Mobilní aplikace pro predikci akciových kurzů

Adam Mitrenga

Bakalářská práce
2021

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Adam Mitrenga**
Osobní číslo:	**A18066**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Mobilní aplikace pro predikci akciových kurzů**
Téma práce anglicky:	**A Mobile Application for Stock Price Prediction**

Zásady pro vypracování

1. Nastudujte a popište problematiku spojenou akciovými trhy.
2. Rozeberte problematiku vývoje mobilních aplikací.
3. Zvolte vhodné technologie pro implementaci mobilní aplikace a predikci ceny akcií.
4. Proveďte implementaci vašeho řešení.
5. Vhodně vyhodnoťte a reprezentujte výsledky.

Forma zpracování bakalářské práce: **Tištěná/elektronická**

Seznam doporučené literatury:

1. ANNUZZI, Joseph, Lauren DARCEY a Shane CONDER. Advanced Android application development. Fourth edition. Upper Saddle River: Addison-Wesley, [2015], xxxi, 570 s. Developer's library series. ISBN 9780133892383.
2. SIEGEL, Jeremy J. Investice do akcií: běh na dlouhou trať. Praha: Grada, 2011, 295 s. Finance. ISBN 9788024738604.
3. ČERMÁK, Petr. Investice do akcií: základy value investování. Praha: Brána, 2018, 268 s. ISBN 9788075840684.
4. CIPRA, Tomáš. Matematika cenných papírů. Praha: Professional Publishing, 2013, 288 s. ISBN 9788074310799.

Vedoucí bakalářské práce: **Ing. Petr Žáček**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **15. ledna 2021**
Termín odevzdání bakalářské práce: **17. května 2021**

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

prof. Mgr. Roman Jašek, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 15. ledna 2021

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Adam Mitrenga v. r.
podpis studenta

ABSTRAKT

Tato bakalářská práce se zabývá vývojem nativní aplikace pro operační systém Android, vytvořením API serveru a implementací algoritmu pro predikce akciových trhů do aplikace. Práce je rozdělena do dvou částí. V teoretické části je představen akciový trh a popsán vývoj nativní aplikací. V Praktické části je popsán vývoj aplikace pro Android a funkcionalita této aplikace, vývoj API serveru a představen implementovaný algoritmus pro predikci akcií.

Klíčová slova: Android, API server, strojové učení, akciový trh

ABSTRACT

This bachelor thesis deals with the development of a native application for the Android operating system, the creation of an API server and the implementation of an algorithm for stock market prediction in the application. The work is divided into two parts. In the theoretical part, the stock market is introduced and the development of native applications is described. The practical part describes the development of an application for Android and the functionality of this application, the development of an API server and introduces an implemented algorithm for prediction of actions.

Keywords: Android, API server, machine learning, stock market

Na tomto místě bych rád poděkoval panu Ing. Petru Žáčkovi, za cenné rady a čas věnovaný konzultaci této práce.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Lidé nakupovali a prodávali akcie již od 17. století. S postupem času se finanční trhy vyvíjeli, přešli jsme z nutnosti kupovat fyzické akcie, na pohodlnější možnost kupovat akcie přes telefonní komunikaci s makléřem, až do dnešní doby, kde nákup a prodej akcií je otázka stisknutí pár tlačítek na mobilním zařízení nebo počítači. Mnoho investorů si chce vyzkoušet obchodování akcií nanečisto. Tato bakalářská práce umožňuje uživatelům aplikace vyzkoušet si nákup a prodej akcí, s možností podívat se na návratnost jejich investic. K rozhodování, zda prodat nebo koupit akcii mají uživatelé k dispozici základní informace o akcii, její historii, a nakonec možnost předpovídat cenu dané akcie.

Tato bakalářská práce je rozdělena do dvou částí: teoretické a praktické. V teoretické části jsou uvedeny základní informace o akciovém trhu a akciích. Dále je nastíněn vývoj nativních mobilních aplikací a architektur, které se využívají při jejich vývoji.

V praktické části jsou popsány požadavky na aplikaci, vývoj nativní aplikace pro Android v jazyku Java a její design, práce s MVVM modelem v Androidu. Dále je popsáno, jak byl vygenerován RESTful API serveru pomocí swagger.io. Představena jeho funkcionalita a databáze. Nakonec je popsán algoritmus, který byl implementován na server pro predikci ceny akcie.

