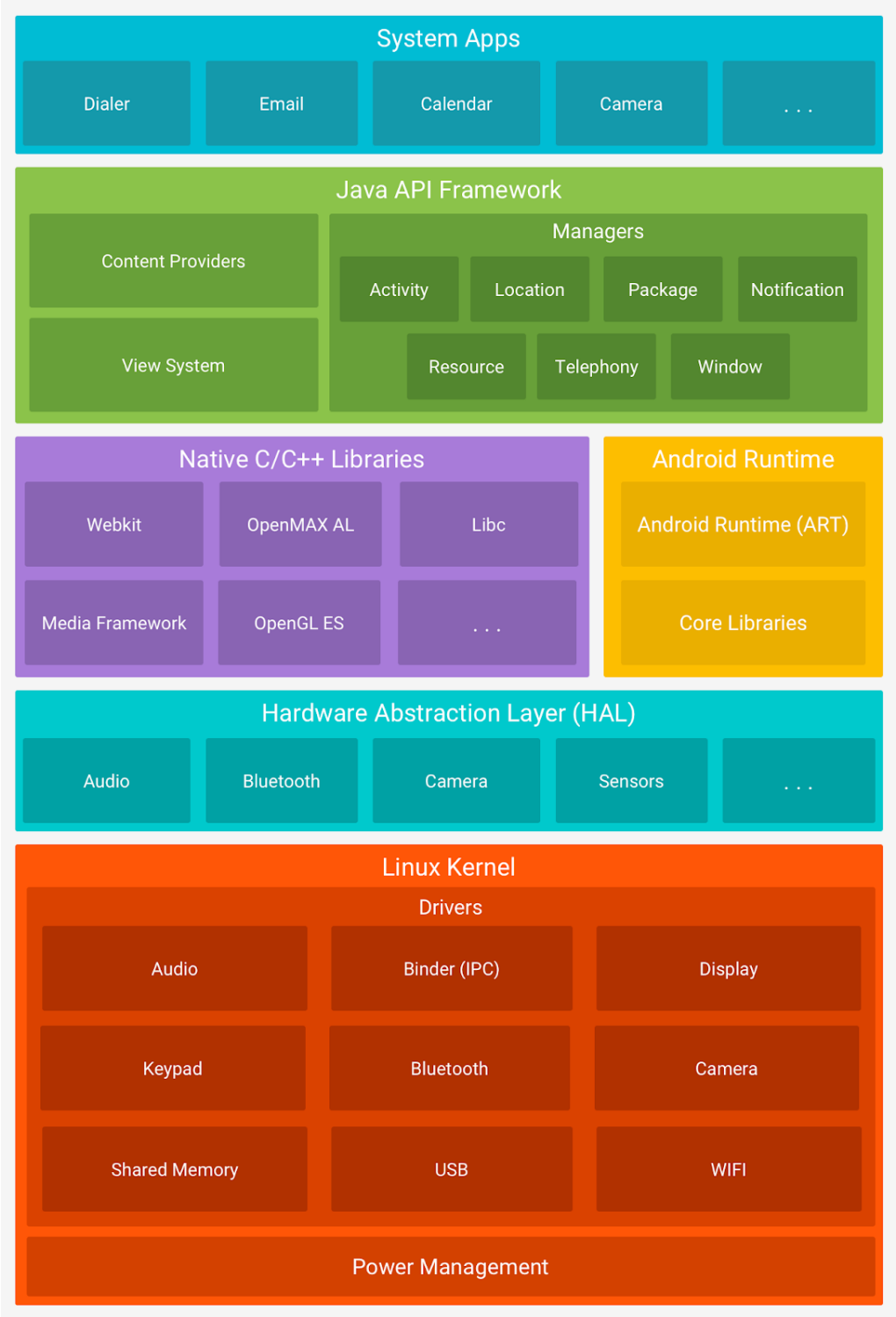
Android 8.0 přináší funkci automatického vyplňování přihlašovacích údajů, chytrý výběr textu, režim obrazu v obraze umožňující překrýt část obrazovky jinou aplikací.

Novinkou jsou také instantní aplikace, které umožňují spuštění z internetového prohlížeče bez nutnosti instalovat je (59).

Nejnovější verzí je Android 9.0 a Google si na této verzi dal opravdu záležet. Změn a nových funkcí je zde opravdu mnoho. Mezi nejpodstatnější patří podpora Wi-Fi RTT, výřezů v displeji, více fotoaparátů a fotoaparátů připojených přes USB, podpora HDR videí a HEIF formátu obrázků. Další novinkou jsou funkce přizpůsobující se uživateli např. adaptivní jas a adaptivní baterie nebo funkce App Actions, která podle aktuálních dat na displeji umí vyvolat akci z jiných aplikací, nebo funkce Slices, která zobrazuje akce z aplikací ve vyhledávání Google a na dalších místech. Změn se dočkalo i ovládání systému, místo klasických softwarových tlačítek lze telefon ovládat jedním tlačítkem a gesty (58).

3.1.2 Architektura systému

Operační systém Android je založen na Linuxovém jádře. Systém spoléhá na základní funkce Linuxového jádra, jako je například správa vláken a paměti. Výhodou použití Linuxového jádra je jeho kvalitní zabezpečen a možnost využít již vyvinutých ovladačů. Další výhodou je, že systém lze snadno spustit na různých zařízeních, a tím je usnadněná jeho přenositelnost a rozšíření. Následující obrázek znázorňuje architekturu systému a ukazuje její součásti (1).



Obrázek č. 1. Architektura systému Android (1)

3.1.3 Zastoupení na trhu

Operační systém Android vládne trhu s mobilními telefony. Využívá jej 71 % všech mobilních telefonů, druhý je operační systém iOS s 28 %. Výjimkou je severní Amerika a

Oceánie, kde iOS lehce vyhrává. Poměr Androidu vůči systému iOS odpovídá zhruba celosvětovému poměru, k lednu 2019 činí přesně 76,2 % ku 21,9 %. Operační systém iOS zaznamenal výrazný nárůst v září 2018, kdy byla vydána trojice iPhonů Xs, Xs Max a Xr. Tento nárůst výrazně ovlivnil poměr rozšířenosti systémů. V České republice podíl systému iOS od září do listopadu vzrostl ze 17 % na necelých 25 %, v severní Americe dokonce předčil dosud vítězící Android (3).

V rozšířenosti jednotlivých verzí je první Android 6.0 s kódovým názvem Marshmallow, který běží na 21,3 % zařízení s Androidem. Druhý je Android Nougat ve verzi 7.0, který najdeme v 18,1 % zařízení, a třetí Android Lollipop verze 5.1 s 14.4 % (4).

Version	**Codename**	**API**	**Distribution**
2.3.3 - 2.3.7	Gingerbread	10	0.2 %
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3 %
4.1.x	Jelly Bean	16	1.1 %
4.2.x		17	1.5 %
4.3		18	0.4 %
4.4	KitKat	19	7.6 %
5.0	Lollipop	21	3.5 %
5.1		22	14.4 %
6.0	Marshmallow	23	21.3 %
7.0	Nougat	24	18.1 %
7.1		25	10.1 %
8.0	Oreo	26	14.0 %
8.1		27	7.5 %

Tabulka č. 1. Rozdělení verzí Androidu (4)

3.2 Android aplikace

Aplikace pro operační systém Android mohou být napsány v programovacích jazycích Kotlin, Java, Dart, C++, nebo C#. Kód je kompilován sadou nástrojů Android SDK.

Výsledkem kompilace je archiv s příponou apk, který obsahuje zkompilovaný kód, data a zdrojové soubory. Jediný apk soubor pak obsahuje vše potřebné k instalaci aplikace (5).

Každá Android aplikace běží ve svém vlastním prostoru zabezpečeném vícero bezpečnostními prvky. První z bezpečnostních prvků spočívá v tom, že systém Android je ve své podstatě multiuživatelský systém Linux, díky čemuž je každá aplikace jiným uživatelem. Jako další zabezpečení systém přiřadí každé aplikaci unikátní uživatelskou ID, která je používána systémem, ale aplikace ji nezná. Následně systém nastaví oprávnění pro všechny soubory v aplikaci, takže pouze uživatelská ID přiřazená aplikaci k nim může mít přístup. Také kód aplikace je spuštěn odděleně od ostatních aplikací, protože každý proces běží na samostatném virtuálním stroji. Dále je aplikace zabezpečena tím, že každá aplikace běží na samostatném Linuxovém procesu. Pokud je potřeba, aby byla spuštěna některá ze součástí aplikace, systém proces spustí. Když není proces potřeba, nebo nemá systém dostatek paměti, je proces ukončen. Systém Android umožňuje aplikacím přístup pouze k těm komponentám, které jsou nezbytné pro běh aplikace, proto si aplikace musí o přístup k datům ze zařízení, jako jsou například kontakty, SMS, hovory, úložiště, Bluetooth nebo fotoaparát, požádat (5), (56).

3.2.1 Součásti Android aplikace

Základem aplikace je jedna nebo více komponent, z nichž každá je vstupním bodem pro systém nebo uživatele. Existují různé komponenty: Activities, Services, Broadcast receivers a Content providers, které se liší svým účelem a životním cyklem.

Activities neboli aktivity, mají uživatelské rozhraní a slouží k interakci s uživatelem. Aktivita obvykle reprezentuje jednu obrazovku. Aplikace může obsahovat jednu nebo více aktivit (5).

Services neboli služby, jsou komponenty, které se starají o běh operací na pozadí, nemají uživatelské rozhraní. Existují tři typy služeb. První typ provádí operace viditelné pro uživatele, protože musí zobrazit viditelnou notifikaci. Tento typ také zůstává běžet na pozadí i ve chvíli, kdy uživatel skryje aplikaci. Využívají např. aplikace přehrávající hudbu. Další typ slouží k déle trvajícím operacím a není pro uživatele přímo viditelný. Příkladem může být stahování nebo ukládání dat. Poslední typ nabízí interakci se službou pomocí principu klient-server. Komponenty mohou službě posílat požadavky a přijímat výsledky. Tento typ služby běží, pouze pokud je svázán s jednou nebo více komponentami (5), (6).

Broadcast receiver je komponenta, která naslouchá systémovým nebo uživatelským událostem a umí na ně reagovat. Např. změna připojení internetu, slabá baterie nebo zachycení snímku fotoaparátem.

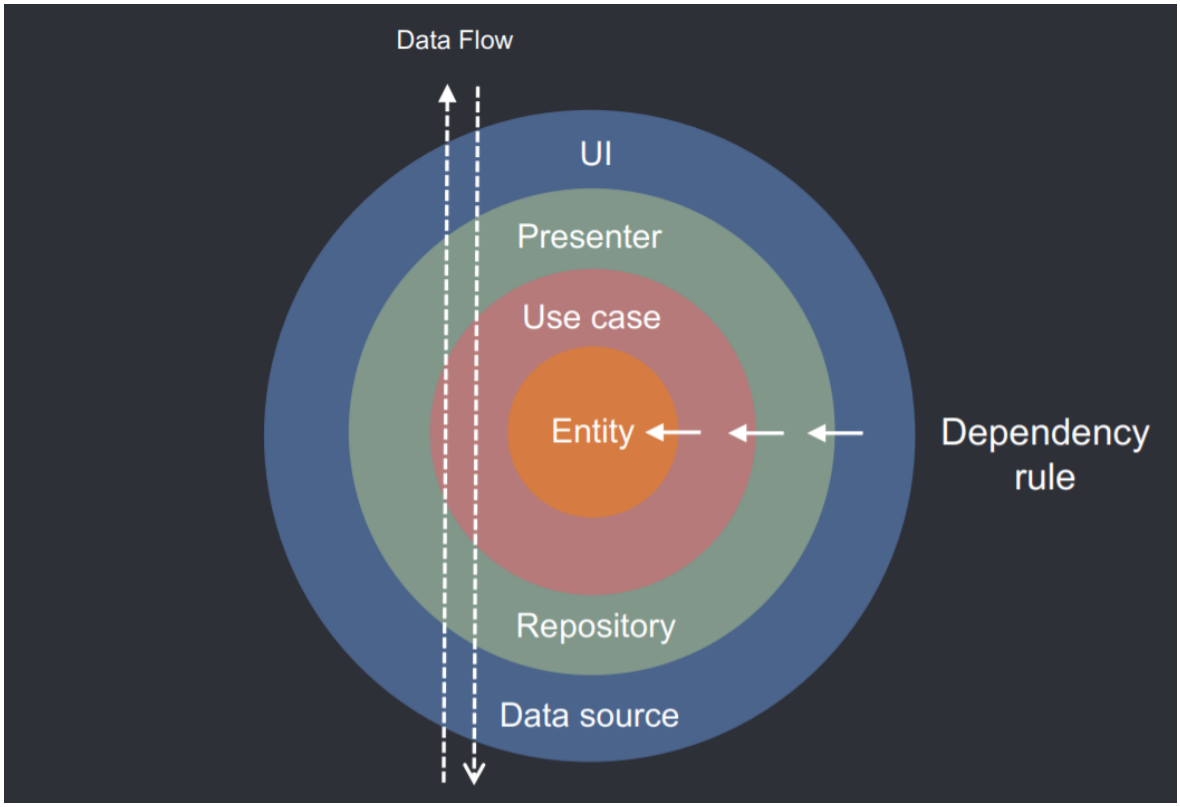
Další komponentou je Content provider. Ten se stará o správu dat aplikace a poskytuje uživatelské rozhraní pro práci s nimi. Primárně je určen pro bezpečný přístup k datům mezi aplikacemi (7).

3.3 Architektura aplikace

Při navrhování aplikace musíme navrhnout strukturu, která bude brát v potaz všechny funkční i technické požadavky, případy použití, design, zabezpečení, udržitelnost i rozšiřitelnost aplikace. Zvolení správné struktury má velký vliv na kvalitu a výkon aplikace. V opačném případě může být aplikace nestabilní a málo flexibilní pro budoucí změny, lze v ní těžko odstraňovat chyby, nelze je efektivně testovat a může obsahovat mnoho zbytečného nebo duplicitního kódu. Pro návrh architektury lze využít některý z osvědčených architektonických vzorů, které představím v následujících kapitolách (54).