I. TEORETICKÁ ČÁST

1 AKCIOVÝ TRH

V této kapitole je obecně představen akciový trh. Je zde krátce popsána jeho historie, základní informace o akciích, faktory pohybující cenou akcie, finanční ukazatele využívané při porovnávání akcií, představeny typy projektů predikcí, které byly zváženy k implementaci v této práci.

1.1 Peněžní Burza

Peněžní burza je druh instituce, kde mohou makléři a obchodníci nakupovat a prodávat finanční prostředky, například tedy akcie vydané veřejně obchodovanými společnostmi, podílové fondy, deriváty, investiční produkty a dluhopisy. Burzy cenných papírů mohou také poskytovat prostředky pro vydávání a odkup těchto cenných papírů, jako například výplaty výnosů a dividend. Obvykle existuje centrální umístění alespoň pro vedení záznamů, avšak obchod je stále více prováděn online, protože moderní trhy využívají sítě elektronických komunikací, což jim přináší výhody vyšší rychlosti a snížení nákladů na transakce. Obchod na těchto burzách je omezen na makléře, kteří jsou členy dané burzy.

Dnes má téměř každá země burzu a každý den se na nich obchoduje s miliony dolarů. Největší burza je v Spojených státech amerických – NYSE (New York Stock Exchange). Tato burza byla založena 17. května 1792. Její tržní kapitalizace je přes 24 trilionů dolarů. [1]

1.2 Akcie

Akcie jsou cenné papíry, které představují vlastnictví zlomku společnosti. To opravňuje vlastníka akcií k podílu aktiv společnosti a ziskům rovným tomu, kolik akcií vlastní.

1.2.1 Cena akcie

Cena akcie je poháněna nabídkou a poptávkou. Když je akcie prodána, tak cena, za kterou se akcie nakupuje, se stává novou tržní cenou. Při prodeji druhé akcie se tato cena stane nejnovější tržní cenou. Čím větší je poptávka po akciích, tím vyšší je její cena a vice versa.

1.2.2 Faktory ovlivňující chování akcí

V této kapitole jsou popsány pouze některé faktory, které mohou ovlivnit cenu akcií. Změny cen akcií jsou většinou způsobeny vnějšími faktory, jako jsou socioekonomické podmínky, inflace a směnné kurzy…[2]

1.2.2.1 Poptávka a nabídka

Jedním z hlavních faktorů ovlivňující cenu akcie jsou poptávka a nabídka. Když poptávka po akciích převyšuje nabídku, což znamená, že kupujících je více než prodávající, ceny rostou. Pokud je poptávka menší než nabídka, což znamená, že kupujících je méně než prodejců, ceny klesají.

1.2.2.2 Dividendy

Dividendy mohou naznačit pohyby cen akcií. Když společnosti uskuteční oznámení o dividendách, ceny akcií těchto společností se mohou zvýšit. Naskytuje se zde také možnost, že pokud jsou dividendy nižší než očekávání investorů, ceny akcií většinou klesají.

1.2.2.3 Vedení společnosti

Management má významný vliv na úspěch společnosti a následně také ceny akcií společnosti. Pokud se vedení skládá ze zkušených profesionálů s prokazatelnými výsledky, ceny akcií budou pravděpodobně vyšší. Pokud vedení, které převezme společnost, postrádá integritu, ceny akcií mají tendenci klesat. Jako příklad zde můžeme uvést významnou společnost Tesla, která kdyby měla ve vedení kohokoliv jiného, než Elon Muska, tak by s největší pravděpodobností nebyla tak vysoce hodnocena. Z toho vyplývá, že investor do společnosti může investovat i na základě vedení společnosti.

1.2.2.4 Zpráva o příjmech

Každá veřejně obchodovaná společnost musí zveřejňovat každé čtvrtletí zprávu o příjmech a nákladech společnosti. Tato čtvrtletní zpráva musí obsahovat hrubé výnosy společnosti, čistý zisk, provozní náklady a cash-flow. Často také společnosti přidávají zprávu o tom, jaké měli v tomto čtvrtletí problémy a příležitosti. Kromě čtvrtletních zpráv musí společnost vydávat jednoročně Výroční zprávu, kde jsou všechna čtvrtletí sečteny. Zde je uveden také propagační materiál společnosti. [3]

Celkově jsou jak čtvrtletní zprávy, tak celoroční zprávy velice anticipované a jsou předmětem velkých spekulací pro akciový trh. Pokud má firma dobré zprávy a překonala očekávání analytiků, mělo by se to odrazit v ceně akcie.

1.2.2.5 Politické faktory

Ceny akcií mohou ovlivnit politické faktory, které sahají od vztahů s jinými národy až po vládní politiku. Například vláda se může rozhodnout, že zvýší zdanění výnosu z prodeje akcii, toto rozhodnutí může negativně ovlivnit celý trh.

1.3 Finanční ukazatele – terminologie

V této kapitole jsou popsány nejznámější finanční ukazatele, které pomáhají investorům při rozhodování. Co tyto ukazatele vykazují? Přidávají reálná čísla, která se dají porovnávat mezi ostatními firmami například ve stejném odvětví.

1.3.1 Výdělek za akcii

Zisk za akcii (EPS) se vypočítá jako zisk společnosti, dělen počtem akcií. Tento indikátor udává tedy číslo, jak je společnost výdělečná. [4]

$$EPS = \frac{\text{Čistý zisk společnosti} - \text{Preferenční dividendy}}{\text{Počet akcií}}$$

1.3.2 Cena za poměr příjmů

Cena za poměr příjmů nebo ve zkratce P/E je ukazatel, který přemění zisky společnosti do jednoho snadně srovnatelného čísla. Říká investorům, kolik aktuálně ostatní investoři platí za každou měnovou jednotku, kterou společnost vydělá. [5]

$$P/E = \frac{\text{Tržní hodnota akcie}}{EPS}$$

Například společnost AMD má P/E = 33,09, to znamená, že za každý dolar, co společnost vydělá, platí investoři 33,09 dolarů. Toto může signalizovat, že je například společnost předražená, a nebo také, že je od společnosti očekáván velký růst.

1.3.3 Poměr dluhu ke vlastnímu kapitálu

Poměr dluhu k vlastnímu kapitálu (D/E) je měřítko míry, jak společnost financuje své operace. Jestli je financuje prostřednictvím dluhu nebo pomocí vlastního kapitálu. Tento ukazatel odráží schopnost společnosti splatit nesplacené dluhy. [6]

$$D/E = \frac{\text{dluh}}{\text{vlastní kapitál}}$$

1.3.4 Volný tok peněz

Volný tok peněz nebo také free cash flow (FCF) reprezentuje hotovost, kterou společnost generuje po zaúčtování příjmů a výdajů na podporu operací společnosti.

1.3.5 PEG

PEG neboli cena za poměr příjmu k růstu společnosti. Popisuje relativní cenu mezi cenou akcie, výdělkem za akcii (EPS) a očekávaným růstem dané společnosti.

$$PEG = \frac{Cena/EPS}{Roční\ EPS\ růst}$$

PEG se často používá často s kombinací EPS a to například, když u firmy je EPS nízký, což signalizuje dobrou hodnotu nákupu, tak s PEG lze zjistit, že například společnost se nijak nerozvíjí a stagnuje.[7]

1.4 Makléř

Makléř je fyzická nebo právnická osoba, která působí jako prostředník mezi investorem a burzou. Jelikož burzy cenných papírů přijímají objednávky pouze od fyzických osob nebo společností, které jsou členy dané burzy, jednotliví obchodníci a investoři potřebují služby členů burzy. Zprostředkovatelé poskytují tuto službu a jsou kompenzováni různými způsoby. Zpravidla se jedná o provize, poplatky nebo vyplácením samotnou burzou.

1.5 Predikce akcií

Zde jsou popsány dva typy projektů, které byly zváženy k implementaci v této práci do serveru.