3.3.1 Vícevrstvá architektura

Protože můžeme Android aplikace libovolně organizovat a strukturovat, bývají výsledkem velké aktivity a fragmenty, které mají příliš mnoho zodpovědnosti a chyb. A proto je důležité zvolit vhodnou architekturu, která oddělí aplikační logiku od dat a uživatelského rozhraní. Architekturu, nezávisle na typu aplikace, můžeme rozdělit do několika skupin softwarových komponent. Tyto logické skupiny tvoří vrstvy, které se odlišují funkcionalitou komponent a umožňují znuvupoužitelnost. Nadřazené vrstvy představují klienta nižším vrstvám, a naopak nižší vrstvy poskytují služby vrstvám nadřazeným. Závislost ve vícevrstvé architektuře je viditelná z následujícího obrázku (54).



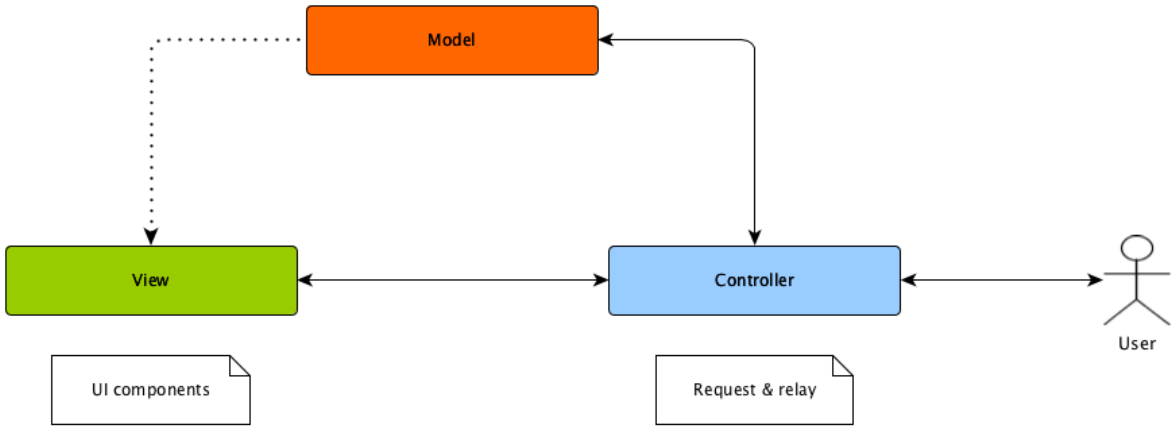
Obrázek č. 2. Závislost vrstev a tok dat ve vícevrstvé architektuře (11)

Strukturu lze obecně rozdělit na vrstvu s uživatelským rozhraním, doménovou vrstvu, prezentační vrstvu, datovou vrstvu, síťovou vrstvu a vrstvu s databází. Pro konkrétní případy není nutné použít všechny vrstvy, ve své aplikaci například nevyužiji síťovou a doménovou vrstvu, protože není struktura mé aplikace natolik složitá a nevyžaduje připojení k vzdáleným zdrojů například k serverům. Vrstva s uživatelským rozhraním využívá Android Framework, který obsahuje třídy sloužící jako základní prvky pro tvorbu rozhraní. Tato vrstva získává data z vrstvy prezentační a předává jí události z uživatelského rozhraní. Pro propojení těchto dvou vrstev jsem využil metody data bindingu, které se podrobněji věnuji v samostatné kapitole. Prezentační vrstva je zodpovědná za předávání dat vrstvě s uživatelským rozhraním ale zároveň neví nic o rozhraním samotném, není proto závislá na Android frameworku. Doménová vrstva, někdy také nazývaná business vrstva, obsahuje základní funkce systému. Tuto vrstvu jsem ve své aplikaci vynechal, neboť v ní neprobíhají žádné složité operace. Datová vrstva poskytuje data vyšším vrstvám a stará se o komunikaci s vrstvami se zdroji dat. Zdrojových vrstev může být více například databáze nebo server. V aplikaci využívám pouze vrstvu pro lokální databázi, tato vrstva se stará o komunikaci s databází (54).

K implementaci vícevrstvé architektury je vhodné použít některý z návrhových vzorů. Mezi nejpoužívanější návrhy patří Model-View-Controller (MVC) a z něj vycházející Model-View-Presenter (MVP) a Model-View-ViewModel (MVVM). V následujících kapitolách se věnuji jejich vlastnostem a rozdílům (8), (9).

3.3.2 MVC

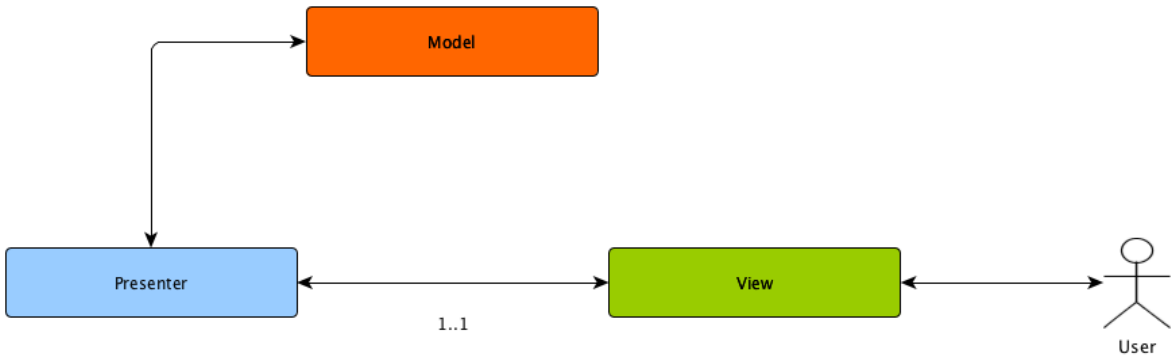
MVC je architektonickým vzor, který rozděluje datový model, uživatelské rozhraní a řídící logiku. Vrstvy jsou oddělené a jejich úpravy mají minimální vliv na ostatní vrstvy. Model reprezentuje třídy, které popisují data a operace s nimi. Vrstva View reprezentuje uživatelské rozhraní a je odpovědná pouze za zobrazování dat, jež dostane v podobě Modelů od Controller vrstvy. Controller zpracovává požadavky od uživatele, které dostane skrze vrstvu View. Zpracovaná data předává zpátky vrstvě View. Controller v tomto vzoru funguje jako koordinátor mezi daty a uživatelským rozhraním (10).



Obrázek č. 3. MVC (10)

3.3.3 MVP

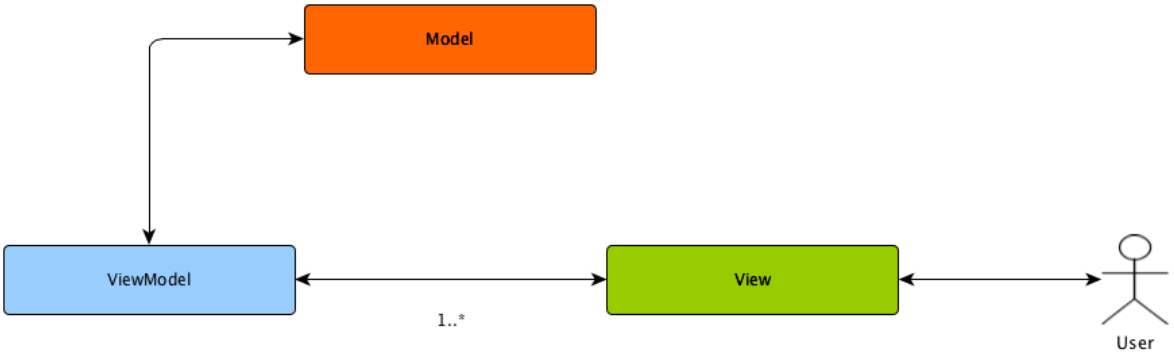
Vzor MVP podobně jako MVC dělí architekturu na tři části. Model stejně jako u MVC je sadou tříd, které popisují obchodní logiku neboli data a možnosti manipulace s nimi. Vrstva View má na starosti zobrazování dat od Presenteru. Presenter narozdíl od Controlleru je oddělen od vrstvy View a komunikuje s ní pomocí rozhraní. O vstupy od uživatele se stará vrstva View. Mezi View a Presenterem je vztah jeden k jednomu, to znamená, že jeden View je spojen jen s jedním Presenterem. Vrstva View nemá žádnou vazbu na Model (10).



Obrázek č. 3. MVP (10)

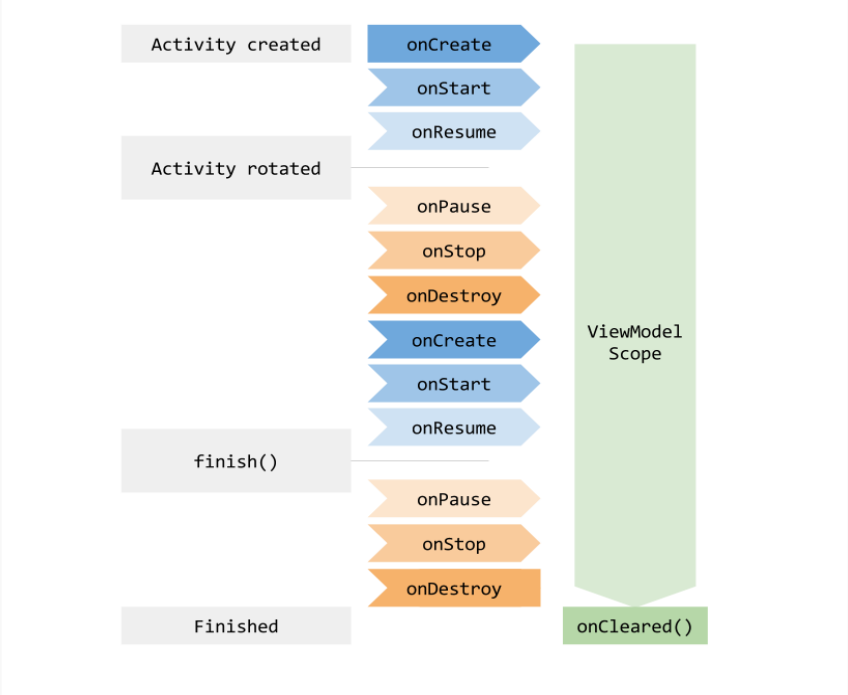
3.3.4 MVVM

Stejně jako předchozí vzory, vzor MVVM neboli Model–View–ViewModel dělí architekturu na tři vrstvy. Na rozdíl od předchozích dvou vzorů, vzor MVVM podporuje dvoucestný data binding mezi vrstvami View a ViewModel. To umožňuje automatické zobrazení změn ve ViewModelu do View. Obvykle se pro toto chování využívá princip pozorování (Observable pattern). Model stejně jako u obou předchozích vzorů je sadou tříd, které popisují obchodní logiku čili data a možnosti manipulace s nimi. Vrstva View představuje uživatelské rozhraní. View pozoruje změnu dat ve vrstvě ViewModel, tyto změny následně aplikuje do uživatelského rozhraní. Opačným směrem View předává vrstvě ViewModel data o událostech z uživatelského rozhraní. Vrstva View by měla obsahovat minimum z aplikační logiky a měla by se starat pouze o zobrazování dat a odesílání informací o událostech vrstvě ViewModel. ViewModel je vrstva spojující uživatelské rozhraní s daty, obsahuje metody a vlastnosti, které udržují stav vrstvy View, a na základě událostí ve View manipuluje v modelem. Více View může být spojeno s jednou vrstvou ViewModel. View má referenci na ViewModel, ale naopak ViewModel nemá žádnou referenci na View nebo jinou část Android Frameworku (10), (13), (14), (15), (16).



Obrázek č. 4. MVVM (10)

V projektu jsem se rozhodl využít vzor MVVM především pro možnost použití data bindingu a knihoven z kolekce Android Jetpack. Pro komunikaci mezi vrstvami využívám k předávání dat třídu LiveData, která umožňuje použití principu pozorování. Na rozdíl od běžných tříd využívajících principů pozorování umí třída LiveData pracovat s životním cyklem komponent aplikace, jako jsou aktivity a fragmenty. Pro tvorbu ViewModel využívám stejnojmennou třídu, která je také z kolekce Jetpack (17).



Obrázek č. 5. Životní cyklus třídy ViewModel (16)

3.3.5 Data binding

Data binding je technika, která propojuje data s uživatelským rozhraním. Data se automaticky přenáší do uživatelského rozhraní. Pro data binding lze využít i princip pozorování a tehdy se přenáší i změny dat. Data binding může fungovat i naopak, kdy se změna v rozhraní automaticky ukládá. Výhodou data bindingu je značné snížení množství potřebného kódu. Pro tuto techniku využívám knihovny z kolekce Android Jetpack, což zajišťuje výbornou kompatibilita s architekturou MVVM (18).

Pro použití data bindingu v uživatelském rozhraní je nutné jasně definovat strukturu. To se provádí speciálními xml tagy na začátku zdrojového xml souboru. Kořenovým elementem souboru musí být element layout, v něm vnořen element data a kořenový element uživatelského rozhraní, např. LinearLayout. V elementu data se definují proměnné pomocí elementů variable. Každý element variable musí mít definován atribut type a name. Atribut name slouží k použití dané proměnné a atribut type definuje typ proměnné, kterým může být základní datový typ, objekt, pole, kolekce nebo třeba interface (18).

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto">
    <data>
        <variable
            name="viewmodel"
            type="com.myapp.data.ViewModel" />
    </data>
    <ConstraintLayout... /> <!-- UI layout's root element -->
</layout>
```

Obrázek č. 6. Definování dat v data bindingu (18)

Pokud je daný typ pozorovatelný, jsou data zobrazená v uživatelském rozhraní aktualizována se změnou dat. K tomu lze využít knihovny LiveData a ViewModel, které jsou také z kolekce Android Jetpack. Použití definované proměnné se provádí pomocí výrazu @{}, příklad použití je viditelný na následujícím obrázku (18).