1.5.1 Historické data

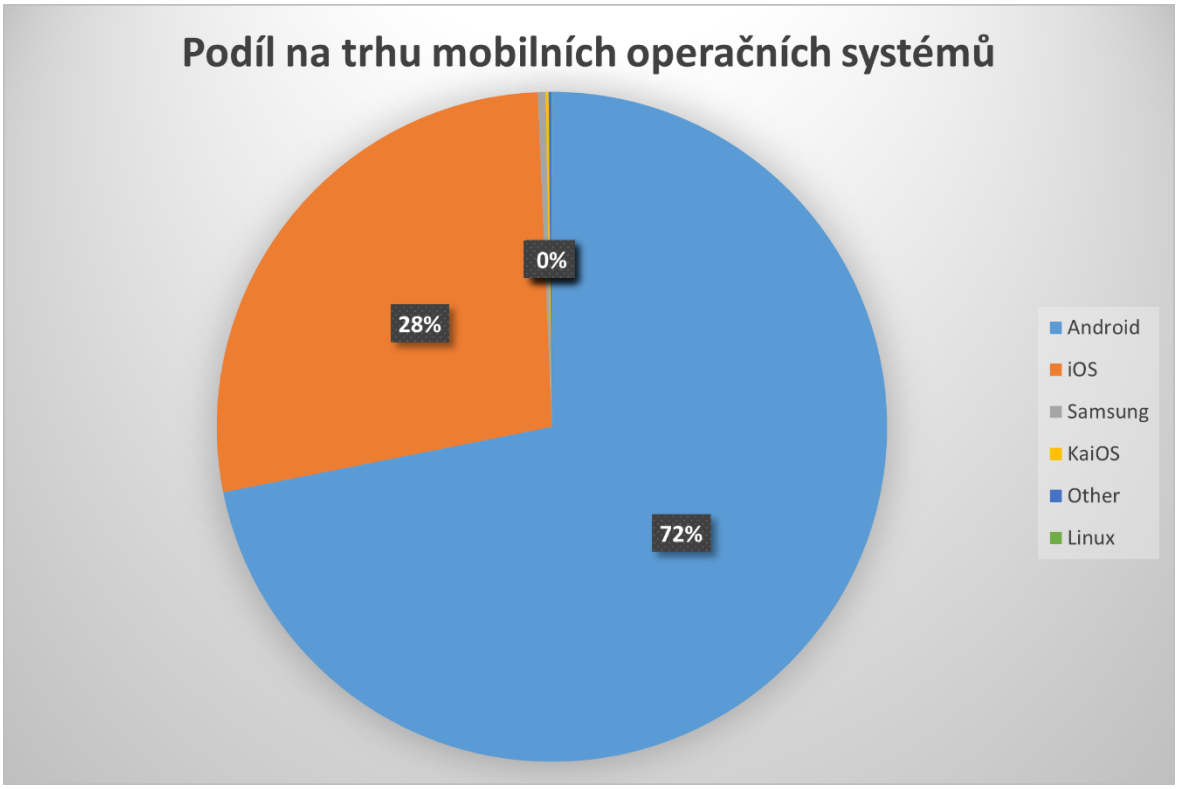
Projekty, které využívají historická data pro předpověď akcií, získávají data z API serverů nebo webscraperů. Získaná data jsou především historické ceny akcií a ty se liší od ceny, kdy se otevřel a zavřel trh, nejvyšší cena v daný den a jiné. Dále je využito umělé neuronové sítě RNN, nejpopulárnějšího v tomto poli LSTM, ale jsou využívány i jiné typy RNN, jako SimpleRNN nebo GRU, ze kterých je poté vytvořen model pro predikci. Tento model je natrénován, otestován, a nakonec je předpovězena cena.

1.5.2 Sentiment zpráv

V dnešní době je finanční trh velice dobře dokumentován a informace o akciích jsou sdíleny jednak přes profesionálně psané články a důvěryhodné zdroje, druhak i na sociálních sítích od entuziastů. Projekty, jež předpovídají cenu podle zpráv, využívají dolování textu z těchto zdrojů. Poté, co je text získán, je využito techniky zpracování přirozeného jazyka (NLP) a výpočetních technik, a to vše za cílem extrahovat sentiment daného textu. Následuje určení, zda text byl pozitivní, negativní nebo neutrální.

2 VÝVOJ NATIVNÍCH MOBILNÍCH APLIKACÍ

V této kapitole jsou popsány dva nejpopulárnější operační systémy pro vývoj nativních mobilních aplikací Android a iOS. Poté jsou popsány designové architektury, které se využívají při vývoji aplikací na těchto platformách. Je viditelné, že na trhu mobilních operačních trhů dominuje v dnešní době Android, ale to se může samozřejmě změnit. Zároveň by to neměl to být jediný faktor, který rozhoduje ve výběru, na kterém operačním systému vyvíjet aplikaci.



Obrázek 1 - Podíl na trhu mobilních operačních systému k datu 2021.3 [8]

2.1 Architektury pro vývoj aplikací

Základní představení tří designových vzorů pro vývoj aplikací s objektově orientovanými programovacími jazyky. Tyto architektury nejsou jediné a každá platforma má své specifické architektury nebo používá kombinaci, avšak uvedené tři architektury jsou rozhodně nejpopulárnější a nejuniverzálnější při vývoji jak mobilních, tak webových aplikací.