```
<TextView
    android:text="@{viewmodel.userName}" />
```

Obrázek č. 7. Použití data bindingu (18)

Knihovna pro data binding vygeneruje podle zdrojového xml souboru třídu umožňující přístup k definovaným proměnným a prvkům uživatelského rozhraní.

Pro použití data bindingu, kdy se data projevují v rozhraní a zároveň se změna v rozhraní projevuje v datech, lze využít dvoucestného data bindingu. Ten se použije pomocí zápisu @={} (19).

```
<CheckBox
    android:id="@+id/rememberMeCheckBox"
    android:checked="@={viewmodel.rememberMe}"
/>
```

Obrázek č. 8. Použití dvoucestného data bindingu (19)

Dvoucestný data binding vyžaduje složitější implementaci než jednoduchý data binding. Pro naslouchání datům lze využít třídu BaseObservable a anotaci @Bindable (19).

```
class LoginViewModel : BaseObservable {
    // val data = ...

    @Bindable
    fun getRememberMe(): Boolean {
        return data.rememberMe
    }

    fun setRememberMe(value: Boolean) {
        // Avoids infinite loops.
        if (data.rememberMe != value) {
            data.rememberMe = value

            // React to the change.
            saveData()

            // Notify observers of a new value.
            notifyPropertyChanged(BR.remember_me)
        }
    }
}
```

Obrázek č. 9. Použití třídy BaseObservable a anotace Bindable (19)

Pro každý výraz použitý ve xml návrhu rozhraní existuje adaptér, který provádí volání z frameworku. Například pro použití výrazu *android:text=”ukázkový text”* existuje

funkce *setText*, která hodnotu *text* danému prvku nastaví. Pro vlastní potřeby, například pro využití při data bindingu, je možné vytvořit si adaptér vlastní (18).

```
@BindingAdapter("app:goneUnless")
fun goneUnless(view: View, visible: Boolean) {
    view.visibility = if (visible) View.VISIBLE else View.GONE
}
```

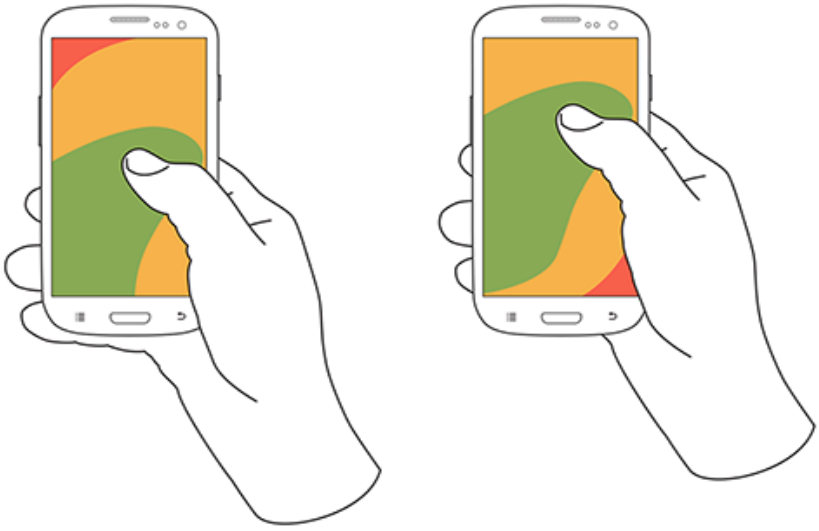
Obrázek č. 10. Ukázka vlastního adaptéru (18)

3.4 Uživatelské rozhraní

Uživatelské rozhraní neboli UI (z anglického User Interface) slouží k interakci člověka s počítačem. Uživatelským rozhraním může být jakákoliv možnost uživatele ovlivňovat běh stroje, ať už se jedná o tlačítko, myš nebo třeba klávesnici. Cílem této interakce je efektivně ovládat stroj, zatímco je uživatel informován o jeho stavu (20).

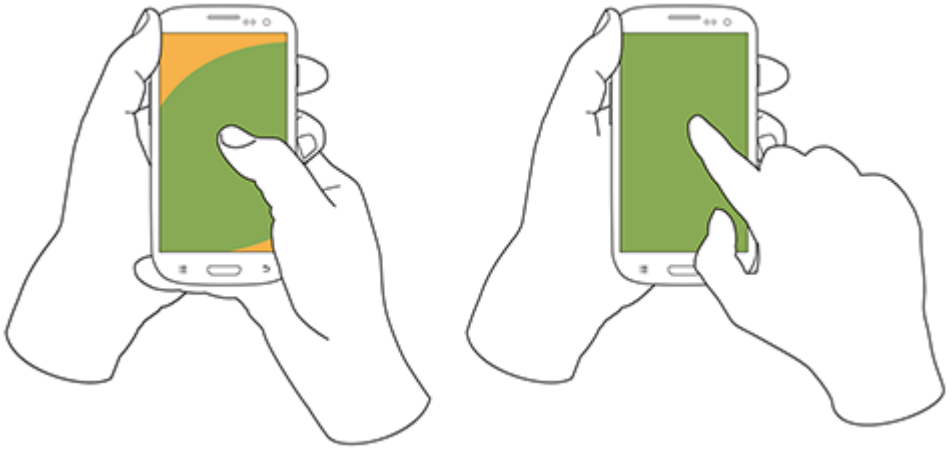
Speciálním případem uživatelského rozhraní je grafické uživatelské rozhraní, které umožňuje ovládání pomocí interaktivních grafických prvků. Mobilní telefony jsou založeny na ovládání přes grafické rozhraní pomocí dotykového panelu zabudovaného přímo v displeji zařízení. Protože je tento typ ovládání až na výjimky stejný pro všechna zařízení s operačním systémem Android, budu se věnovat pouze jemu (20).

Velkou roli v ovládání mobilních telefonů hraje způsob, jakým je telefon držen a ovládán. Na základě výzkumu Stevena Hoobera drží a ovládá telefon jednou rukou 49 % uživatelů, 36 % drží telefon v jedné ruce a druhou jej ovládá a 15 % používá k ovládání obě ruce. Je proto velmi důležité, aby rozhraní bylo dobře ovladatelné palcem a aby všechny důležité prvky rozhraní byly dobře dostupné (21).



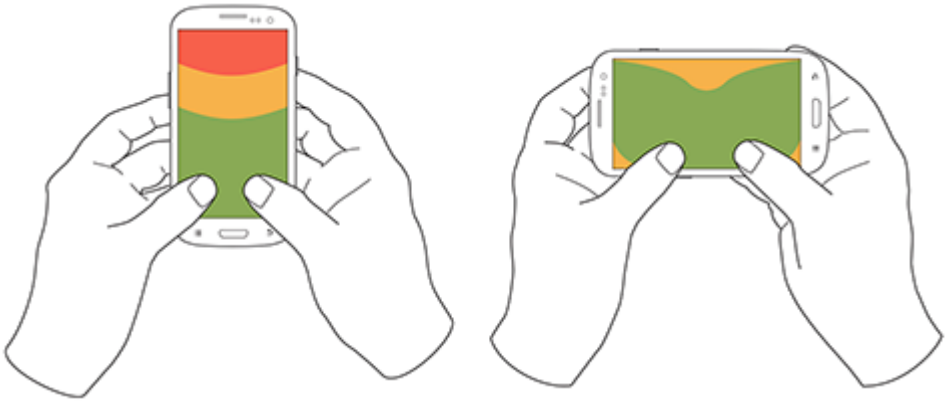
Obrázek č. 11. Držení telefonu jednou rukou a ovládání palcem (21)

Při držení telefonu jednou rukou a využívá k ovládání telefonu 72 % uživatelů palec druhé ruky a 28 % ukazováček druhé ruky (21).



Obrázek č. 12. Držení telefonu jednou rukou a ovládání prsty druhé ruky (21)

Další možností je držet telefon oběma rukama a zároveň použít oba palce. Tento způsob uživatel nejčastěji využívá při psaní na virtuální klávesnici (21).



Obrázek č. 13. Držení telefonu oběma rukama a ovládání dvěma palci (21)

Hoober také uvádí, že nejpřesnější doteky má uživatel uprostřed obrazovky. U krajů obrazovky jsou doteky až dvakrát méně přesné (21).

3.4.1 Hodnocení uživatelského rozhraní

Zda je uživatelské rozhraní intuitivní nebo uživatelsky přívětivé, můžeme měřit pomocí znaků použitelnosti. Jedná se o měřitelné veličiny, pro které si musíme stanovit kritéria a porovnat uživatelská rozhraní mezi sebou. Jakob Nielsen uvádí pět znaků použitelnosti:

1. **naučitelnost** – jak jednoduché je pro uživatele provádět základní úkoly při prvním setkání s uživatelským rozhraním?
2. **efektivnost** – jak rychle zvládnou uživatelé provádět úlohy poté, co si osvojí uživatelské rozhraní?
3. **zapamatovatelnost** – když se uživatelé vrátí do rozhraní po delší době, jak rychle se jim obnoví znalost uživatelského rozhraní?
4. **chybovost** – jak často uživatelé chybují, jak závažných chyb se dopouštějí a jak snadno se mohou z chybového stavu vrátit?
5. **spokojenost** – jak je pro uživatele uspokojující používání uživatelského rozhraní?

Kromě použitelnosti je důležitým znakem také **přínosnost**, která ukazuje, jak moc uživatelské rozhraní slouží k tomu, k čemu bylo vytvořeno. **Přínosnost** společně s **použitelností** ukazují, jak je uživatelské rozhraní ve výsledku **užitečné** (22).

3.4.2 Material Design

Material Design je jednotný grafický styl pro návrh aplikací vytvořený Googlem a představený v roce 2014. Jedná se o jednoduchý a minimalistický styl, který vychází z flat

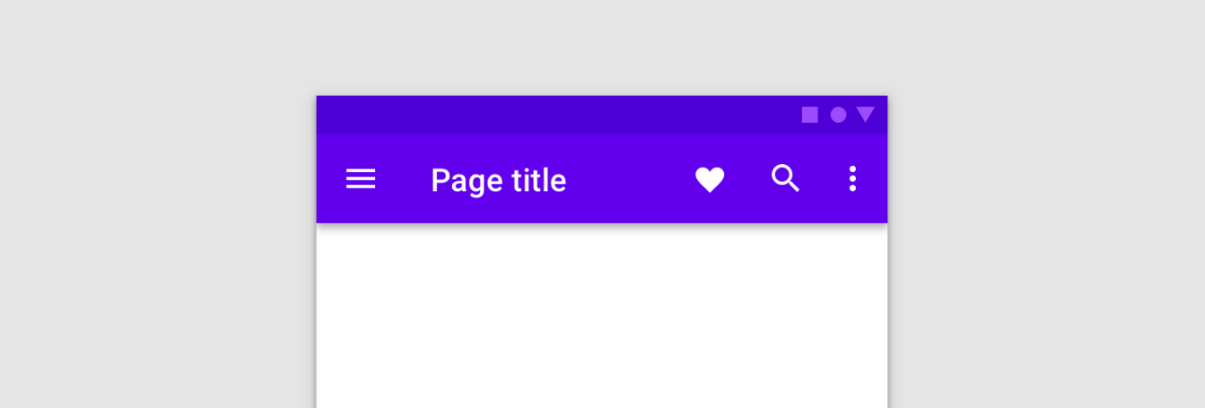
designu. Cílem Material Designu je sjednotit styl počítačových, mobilních a webových aplikací (25).

Material Design využívá principy tištěného designu. Prvky designu reagují na uživatele a využívají energii interakce s uživatelem k tomu, aby se transformovaly nebo animovali. Díky tomu se zlepšuje přehlednost a orientace v rozhraní. Každá vykreslená část aplikace je jakoby nakreslená inkoustem na papír. Papír má tvar a pozici a je vybarven inkoustem, protože sám o sobě barvu nemá. Papír může mít různou šířku nebo výšku, může dokonce pokrýt celou obrazovku, nebo mít naopak velikost jediného tlačítka, se i přesouvat a měnit tvar. Tloušťka papíru je 1dp. Papír může mít tvar obdélníku nebo kruhu, nebo může mít papír zaoblené či zkosené rohy. Prostor v Materiál Designu má hloubku a jednotlivé papíry v něm mohou být různě vysoko, a podle toho vrhají stíny. Tato výška se také uvádí jako vyzdvižení (elevation) (23). Google vydal k Material Designu soubor pravidel, jak jej použít, aby nedošlo k porušení principů (24).

3.4.3 Komponenty Android Frameworku

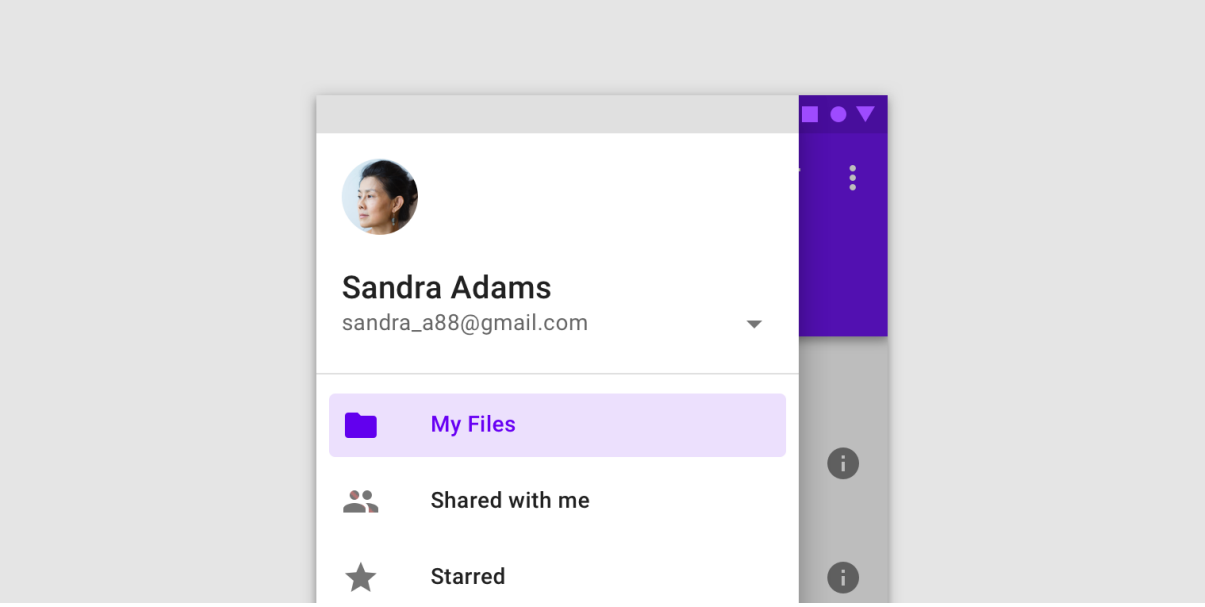
Pro návrh aplikace podle Material Designu lze použít komponenty z Android Frameworku. Mezi nejpoužívanější komponenty patří AppBar, Tabs, Navigation Drawer, Bottom Navigation, Floating Action Button a další.

AppBar zaujímá místo ve vrchní části obrazovky a jeho obsah se vždy vztahuje k aktuální obrazovce. AppBar může zobrazovat text, kterým obvykle bývá název obrazovky. V levé části se nachází navigační tlačítko s ikonou sloužící pro primární akci navigace, může zde být také tlačítko zpět nebo třeba takzvané “hamburger” tlačítko, které otevírá Navigation Drawer či jiné menu. V pravé části AppBaru se nachází menu s dalšími akcemi, jednotlivá tlačítka tohoto menu jsou buď zobrazena formou ikony přímo na AppBaru nebo jsou schovaná za ikonou tří teček, jež rozbalí další možnosti menu. AppBar má výchozí úroveň vyzdvižení (elevation) 4dp (26).



Obrázek č. 14. AppBar (26)

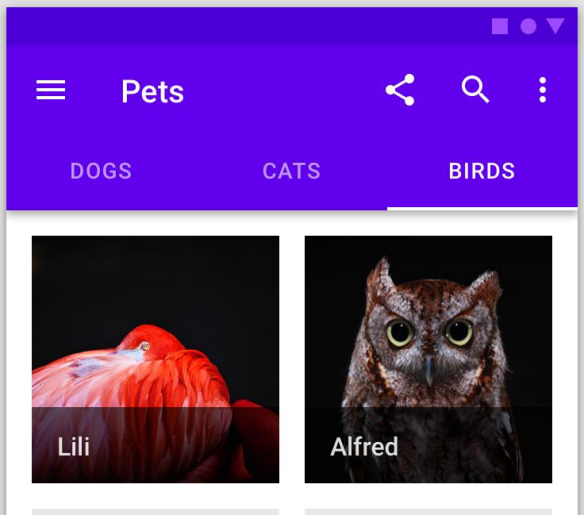
Navigation Drawer umožňuje přímý přístup k cílovým obrazovkám v aplikaci a hodí se pro větší počet cílových obrazovek. Navigation Drawer se otevírá tahem z kraje obrazovky nebo "hamburger" tlačítkem na Appbaru. Prvky menu jsou organizovány podle důležitosti a podle své funkce do skupin. Navigation Drawer může mít hlavičku, kde se často používá obrázek, text nebo další tlačítka s akcemi. Pokud Drawer obsahuje více prvků, než se vejde na display, lze jeho obsah scrollovat. Prvky v menu mohou mít ikonu a jejich označení by mělo být co nejkratší a pouze na jeden řádek. Navigation Drawer má výchozí úroveň vyzdvižení 16dp (27).



Obrázek č. 15. Navigation Drawer (27)

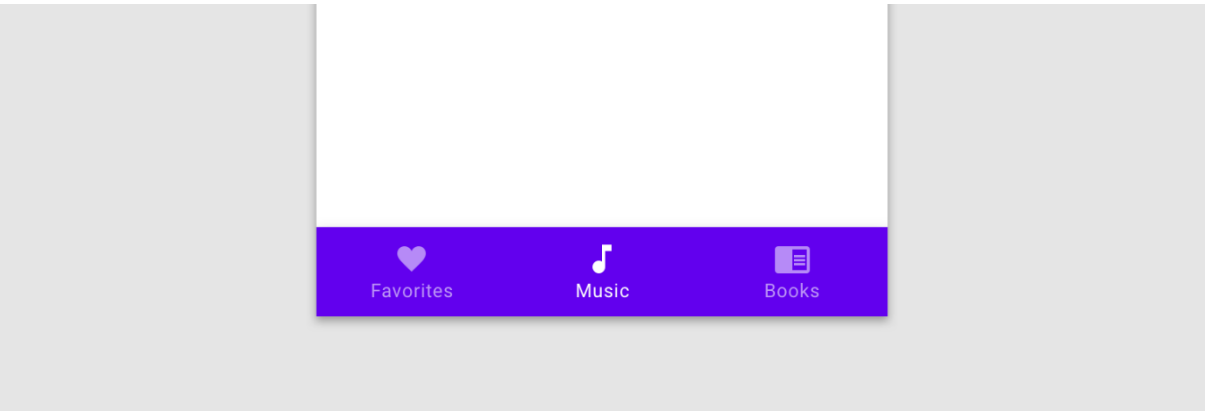
Tabs je komponenta, která slouží k organizaci obrazovek stejné úrovně v hierarchii. Jedná se o záložky aktuálně svázané se zobrazenou obrazovkou. Záložek může být libovolné

množství. Mezi záložkami lze libovolně scrollovat nebo mohou být zafixovány, pokud jich je menší množství. Záložky lze také přepínat gestem swipe, toto gesto lze provést tažením prstu po obsahu zobrazeném na obrazovce doprava nebo doleva. Každá záložka má název, nebo ikonu, případně obojí, styl všech záložek by však měl být jednotný. Aktivní záložka je indikována barevným podtržením a barevně zvýrazněným textem či ikonou. Velikost textu by měla být na všech záložkách stejná a text by neměl být na víc než jeden řádek (28).



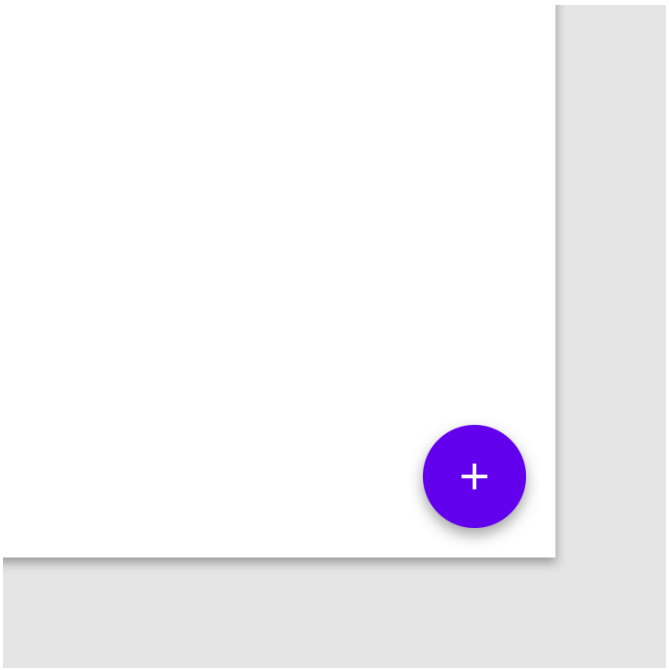
Obrázek č. 16. Tabs (28)

Bottom Navigation je navigační komponenta umístěná ve spodní části obrazovky. Její menu slouží k navigaci mezi obrazovkami nejvyššího stupně v navigační hierarchii. Použití Bottom Navigation se doporučuje pouze pro tři až pět položek v menu. Jednotlivé položky mají textový popisek, který může být u neaktivních položek skryt. Text položek by neměl přesahovat jeden řádek a neměl by být příliš dlouhý. Položky mají také ikonu, která je vždy zobrazena. Aktivní položky se mohou lišit od neaktivních podbarvením nebo sytostí barvy (27).



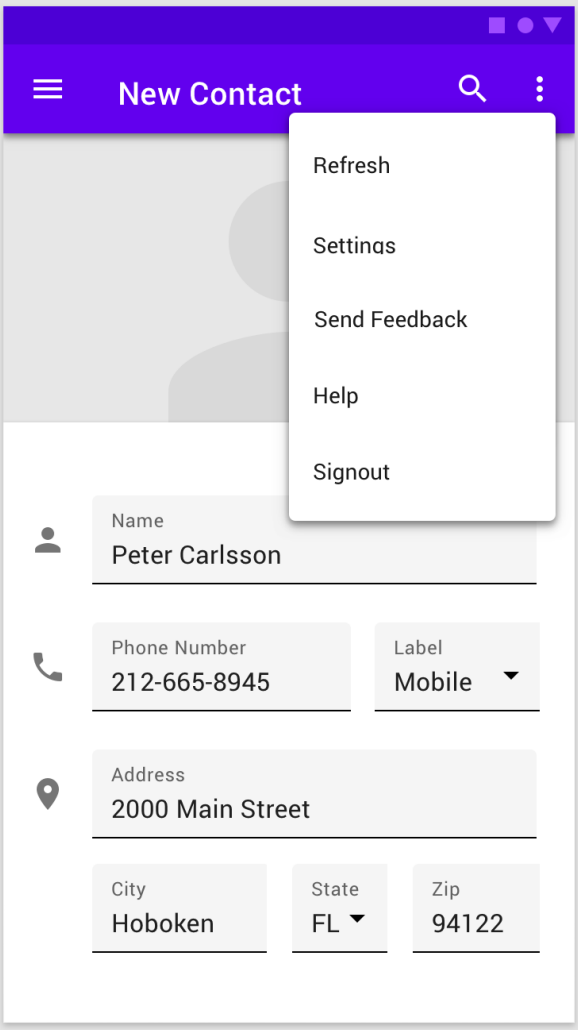
Obrázek č. 17. Bottom Navigation (27)

Floating Action Button (FAB) je plovoucí tlačítko, které se používá pro nejběžnější primární akci vztahující se k obsahu obrazovky. Zpravidla je umístěno v pravém dolním rohu obrazovky a vyskytuje se na obrazovce pouze jednou. Nejčastěji má kulatou podobu a ikonou uprostřed. Akce vyvolaná po jeho stisknutí by měla být konstruktivní a měla by patřit mezi důležité akce. Tlačítko by nemělo být použito pro mazání nebo třeba archivování obsahu. Tlačítko je možné schovávat, přesouvat, nebo transformovat do jiných podob. Alternativní podobou tlačítka je ovál s textem nebo textem a ikonou. Pokud je FAB zobrazen s textem, neměl by jeho text přesahovat jeden řádek ani být zkrácen kvůli velikostí tlačítka. Výchozí vyzdvižení této komponenty má hodnotu 6dp (29).