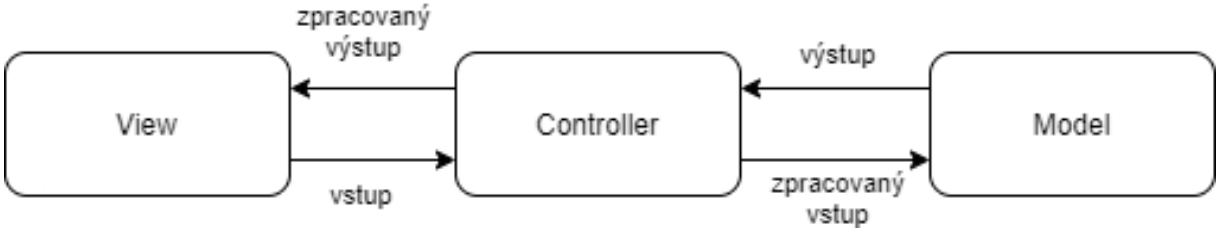
2.1.1 MVC

Model-View-Controller je designový vzor softwaru využívaný rovněž v nativních mobilních aplikacích, ale tak ve webových aplikacích. MVC spočívá v tom, že rozdělí aplikaci do tří logických komponentů.

Model – ukládá a spravuje data aplikace. Ukládání dat může být zprostředkováno například pomocí databáze.

View – komponent, který se stará o veškerou UI logiku.

Controller – komponent, který propojuje View a Model dohromady. Stará se o přesouvání a zpracování informací mezi uživatelem a modelem.



Obrázek 2 - Diagram MVC

Jedna z hlavních výhod využívání MVC je jeho zapouzdření a rozdělení různých částí aplikace. Přesto, že všechny komponenty s MVC spolu musí spolupracovat, jejich funkce jsou na sobě naprosto nezávazné. Díky tomu je MVC lehce škálovatelné a lépe testovatelné. A při spolupráci větších týmu na projektu umožňuje lehčí rozdělení práce. [9]

2.1.2 MVP

Model-View-Presenter je jedna z variací MVC, kde místo controlleru, který zpracovává jak změny v modelu, tak i změny pro view, jsou view a model rozděleny. Spolu poté komunikují jen přes prvek presenter, který v sobě obsahuje většinu business logic programu. Funkcionalita a zodpovědnost controlleru je tudíž rozdělena mezi view a presenter. Presenter zpracovává tok dat, odebírá business logic z controlleru a view se zaměřuje na zobrazování dat.

Obrázek 3 – Diagram MVP

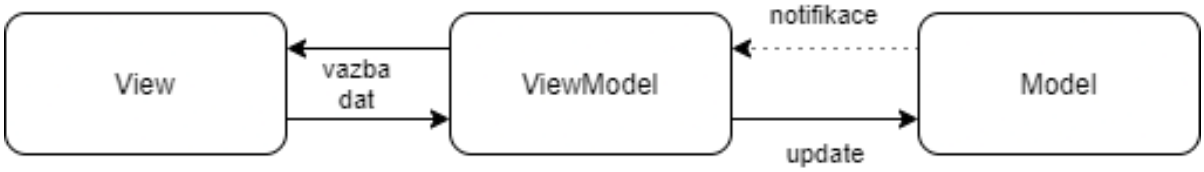
Model – stejně jako u MVC drží data aplikace v jednoduchých třídách a spravuje je.

View – zobrazuje prvky, s kterými interaguje uživatel.

Presenter – aktualizuje uživatelské rozhraní na základě změn v datovém modelu a také zpracovává vstupy uživatelů.[10]

2.1.3 MVVM

Model-View-ViewModel také známý pod názvem MVB pomáhá oddělit prezentační a programovou logiku od uživatelského rozhraní, a to díky třem hlavním komponentům model, view a viewmodel. Hlavním cílem MVVM je odstranit kód z UI části aplikace, a to pomocí vazby dat. Data s view jsou vázaný pomocí data bindingu na ViewModel, tím se úplně odděluje UI část od business logic. Toto ulehčuje i testování aplikace, jelikož jsou části kódu rozděleny. Například v Androidu je dokonce koncipováno, že VM může být flexibilní a nemusí odpovídat jednomu view.



Obrázek 4 - Diagram MVVM

Model – se nemění a stejně jak u předešlých dvou architektur drží data aplikace.