Obrázek č. 18. Floating Action Button (29)

Popup menu zobrazuje seznam akcí menu na dočasné kartě, která se po vyvolání objeví nad místem, odkud bylo menu vyvoláno. Položky menu mohou obsahovat další podmenu. Položky menu mohou být reprezentovány textem nebo textem s ikonou. Akce menu vždy souvisí s obsahem, ze kterého bylo menu vyvoláno. Styl karty nebo položek může být upraven tak, aby seděl se stylem celé aplikace. Výchozí vyzdvižení Popup menu má hodnotu 8dp (30).



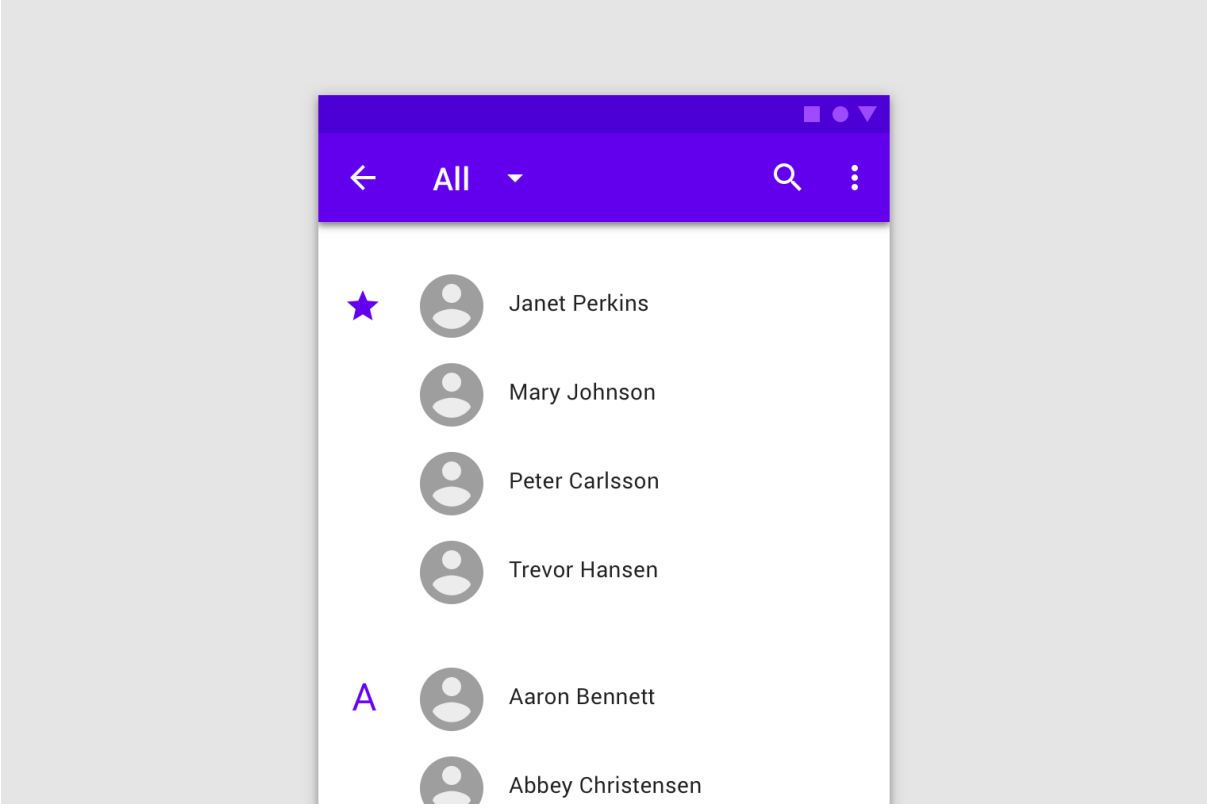
Obrázek č. 19. Popup menu (31)

Card je komponenta, která slouží jako kontejner pro další komponenty. Obvykle mívá zaoblené rohy a vyzdvižení 1 až 8dp. Používá se na oddělení obsahu v seznamech nebo

tabulkách. Při dotyku prstem by měla karta zvýšit vyzdvižení a indikovat tak akci s uživatelem. Obsah karet je možné dynamicky zvětšovat, ale je zakázáno karty dělit nebo slučovat. Obsah karty by neměl být scrollovatelný, ale místo toho by se měla karta zvětšit tak, aby byl celý její obsah zobrazen (32).

Obrázek č. 20. Cards (32)

Další komponenta List slouží jako kontejner pro větší počet položek, které jsou si vizuálně podobné. Položka listu se může skládat z dalších komponent, například z tlačítek, obrázků nebo textů (33).



Obrázek č. 21. List (33)

3.5 Vývojové prostředí Android Studio

Android Studio je oficiální prostředí pro tvorbu aplikací pro Android. Je vyvíjeno jako otevřený software společnostmi Google a JetBrains. První předběžná verze vyšla 5. května v roce 2013 a první stabilní verze v prosinci 2014. Poslední aktuální verze 3.3 vyšla v lednu 2019. Android Studio je postavené na prostředí IntelliJ IDEA, od společnosti JetBrains, a díky tomu využívá všechny jeho přednosti, mezi které patří chytré doplňování kódu, podpora různých programovacích jazyků, praktické uživatelské prostředí, funkce pro refactoring a inspekci kódu atd. Android Studio disponuje spoustou nástrojů a funkcí pro vývoj aplikací. Google je neustále vylepšuje a přidává další, díky čemuž se Android Studio stává komplexním nástrojem. Mezi nejdůležitější patří následující nástroje (34).

Navigační editor slouží k vizualizaci a návrhu navigace v aplikaci. Pro použití editoru je nutno vložit knihovnu Navigation z kolekce Android Jetpack. Vizualizace zobrazuje aktivity a fragmenty, které pomocí spojení tvoří diagram. V editoru lze vkládat do spojení parametry a definovat jejich název a typ. Aktivity, fragmenty, přechody a parametry lze pak dohledat a použít podle názvů v automaticky vygenerovaném kódu, který vygeneruje knihovna (34).

Editor uživatelského rozhraní umožňuje díky grafickému náhledu praktické a snadné navrhování uživatelských rozhraní. Prvky lze do rozhraní vkládat přetažením z nabídky místo psaní XML kódu. Editor umožňuje dynamicky měnit náhled pro různé velikosti obrazovek a rozlišení, čímž lze dobře vyzkoušet, jak bude rozhraní vypadat na různých telefonech. Součástí editoru je paleta dostupných prvků, náhled na strukturu rozhraní a panel s atributy aktuálně vybraného prvku rozhraní (35).

APK analyzátor slouží k analýze výsledných APK souborů, které vznikají kompilací projektu. Analyzátor rozloží soubor na jeho součásti, jež lze procházet a prohlížet, a také zobrazí jejich velikost i jejich velikost po ZIP kompresi. Analyzátor rovněž umí porovnávat dva APK soubory mezi sebou (36).

Profiler je nástroj pro optimalizaci výkonu aplikace, umožňuje sledovat využití procesoru, paměti, sítě a baterie (37).

Gradle je nástroj se zaměřením na automatizaci kompilace projektu, správu zdrojových souborů a knihoven. Gradle využívá pluginy k rozšířen svých funkcí (38).

Funkce Android Studia lze rozšířit velkým množstvím dostupných pluginů, které dělají Android Studio ještě praktičtějším nástrojem.

3.6 Android SDK

Android SDK je sada nástrojů, nezbytná pro vývoj aplikací pro Android. Mezi nejdůležitější nástroje Android SDK patří Build-Tools, Android Debug Bridge (ADB) a Android Emulátor. Build-Tools obsahují nástroje pro kompilaci zdrojových souborů a nástroje pro práci s výsledným apk souborem, například apksigner pro podepisování balíčků nebo zipalign pro redukování výsledné velikosti souboru apk. Android Debug Bridge je program, který slouží pro komunikaci se zařízením. Android Emulátor umožňuje spustit virtuální zařízení na počítači a používat jej k vývoji aplikací (39).

3.7 Programovací jazyky a technologie

3.7.1 Kotlin

Programovací jazyk Kotlin byl představen společností JetBrains v roce 2011, od roku 2017 se stává oficiálním jazyk pro vývoj Android aplikací. Kotlin je vyvíjen jako otevřený software. Jedná se o moderní jazyk, který stejně jako Java využívá JVM (virtuální stroj pro jazyk Java), proto je plně kompatibilní s Javou, díky čemuž může využívat Java knihovny. Jazyk Kotlin je null-safety, což znamená, že umí pracovat s prázdnou hodnotou *null*. Mezi

další výhody jazyka patří možnost rozšířit třídu o další funkce bez nutnosti znát kód této třídy, automatické odvozování datových typů, jednodušší lambda výrazy, singletony, a další (40), (41).

3.7.2 XML

Jazyk XML je značkovací jazyk, byl navržen a vyvinut konsorciem W3C. Při vývoji Android aplikací se používá k navrhování rozvržení obrazovek nebo jednotlivých prvků, k definování zdrojových textů, stylů, barev, manifestu a dalších zdrojových souborů. Ve vývoji pro Android je využita rozšiřitelnost jazyka. Jsou zde definovány použitelné tagy a lze vytvářet další (42).

3.7.3 SQL

SQL je dotazovací jazyk používaný pro komunikaci s relační databází. Existují různé dialekty jazyka SQL, většina však umí základní příkazy, jako jsou Select, Insert, Update, Delete, Create a Drop. Operační systém Android umožňuje aplikacím ukládat data do databáze SQLite (43).

3.7.4 JSON

JSON je speciální formát pro výměnu dat. Je navržen tak, aby byl snadno čitelný člověkem i strojem. Jeho použití při programování je nezávislé na použitém programovacím jazyce. Pro ukládání dat využívá univerzální datové struktury, které podporují téměř všechny moderní programovací jazyky. Ve své aplikaci využívám JSON k uchování výchozích dat v aplikaci (44).

3.7.5 GIT

Git je verzovací systém použitelný při vývoji software. Pomocí Gitu lze evidovat změny v kódu a slučovat změny při práci více lidí na jednom projektu. Git dává přístup k celé historii projektu a změny lze proto prohlížet a případně vracet. Git pracuje lokálně a stahování změn a jejich ukládání na server probíhá po akci uživatele (45).

4 Aplikace Nákupní seznam

Aplikace Nákupní seznam je praktickou aplikací pro zjednodušení nakupování. Aplikace umí spravovat nákupní seznamy a položky v nich. Umožňuje položky zařadit do kategorií a podle nich je v seznamech filtrovat, což můžeme využít při nákupu určitého sortimentu zboží, například při návštěvě drogerie si můžeme zobrazit pouze položky z tohoto obchodu. Položky lze označovat jako oblíbené a v budoucnu znovu vložit do seznamu. Tato funkce se nejlépe hodí k označování pravidelně nakupovaných produktů, jakými může být mléko, máslo nebo vajíčka. Seznamy lze archivovat a posléze použít znovu, čehož využijeme, pokud pravidelně nakupujeme určitou skupinu produktů, například seznam surovin z receptu. Nejdůležitější funkcí, kterou použijeme při samotném nakupování, je funkce označení položky jako nakoupené. Díky tomu, že se položky označené jako nakoupené řadí na konec seznamu, si snadno udržíme přehled o nákupním košíku a zároveň nezapomeneme nic koupit.

4.1 Požadavky

Funkční požadavky

- Přidání seznamu

Aplikace bude umožňovat vytvořit nový seznam.

- Zobrazení seznamů

Aplikaci bude umět zobrazit seznamy a stručné informace o nich.

- Akce nad seznamem

Nad seznamem bude možné vyvolat seznam akcí, mezi které bude patřit: smazání, přejmenování, duplikování a archivování. Každá akce bude vyžadovat potvrzení.

- Zobrazení detailu seznamu

Aplikace bude umět zobrazit detail vybraného seznamu. V detailu bude zobrazený název vybraného seznamu a seznam položek, které do seznamu patří. Položky budou rozděleny na označené a neoznačené a seřazeny podle data přidání.

- Filtrace položek

Zobrazené položky v seznamu bude možné filtrovat podle jejich kategorie,

- Přidání položky

Do seznamu bude možné přidat položky. Při přidávání položky bude nutné zvolit název položky a kategorii, do které patří.