View – se také nemění ve funkcionalitě a zobrazuje prvky s kterými interaguje uživatel.

ViewModel – pracuje s informacemi z modelu, tak aby jejich finální forma mohla být vykreslena ve view. [11]

2.2 Android

Android je nejvíce používaný ze všech mobilních operačních systému dnešní doby viz Obrázek 1. Android je založen na Linux kernelu a je aktuálně vyvíjen firmou Google. Systém je primárně designovaný pro chytré telefony a tablety, ale v dnešní době je možné najít Android na téměř každém přenosném zařízení. Android také poskytuje vývojářům SDK, který obsahuje debugger, knihovny pro vývoj aplikací a emulátor mobilního zařízení. Google rovněž oficiálně vydává dokumentaci s ukázkovými kódy a tutoriály. Pro vývoj aplikací je možné vyžít jak programovací jazyk Java, tak i Kotlin. [12]

2.2.1 Historie a využití API levelu

Architektura mobilního operačního systému byla prvně vyvíjena firmou Android Inc. Android Inc. Byl v roce 2005 zakoupen firmou Google, který v roce 2007 spustil AOSP (Android Open Source Project). Od oficiálního vydání Google vydává minimálně jednu novou verzi systému Android každý rok. Změny a funkce nových verzí jsou v posledních letech ohlášeny na každoroční konferenci Google I/O.

Android Verze	Název	API Level	Datum Vydání
1.0-1.1	Žádný	1	23.9.2008
1.1	Žádný	2	9.2.2009
1.5	Cupcake	3	27.4.2009
1.6	Donut	4	15.2009
2.0-2.1	Eclair	5-7	26.10.2009
2.2-2.23	Froyo	8	20.5.2010
2.3-2.37	Gingerbread	9-10	6.12.2010
3.0-3.2.6	Honeycomb	11-13	22.2.2011
4.0-4.04	Ice Cream Sandwich	14-15	18.102011
4.1-4.3.1	Jelly Bean	16-18	9.7.2012
4.4-4.4.4	KitKat	19-20	31.10.2013
5.0-5.1.1	Lollipop	21-22	12.11.2014
6.0-6.0.1	Marshmallow	23	5.10.2015
7.0-7.1.2	Nougat	24-25	22.8.2016
8.0	Oreo	26	21.8.2017
8.1	Oreo	27	5.12.2017
9	Pie	28	6.8.2018
10	Android 10	29	3.9.2019
11	Android 11	30	8.9.2020
12	Android 12	31	TBA

Tabulka 1. Přehled verzí androidu

2.2.1.1 API level

API level je celočíselná hodnota, která označuje každou novou verzi Androidu a ulehčuje tak vyhledávání funkcionality. Pro vývojáře je mnohem lehčí se systémově ptát, jestli je tato aplikace podporovaná pro API level 3, než 4.04 nebo Ice Cream Sandwich. Každá verze platformy Android ukládá svůj identifikátor úrovně API interně do systému Android. [REF 19]