- Označování položek seznamu

Položku bude možné snadno označit za koupenou.

- Akce nad položkou seznamu.

Položku bude možné smazat, přejmenovat a přidat do oblíbených, nebo vyřadit z oblíbených.

- Oblíbené položky

Aplikace bude umožňovat zobrazit seznam oblíbených položek.

- Přidání oblíbené položky

Ze seznamu oblíbených položek bude možné přidat jednu nebo více položek do již existujícího seznamu.

- Zobrazení archivovaných seznamů

Aplikace bude umět zobrazit archivované seznamy.

- Akce nad archivovaným seznamem

Archivovaný seznam bude možné smazat nebo podle něj vytvořit seznam nový.

- Zobrazení detailu archivovaného seznamu

Aplikace bude umět zobrazit detail vybraného archivovaného seznamu. V detailu bude zobrazený název vybraného seznamu a seznam položek, které do seznamu patří. Položky budou rozděleny na označené a neoznačené a seřazeny podle data přidání.

- Zobrazení kategorií

Aplikace bude umět zobrazit seznam dostupných kategorií.

- Přidání kategorie

Aplikace umožní přidat další kategorie.

- Akce nad kategorií

Kategorie bude možné mazat a přejmenovávat.

Nefunkční požadavky

- Operační systém

Aplikace bude vytvořena pro operační systém Android a bude jej podporovat od verze 4.03 Ice Cream Sandwich.

- Přehledné uživatelské rozhraní

Uživatelské rozhraní aplikace bude jednoduché, přehledné a podle pravidel Material Designu. Aplikace bude snadno ovladatelná palcem jedné ruky.

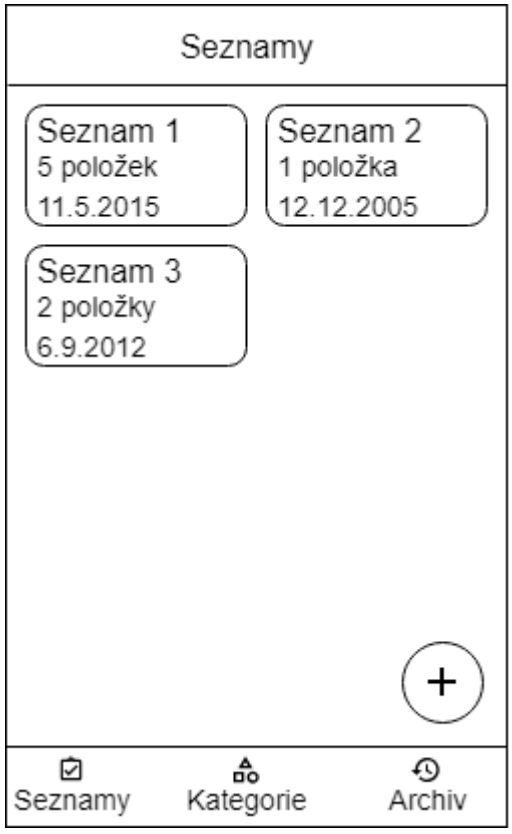
- Stabilita aplikace

Aplikace bude stabilní, nebude obsahovat chyby, které by vedly k pádu aplikace. Aplikace nebude zamrzávat a bude po celou dobu svého běhu rychle reagovat na uživatelské podněty.

4.2 Návrhy uživatelského rozhraní

4.2.1 Návrh 1

První návrh využívá pro navigaci v aplikaci komponentu z Android Frameworku spodní navigační lištu (Bottom Navigation). Aplikace rozdělena na tři hlavní obrazovky Seznamy, Kategorie a Archiv a jednu vedlejší – Detail seznamu. Každá z hlavních obrazovek má svoji záložku v navigační liště. Na každé obrazovce je ve vrchní části AppBar, který zobrazuje název aktuální obrazovky. Pro přidávání obsahu je použita komponenta Floating Action Bar, která se po kliknutí rozbalí na kartu s textovým polem a případně dalšími prvky.

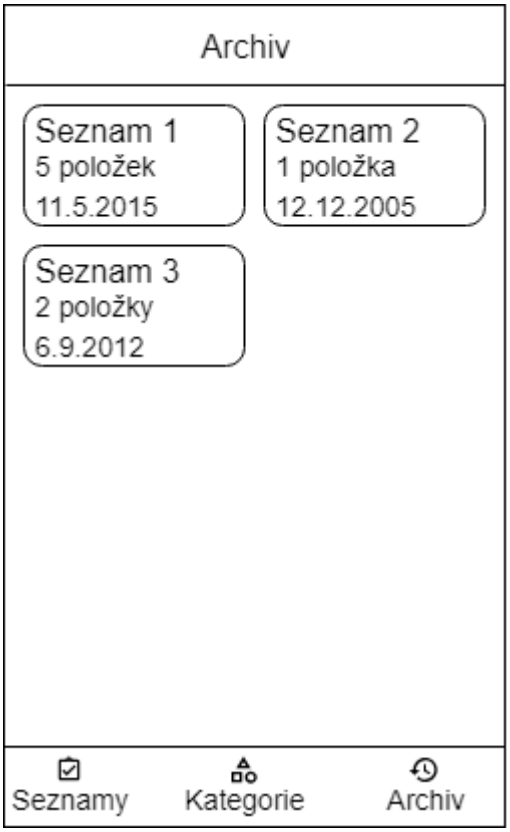


Obrázek č. 22. Návrh 1 - obrazovka seznamy (autor Matouš Vanča)

Obrazovka Seznamy zobrazuje seznamy v tabulce. Každý ze seznamů je zobrazen pomocí komponenty CardView jako karta, která zobrazuje název, informace o počtu položek a datum vytvoření.

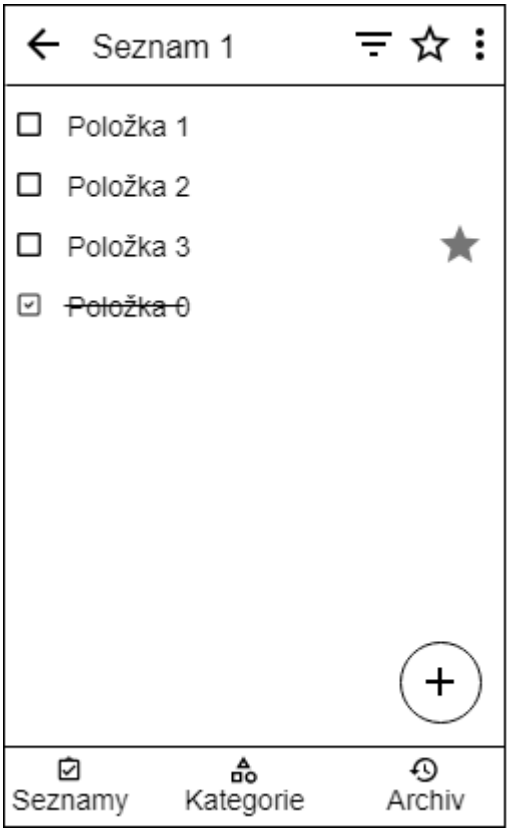
Obrázek č. 23. Návrh 1 - obrazovka kategorie (autor Matouš Vanča)

Obrazovka Kategorie zobrazuje list kategorií, každá kategorie má název a ikonu pro vyvolání menu. Třetí obrazovkou je Archiv, zde jsou zobrazeny seznamy, které byly archivovány.



Obrázek č. 24. Návrh 1 - obrazovka archiv (autor Matouš Vanča)

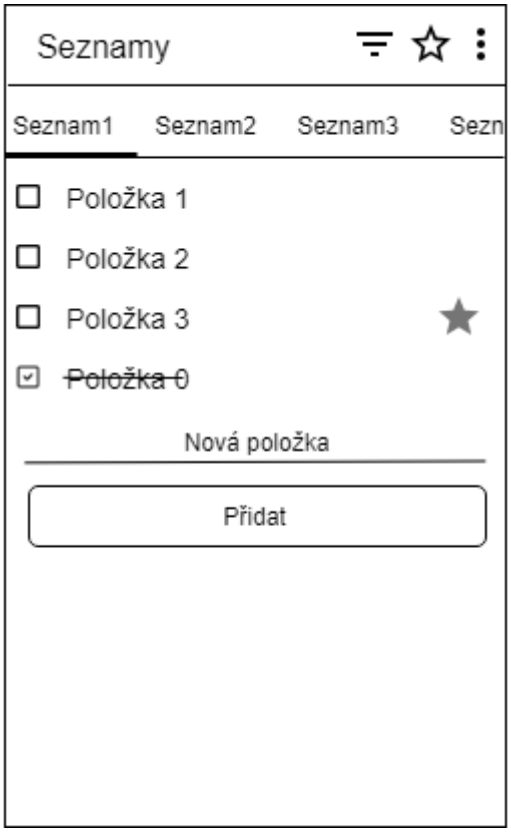
Obrazovka Archiv je vizuálně stejná jako obrazovka Seznamy, liší se jen dostupnými akcemi, které lze provést nad seznamem.



Obrázek č. 25. Návrh 1 - obrazovka Detail seznamu (autor Matouš Vanča)

Poslední obrazovkou je Detail seznamu. Tato obrazovka se vyvolá otevřením seznamu z obrazovky Seznamy nebo z obrazovky Archiv a zobrazuje list položek. Obrazovka Detail seznamu má v AppBaru menu s akcemi dostupnými pro aktuální zobrazený seznam. Akce vyvolané z menu se dokončují a potvrzují dialogem.

4.2.2 Návrh 2

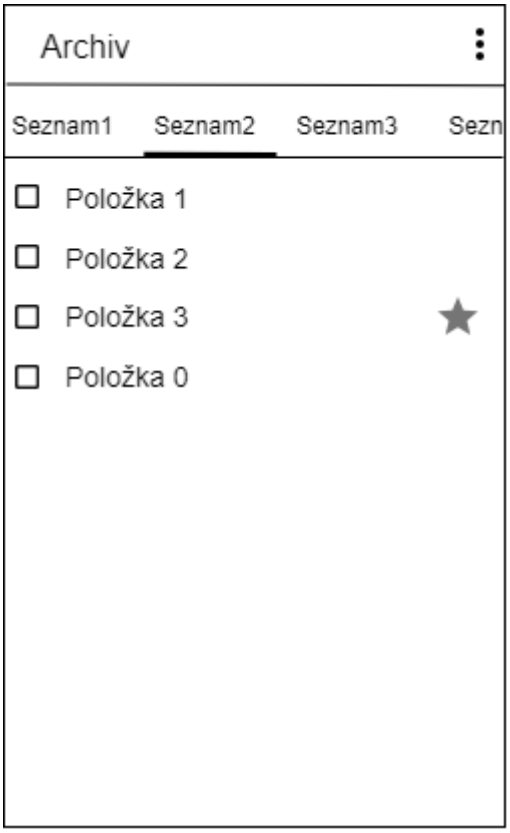


Obrázek č. 26. Návrh 2 - hlavní obrazovka (autor Matouš Vanča)

Tento návrh má jednu hlavní obrazovku a dvě vedlejší. Hlavní obrazovka využívá AppBar a Tabs. AppBar zobrazuje název obrazovky a akce menu. V menu jsou dostupné akce pro aktuální seznam a pro přechod do archivu a kategorií. Záložky komponenty Tabs slouží k zobrazení seznamů jako záložek a umožňuje vybrat seznam. Zbytek obrazovky zobrazuje aktuální vybraný seznam. Mezi seznamy lze také přecházet gestem swipe doprava nebo doleva. Přidání seznamu je možné pomocí akce v menu. Položky do aktuálního seznamu se přidávají pomocí tlačítka a textového pole. Akce vyvolané z menu se dokončují a potvrzují dialogem. Vedlejší obrazovky jsou obrazovka kategorií a archiv.

Obrázek č. 27. Návrh 2 - obrazovka Kategorie (autor Matouš Vanča)

Obrazovka kategorií zobrazuje list kategorií a textové tlačítko s polem pro přidání kategorie nové.

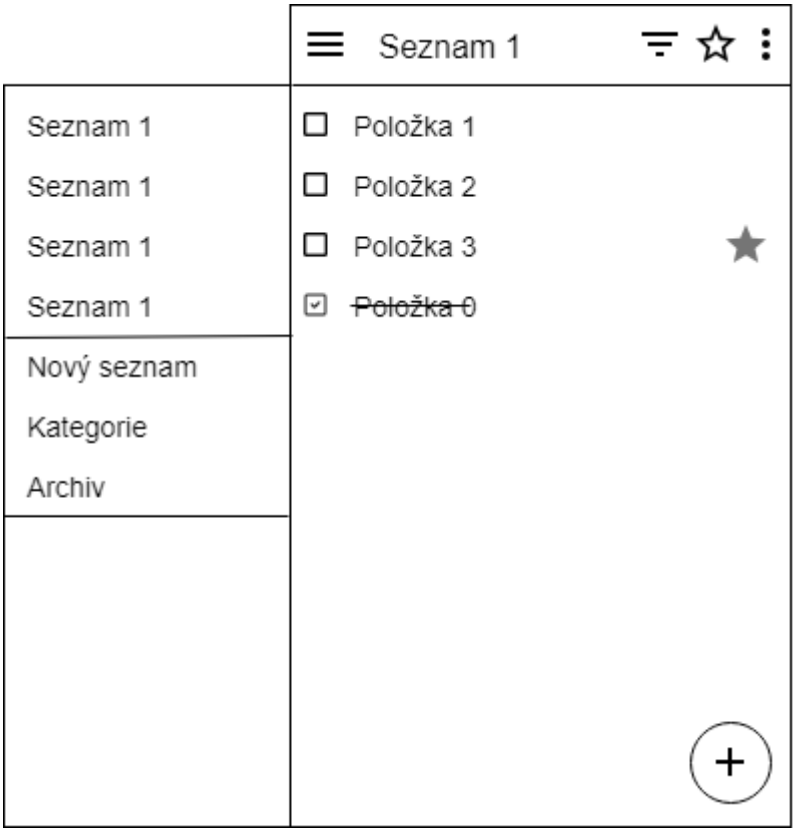


Obrázek č. 28. Návrh 2 - obrazovka Archiv (autor Matouš Vanča)

Obrazovka Archiv zobrazuje archivované seznamy, stejně jako hlavní obrazovka, v záložkách.

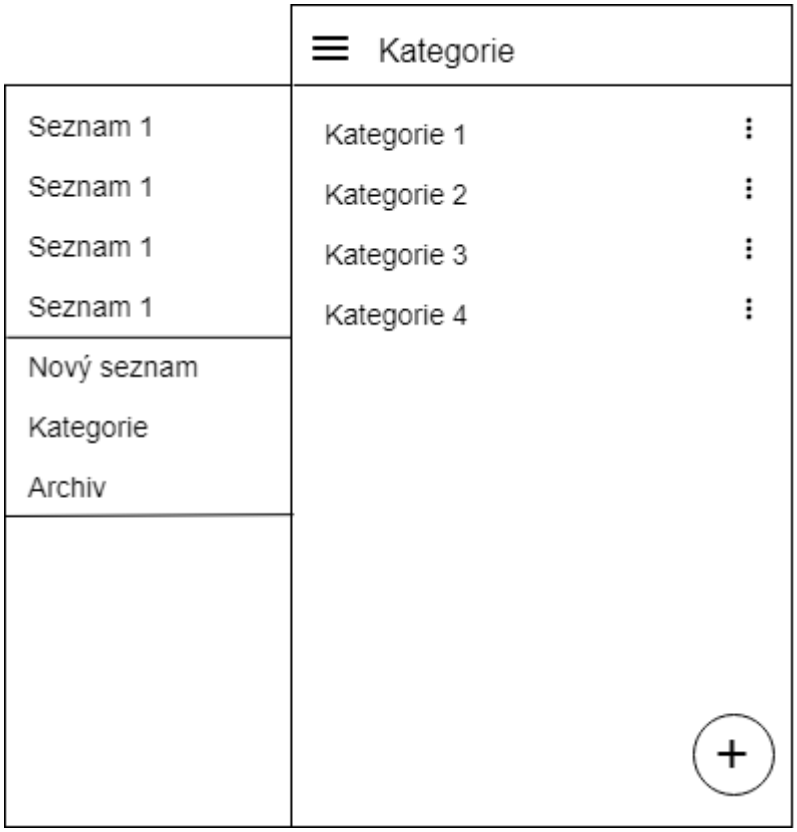
4.2.3 Návrh 3

Poslední návrh využívá komponentu Navigation Drawer, ta obsahuje hlavní menu, ze kterého lze přecházet mezi obrazovkami. Navigation Drawer je možné otevřít pomocí "hamburger" tlačítka v levém horním rohu nebo tažením prstu od levého kraje obrazovky.



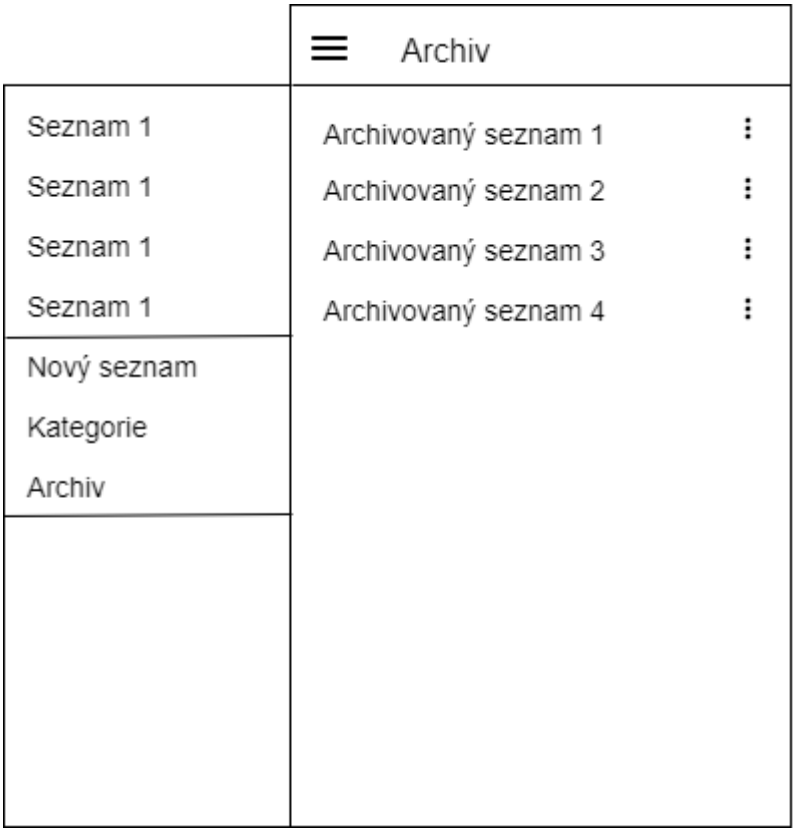
Obrázek č. 29. Návrh 3 - hlavní obrazovka (autor Matouš Vanča)

Hlavní obrazovka zobrazuje detail seznamu, ten je tvořen položkami. Obrazovka detail seznamu má v horní části AppBar, který obsahuje název seznamu a vedlejší menu s akcemi filtrace, oblíbených položek a atd. V pravém dolním rohu se nachází Floating Action Button, který otevírá dialog pro přidávání položek.



Obrázek č. 30. Návrh 3 - obrazovka Kategorie (autor Matouš Vanča)

Další obrazovkou je obrazovka Kategorie, ta zobrazuje v listu jednotlivé kategorie a tlačítka pro jejich editaci. V pravém dolním rohu je stejně jako na první obrazovce Floating Action Button, který slouží pro přidávání kategorií.



Obrázek č. 31. Návrh 3 - obrazovka Archiv (autor Matouš Vanča)

Poslední obrazovkou je Archiv, který zobrazuje v listu archivované seznamy.

4.2.4 Výběr UI

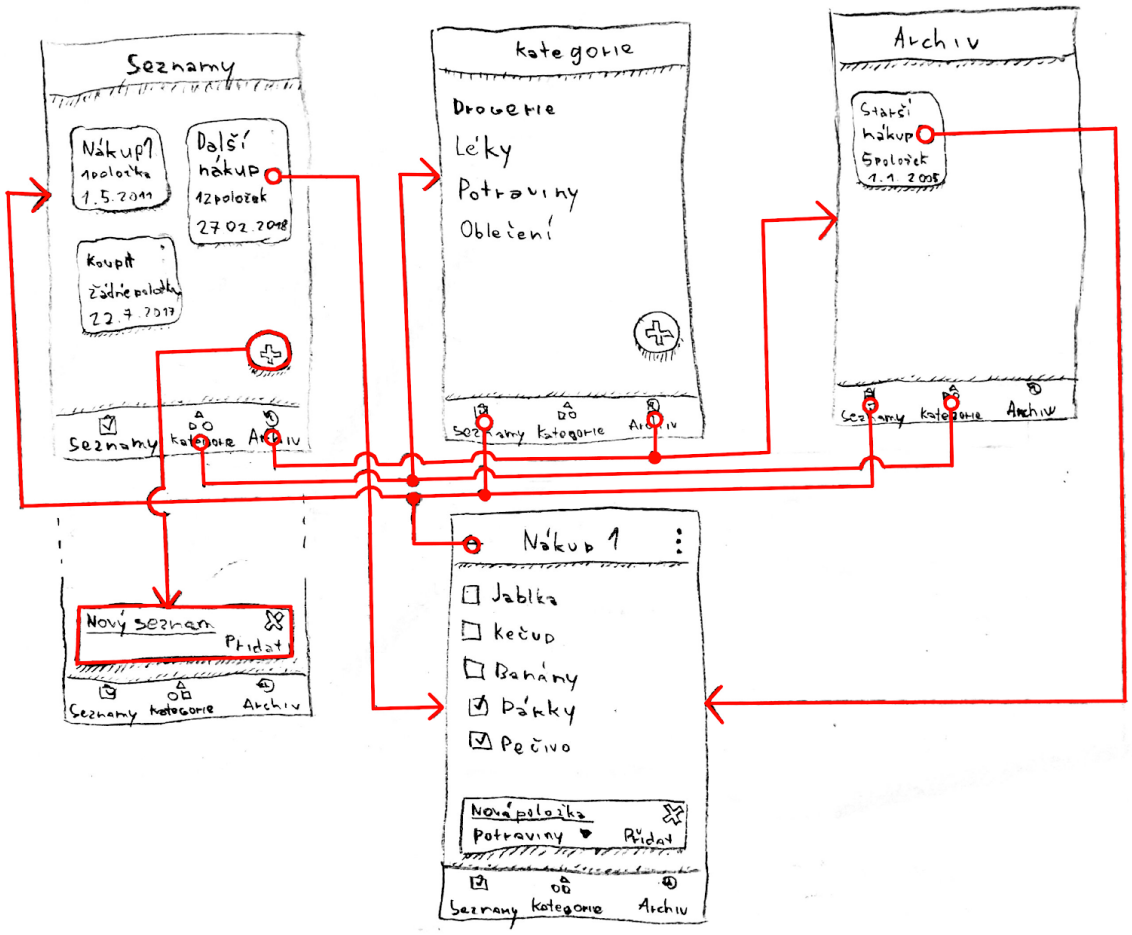
K výběru nejvhodnějšího z představených návrhů jsem využil metody vícekriteriální analýzy variant. Definoval jsem si čtyři kritéria, první z nich je dosažitelnost palcem, dalšími kritérii je přehlednost menu a přehlednost obsahu obrazovky. Posledním kritéria je jednoduchost interakcí, toto kritérium hodnotí návrhy podle počtu a náročnosti interakcí uživatele pro dosažení určitého cíle. Kritériím jsem přiřadil váhy podle toho, jak je považuji za důležité, jako nejdůležitější kritérium jsem zvolil dosažitelnost palcem. Při hodnocení návrhů jsem pečlivě zvážil všechna kritéria a návrhy ohodnotil pomocí bodovací metody. Pro vypočtení kompromisní varianty jsem provedl skalární součin. Jednotlivé body jsem vynásobil vahami a sečetl. Nejlepšího výsledku dosáhl první návrh, který jsem proto použil ve výsledné aplikaci. Obodování návrhů podle kritérií a výpočet kompromisní varianty je viditelný z následující tabulky.

	Dosažitelnost palcem	Přehlednost menu	Přehlednost obsahu obrazovky	Jednoduchost interakcí	Skalární součin
Návrh 1	10	10	7	8	**9,05**
Návrh 2	8	5	5	10	**7,45**
Návrh 3	5	2	10	6	**5,4**
Váha kritéria	0,4	0,2	0,15	0,25	

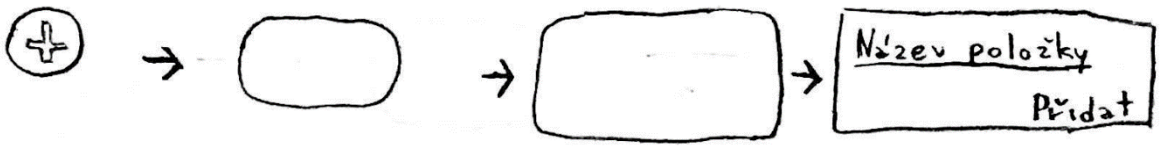
Tabulka č. 2. Vícekriteriální analýza variant (autor Matouš Vanča)

4.3 Drátový model

Drátový model slouží jako první vizuální návrh aplikace, lze podle něj odhadnout výslednou podobu a fungování aplikace. Na následujícím obrázku je viditelné, jak se uživatel může pohybovat aplikací. Jsou zde tři hlavní obrazovky: seznamy, kategorie a archiv a jedna vedlejší, a to detail seznamu. Pro přechod mezi obrazovkami zde slouží spodní menu. Do detailu seznamu se uživatel dostane kliknutím na kartu seznamu a zpět se dostane pomocí šipky. Obrazovky seznamy a archiv obsahují karty se seznamy, obrazovka kategorie výpis kategorií. Obrazovka detail seznamu se dá otevřít z obrazovky se seznamy a z archivu. Zvláštností je zde tlačítko pro přidávání obsahu, je navržené jako rozbalovací a jeho podoby jsou viditelné z první obrazovky nebo z obrázku, který ukazuje návrh animace tohoto tlačítka.



Obrázek č. 32. Drátový model (autor Matouš Vanča)

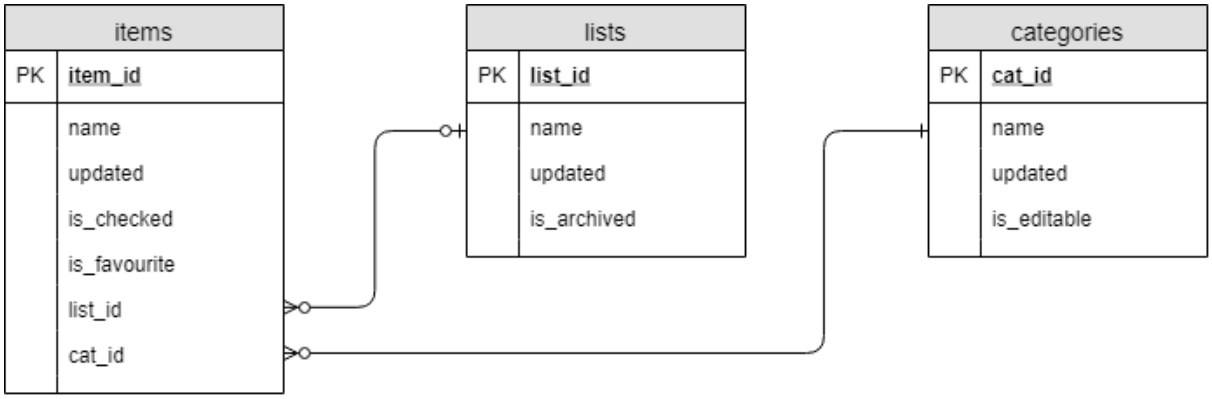


Obrázek č. 33. Návrh animace tlačítka (autor Matouš Vanča)

4.4 Datový model

Datový model znázorňuje vztah entit uložených v databázi aplikace. Ta má tři tabulky: seznamy (lists), kategorie (categories) a položky (items). Mezi položkami a seznamy platí následující vztah: více, jedna nebo žádná položka může patřit do jednoho nebo žádného

seznamu a jeden nebo žádný seznam může mít žádnou, jednu nebo více položek. Pro položky a kategorie platí tento vztah: více, jedna nebo žádná položka patří právě do jedné kategorie a jedna kategorie má více, jednu nebo žádnou položku. Vztahy jsou znázorněny na následujícím ER diagramu.



Obrázek č. 34. Datový model (autor Matouš Vanča)

4.5 Použité knihovny

Ve svém projektu využívám následující knihovny, většina z nich pochází z kolekce Android Jetpack. Tato kolekce vznikla za účelem usnadnění vytváření aplikací, dodržování osvědčených postupů a zjednodušení kódu. Android Jetpack obsahuje knihovny, které jsou odděleny od Api platformy, a proto nabízí lepší zpětnou kompatibilitu a jsou aktualizovány častěji (46).

Pro data binding, architekturu aplikace a práci s databází používám knihovny ze sady Android Jetpack, což mi díky knihovnám Data Binding, ViewModel, LiveData a Room umožnilo dokonalé propojení dat s uživatelským rozhraním. Knihovnu ViewModel používám pro implementaci prostřední části architektury MVVM, ve které využívám třídu ViewModel. Ta je navržena pro skladování a použití dat z uživatelského rozhraní a zároveň umí pracovat s životním cyklem prvků rozhraní. LiveData je knihovna, která mi umožňuje použít stejnojmennou třídu pro manipulaci s daty aplikace. Data v této třídě jsou pozorovatelná a nezávislá na životním cyklu. Pro správu dat v databázi využívám knihovnu Room. Tato knihovna vytváří abstrakční vrstvu nad databází SQLite pro robustnější a efektivnější přístup do databáze. Díky knihovně Room mohu data z databáze dostávat rovnou ve formě LiveData, což umožňuje přímé napojení na mnou zvolenou architekturu (47), (48), (49).

Knihovnu Navigation využívám pro přechody mezi obrazovkami v aplikaci, správu zásobníku a pro zpětnou navigaci. Knihovna umožňuje navrhnout navigaci mezi fragmenty a aktivitami, které reprezentují jednotlivé obrazovky graficky, a také umožňuje definovat argumenty předávané mezi fragmenty nebo aktivitami. Navigační diagram se ukládá do zdrojového souboru v XML jazyce a je podle něj knihovnou vygenerován kód.

WorkManager slouží k provedení delších operací na pozadí. Knihovna si sama poradí, kdy a jakým způsobem operaci provést na základě verze systému na zařízení. Ve své aplikaci využívám tuto knihovnu pro zpracování a uložení výchozích dat dostupných v aplikaci. Ke zpracování dat využívám další z knihoven, jedná se o knihovnu GSON, která umí převádět data uložená ve formátu json do příslušného objektu. Data převedená ze zdrojového json souboru, nyní reprezentovaná objektem, jsou následně uložena do databáze (50).

Další použitou knihovnou je knihovna Anko, která obsahuje spoustu užitečných funkcí určených především k usnadnění programování a k redukování potřebného kódu. Využil jsem především funkce pro práci s vlákny a uživatelským rozhraním.

AppCompat je sada podpůrných knihoven, které zajišťují kompatibilitu nových funkcí a vizuálních prvků pro starší verze Androidu.

Knihovny CardView, RecyclerView a ConstraintLayout umožňují použití dalších prvků pro tvorbu uživatelského rozhraní. CardView je stylový prvek vhodný do seznamů pro zobrazení dat. CardView mají ve výchozím stavu nastavenou výšku, systém tak vykresluje pod tímto prvkem stín, a má podporu zaoblených rohů (51). RecyclerView je kontejner pro zobrazení velkého množství prvků v seznamu, mřížce nebo jiném rozložení. Hodí se pro velké množství dat, protože recykluje vykreslené prvky v případě, že se dostanou mimo obrazovku. Recyklované prvky pak znovu použije, když má dojít k jejich zobrazení. RecyclerView se umí automaticky postarat o změny v datech, které vykresluje, například dojde-li ke změně pořadí prvků, RecyclerView je automaticky seřadí (52). ConstraintLayout je kontejner pro velká a komplexní uživatelská rozhraní. Umožňuje tvořit plochou hierarchii, což znamená, že lze prvky v něm poskládat do výsledné podoby tak, že nemusí žádný z prvků obsahovat další vložené prvky. Prvky v ConstraintLayout jsou mezi sebou spojeny vazbami, které mohou být různorodého typu. Editor v Android Studiu podporuje použití ConstraintLayout a všech jeho funkcí (53).

5 Závěr

Výsledkem bakalářské práce je plně funkční aplikace pro operační systém Android a souhrn teoretických východisek. Aplikace by měla svou jednoduchostí a praktičností nahradit papírové nákupní seznamy. Při její tvorbě byl využit návrhový vzor Model-View-ViewModel (MVVM), byly také dodrženy principy vícevrstvé architektury.

Jako uvedení do teoretického přehledu byl představen operační systém Android, jeho architektura a jedna kapitola byla věnována historii jeho vývoje. Další část teorie se týkala Android aplikací, především jejich součástí. Následně byla představena vícevrstvá architektura a popsány vzory jejího použití. Podrobně byla také vysvětlena technika data bindingu. Další část se věnovala návrhu uživatelských rozhraní a designu. V poslední části bylo popsáno vývojové prostředí Android Studio, nástroje použité k vývoji a použité programovací jazyky.

V praktické části byly vytyčeny požadavky a podle nich vypracovány návrhy uživatelského rozhraní aplikace. Z nich byl pomocí metody vícekriteriální analýzy variant vybrán nejvhodnější návrh, a podle něj zpracován drátový a datový model aplikace. Tyto modely následně sloužily k implementaci výsledné aplikace. Na závěr byly popsány knihovny využité při implementaci.

Přestože existuje již značné množství aplikací sloužících jako nákupní seznam, rozhodl jsem se vytvořit aplikaci se stejným využitím, která by však oproti mnoha ostatním byla praktičtější, přehlednější a jednodušší. Kladl jsem proto při tvorbě důraz na jednoduché ovládání, které spočívá především v tom, že lze aplikaci pohodlně ovládat jedním palcem. Přehlednosti jsem docílil dodržením principů Material Designu. Jednoduchosti jsem se snažil dosáhnout tím, že jsem se soustředil na nejdůležitější funkce, které uživatel využije. Aplikace sice nabízí rozšiřující funkce, jako jsou kategorie nebo oblíbené položky, ale tyto funkce nečiní aplikaci při ovládání složitější.

Výslednou aplikaci jsem publikoval do obchodu Google Play, kde je veřejně dostupná ke stažení. V budoucnu zde budou také dostupné novější verze aplikace, v nichž mám v plánu aplikaci dále rozvíjet. Další vývoj bych rád zaměřil na sdílení a synchronizaci dat. Aplikace by umožňovala uživateli se přihlásit, tím pádem by se data z aplikace synchronizovala do online databáze. Při přihlášení se na vícero zařízení by se data synchronizovala mezi nimi. Dále by bylo možné seznam poslat jinému uživateli, nebo jej s

ním trvale sdílet. Jednou z dalších změn by mohla být lokalizace aplikace do různých jazyků.

6 Seznam použitých zdrojů

(1) Platform Architecture. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/guide/platform/

(2) CALLAHAM, John. The history of Android OS: its name, origin and more. Android Authority [online]. 2018 [cit. 2019-02-19]. Dostupné z: https://www.androidauthority.com/history-android-os-name-789433/

(3) Mobile Operating System Market Share Europe. StatCounter: GlobalStats [online]. 2019 [cit. 2019-02-19]. Dostupné z: http://gs.statcounter.com/

(4) Distribution dashboard. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/about/dashboards/

(5) Application Fundamentals. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/guide/components/fundamentals

(6) Services overview. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/guide/components/services

(7) Content provider basics. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/guide/topics/providers/content-provider-basics

(8) Android Clean Architecture Components Boilerplate. Github [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://github.com/bufferapp/clean-architecture-components-boilerplate#android-clean-architecture-components-boilerplate

(9) BIRCH, Joe. Converting an App to Use Clean Architecture [online]. 2017 [cit. 2019-02-19]. Dostupné z: https://academy.realm.io/posts/converting-an-app-to-use-clean-architecture/

(10) KARPOUZIS, Thanos. Android Architecture: A simple guide for MVC, MVP and MVVM on Android projects [online]. 2015 [cit. 2019-03-10]. Dostupné z: https://android.jlelse.eu/android-architecture-2f12e1c7d4db?gi=8afa30fa11cc

(11) SANOGUERA DE LORENZO, Mario. Clean Architecture Guide [online]. 2018 [cit. 2019-03-10]. Dostupné z: https://proandroiddev.com/clean-architecture-data-flow-dependency-rule-615ffdd79e29vlozit

(12) *Android 6.0 Marshmallow* [online]. [cit. 2019-03-10]. Dostupné z: https://www.android.com/versions/marshmallow-6-0/

(13) JOHNSON, Paul. Using MVVM Light with Your Xamarin Apps [online].

Berkeley, CA: Apress L. P, 2017;2018; ISBN 9781484224748; ISBN 1484224744

(14) DAJBYCH, Václav. MVVM: model-view-viewmodel. DOTNETPORTAL.CZ[online]. 2014 [cit. 2019-02-19]. Dostupné z: https://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel

(15) ALCÉRRECA, Jose. ViewModels and LiveData: Patterns + AntiPatterns. Medium Corporation [online]. 2017 [cit. 2019-02-19]. Dostupné z: https://medium.com/androiddevelopers/viewmodels-and-livedata-patterns-antipatterns-21efaef74a54

(16) SALEH, Hazem. MVVM architecture, ViewModel and LiveData (Part 1). ProAndroidDev [online]. 31.5.2017 [cit. 2019-03-03]. Dostupné z: https://proandroiddev.com/mvvm-architecture-viewmodel-and-livedata-part-1-604f50cda1

(17) LiveData Overview. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/topic/libraries/architecture/livedata

(18) Data Binding Library. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/topic/libraries/data-binding/

(19) Two-way data binding. Android Developers [online]. 2019 [cit. 2019-03-03]. Dostupné z: https://developer.android.com/topic/libraries/data-binding/two-way

(20) NEDBÁLEK, Stanislav. Uživatelské rozhraní mobilních aplikací. Brno, 2016. Diplomová práce. Masarykova univerzita. Vedoucí práce doc. RNDr. Petr Sojka, Ph.D.

(21) HOOBER, Steven. How Do Users Really Hold Mobile Devices?: Mobile Matters. UXmatters [online]. 2013 [cit. 2019-02-19]. Dostupné z: https://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php

(22) NIELSEN, Jacob. Usability 101: Introduction to Usability. Log in to UX Certification Search Nielsen Norman Group logoNielsen Norman Group: World Leaders in Research-Based User Experience [online]. 4.1.2012 [cit. 2019-03-03]. Dostupné z: https://www.nngroup.com/articles/usability-101-introduction-to-usability/

(23) DesignBytes: Paper and Ink: The Materials that Matter. In: Youtube [online]. 27.02.2007 [cit. 2019-03-11]. Dostupné z: https://www.youtube.com/watch?v=YaG_ljfzeUw. Kanál uživatele Google Developers.

(24) Material Design: Introduction. Material Design [online]. [cit. 2019-02-19]. Dostupné z: https://material.io/archive/guidelines/

(25) BEČICA, Michal. Flat Design vs. Material Design - kdo je král?. WDT [online]. 2018 [cit. 2019-02-19]. Dostupné z: https://www.wdt.cz/novinky/flat-design-vs-material-design-kdo-je-kral-a5b34922e7884f507bba09946

(26) App bars: top: The top app bar displays information and actions relating to the current screen. [online]. [cit. 2019-03-03]. Dostupné z: https://material.io/design/components/app-bars-top.html

(27) Navigation Drawer [online]. [cit. 2019-03-03]. Dostupné z: https://material.io/design/components/navigation-drawer.html

(28) Tabs [online]. [cit. 2019-03-03]. Dostupné z: https://material.io/design/components/tabs.html

(29) Buttons: floating action button [online]. [cit. 2019-03-03]. Dostupné z: https://material.io/design/components/buttons-floating-action-button.html

(30) Dropdown menu [online]. [cit. 2019-03-03]. Dostupné z: https://material.io/design/components/menus.html#dropdown-menu

(31) Menus [online]. [cit. 2019-03-03]. Dostupné z: https://material.io/design/components/menus.html

(32) Cards [online]. [cit. 2019-03-03]. Dostupné z: https://material.io/design/components/cards.html

(33) Lists. [online]. [cit. 2019-03-03]. Dostupné z: https://material.io/design/components/lists.html

(34) Android Studio release notes. Android Developers [online]. 2019, 2019 [cit. 2019-03-03]. Dostupné z: https://developer.android.com/studio/releases

(35) Build a UI with Layout Editor. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/studio/write/layout-editor

(36) Analyze your build with APK Analyzer. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/studio/build/apk-analyzer

(37) Measure app performance with Android Profiler. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/studio/profile/android-profiler

(38) KOTAČKA, Vít. Gradle, moderní nástroj na automatizaci. Zdroják [online]. 2013 [cit. 2019-02-19]. Dostupné z: https://www.zdrojak.cz/clanky/gradle-moderni-nastroj-na-automatizaci/

(39) SINICKY, Adam. Android SDK tutorial for beginners. Android Authority [online]. 2017 [cit. 2019-02-19]. Dostupné z: https://www.androidauthority.com/android-sdk-tutorial-beginners-634376/

(40) KODYTEK, Samuel. Lekce 1 - Úvod do jazyka Kotlin, platformy a IntelliJ. ITnetwork.cz [online]. [cit. 2019-02-19]. Dostupné z: https://www.itnetwork.cz/kotlin/zaklady/uvod-do-jazyka-kotlin-platformy-a-intellij

(41) SOUHRADA, Václav. KOTLIN – JAZYK, KTERÝ JE DOBRÉ ZNÁT. Eman.cz [online]. 2017 [cit. 2019-02-19]. Dostupné z: https://www.eman.cz/blog/kotlin-jazyk-ktery-dobre-znat/

(42) XML in Android: Basics And Different XML Files Used In Android. AbhiAndroid [online]. 2018 [cit. 2019-02-19]. Dostupné z: https://abhiandroid.com/ui/xml

(43) WELLING, Luke a Laura THOMSON. MySQL: průvodce základy databázového systému. Brno: CP Books, 2005. ISBN 8025106713.

(44) CROCKFORD, Douglas. Introducing JSON [online]. [cit. 2019-02-19]. Dostupné z: https://www.json.org/

(45) VALKOVIC, Patrik. Lekce 1 - Git – Historie a principy. ITnetwork.cz [online]. 2014 [cit. 2019-02-19]. Dostupné z: https://www.itnetwork.cz/software/git/git-tutorial-historie-a-principy

(46) Android Jetpack. Android Developers [online]. 2019, 2019 [cit. 2019-03-03]. Dostupné z: https://developer.android.com/jetpack

(47) ViewModel Overview. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/topic/libraries/architecture/viewmodel

(48) LiveData Overview. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/topic/libraries/architecture/livedata

(49) Room Persistence Library. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/topic/libraries/architecture/room

(50) Schedule tasks with WorkManager. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/topic/libraries/architecture/workmanager/

(51) Create a Card-Based Layout. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/guide/topics/ui/layout/cardview

(52) Create a List with RecyclerView. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/guide/topics/ui/layout/recyclerview

(53) Build a Responsive UI with ConstraintLayout. Android Developers [online]. 2019 [cit. 2019-02-19]. Dostupné z: https://developer.android.com/training/constraint-layout/

(54) JURČÍKOVÁ, Iveta. *Aplikace architektonických vzorů při vývoji mobilní aplikace pro Android* [online]. Brno, 2017 [cit. 2019-03-04]. Dostupné z: https://is.muni.cz/th/l7n3r/DP.pdf. Diplomová práce. Masarykova univerzita. Vedoucí práce Ing. RNDr. Barbora Bühnová, Ph.D.

(55) KILIÁN, Karel. Historie Androidu v kostce aneb Od verze 1.0 až po Android M. *Svět Androida* [online]. 2019, 2015 [cit. 2019-03-04]. Dostupné z: https://www.svetandroida.cz/historie-androidu/

(56) ALLEN, Grant. Android 4: průvodce programováním mobilních aplikací. Brno: Computer Press, 2013. ISBN 9788025137826.

(57) Industry Leaders Announce Open Platform for Mobile Devices. *Open Handset Alliance* [online]. 5. 11. 2007 [cit. 2019-03-10]. Dostupné z: http://www.openhandsetalliance.com/press_110507.html

(58) *Android 9 Pie* [online]. [cit. 2019-03-10]. Dostupné z: https://www.android.com/versions/pie-9-0/

(59) *Android - 8.0 Oreo* [online]. [cit. 2019-03-10]. Dostupné z: https://www.android.com/versions/oreo-8-0/

(60) *Android – Nougat* [online]. [cit. 2019-03-10]. Dostupné z: https://www.android.com/versions/nougat-7-0/

7 Seznam tabulek

8 Seznam obrázků

9 Seznam použitých zkratek

ADB = Android Debug Bridge

ART = Android Runtime

CDMA = Code Division Multiple Access

FAB = Floating Action Button

FLAC = Free Lossless Audio Codec

HEIF = High Efficiency Image File Format

HDR = High Dynamice Range

JSON = JavaScript Object Notation

MMS = Multimedia Messaging Service

MVC = Model-View-Controller

MVP = Model-View-Presenter

MVVM = Model-View-ViewModel

NFC = Near Field Communication

OTG = On-The-Go

LED = Light emitting diode

SDK = Software development kit

SQL = Structured Query Language

UI = User Interface

USB = Universal Serial Bus

VPN = Virtual Private Network

XML = eXtensible Markup Language