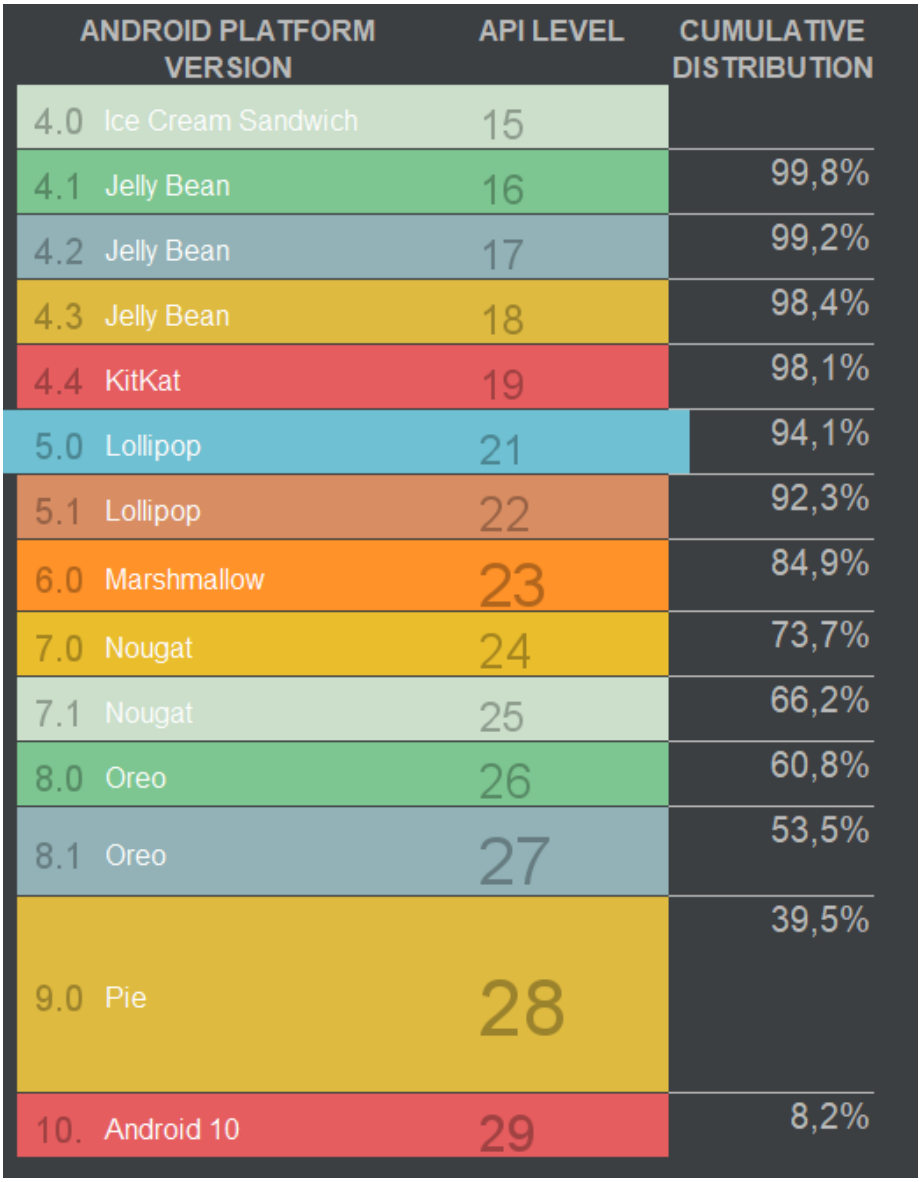
Identifikátor úrovně API hraje klíčovou roli také při zajišťování nejlepšího možného prostředí pro uživatele a vývojáře aplikací, například:

- Umožňuje platformě Android popsat maximální rozhraní API, kterou podporuje
- Umožňuje aplikacím popsat, jaké rozhraní API je potřeba pro korektní funkci
- Umožňuje systému rozpoznat, jakou verzi aplikace může nainstalovat, aby byla kompatibilní [13]

Při vývoji aplikace se dají nastavit tři hodnoty APK, které nám ulehčují vývoj specifické aplikace:

- android:minSdkVersion – minimální API level této aplikace například v tomto projektu je stanovené minimum Sdk Version 21, kdyby byla potencionálně tato aplikace vydána na Google Play obchod, tak uživatel, který má zařízení nižší než API level 21, ani aplikaci v obchodě neuvidí nebo mu ji nebude dovolené nainstalovat.
- android:targetSdkVersion – Určuje úroveň API, na které je aplikace navržena ke spuštění.
- android:maxSdkVersion – Určuje maximální úroveň API, na které je aplikace schopna fungovat. [13]

Z těchto indikátorů je rozhodně nejdůležitější si dávat pozor na minSdkVersion, který i při vývoji informuje vývojáře o tom, kdyby například využil knihovnu nebo funkcionalitu, která není přímo kompatibilní s minimálním vybraným levelem API.



Obrázek 5 – Distribuce API verzí v operačním systému Android

2.2.2 Základní komponenty aplikace

V této podkapitole jsou popsány základní komponenty aplikace, které musí nebo mohou být využity při vývoji aplikace. Je zde popsané k čemu slouží a jak se využívají.

2.2.2.1 Aktivita

Aktivita je vstupním bodem pro interakci s uživatelem. Aktivita se vyskytuje na jedné obrazovce. Aktivity mezi sebou nejsou závislé. Jakákoliv aktivita si může zavolat jinou aktivitu i z jiné aplikace. Například aplikace pro sociální síť si může zavolat aktivitu pro fotoaparát z jiné aplikace. [14]

Aktivita usnadňuje následující klíčové interakce mezi systémem a aplikací:

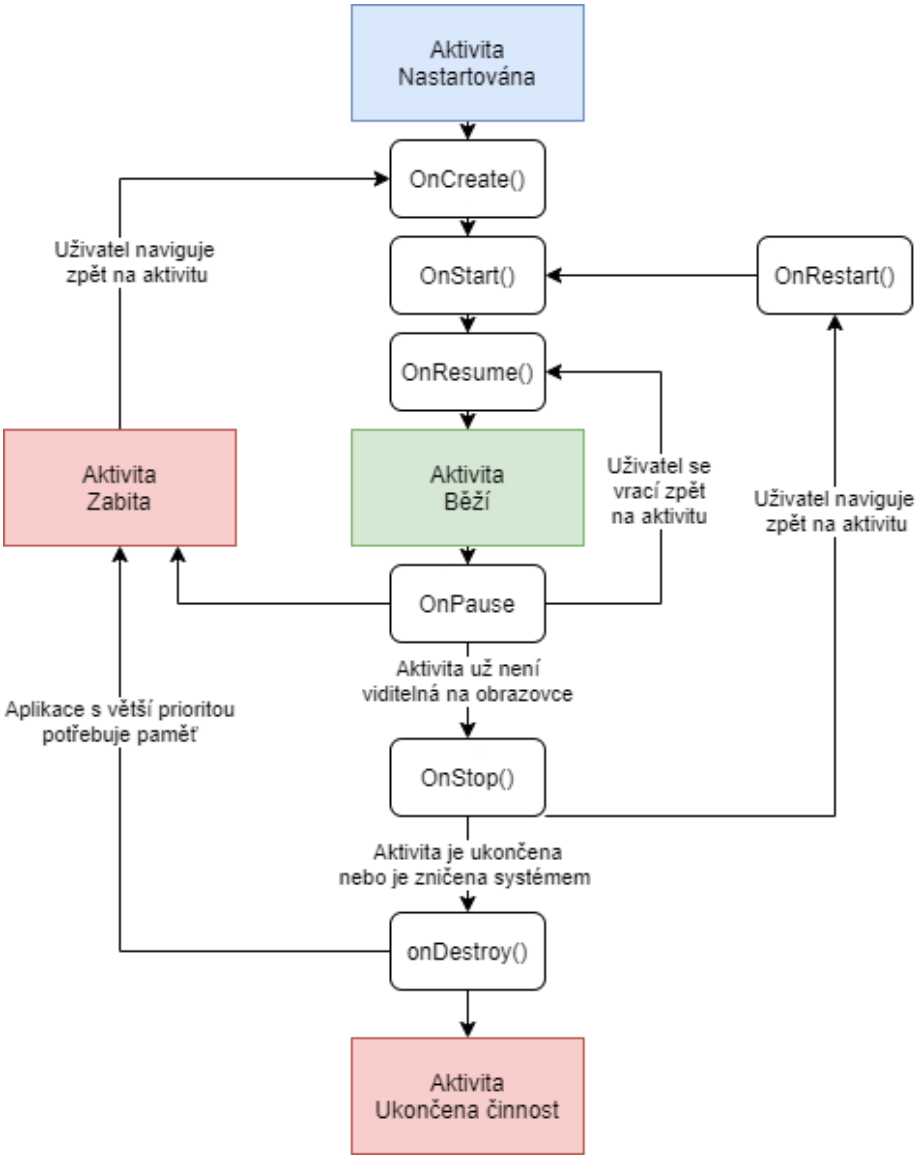
- Systém vždy ví, co aktuálně uživatel sleduje, proto může upřednostňovat aktivitu a tím pádem i aplikaci, kterou uživatel aktuálně sleduje a dodávat jí zdroje systému.
- S vědomím, že dříve používané procesy obsahují data, ke kterým se uživatel může vrátit, a tedy systém více upřednostňuje udržování těchto procesů.
- Sytém pomáhá aplikaci zvládnout zničení jejího procesu, tak aby se uživatel mohl vrátit k aktivitám s obnoveným předchozím stavem.
- Systém poskytuje způsob, jak může aplikace implementuje tok uživatelů v aplikaci. Jak lehce koordinovat cesty uživatele uvnitř aplikace, tak i mimo aplikaci. [14]

2.2.2.2 Životní cyklus aktivity

Při užívání aplikace se často děje, že uživatel nevyužívá pouze jednu aplikaci, ale přepíná do jiných aplikací, nebo přímo z aplikace, kterou aktuálně využívá, otevírá jinou aplikaci nebo aktivitu. Aktivita při interakci s uživatelem prochází různými stavy životního cyklu. Třída aktivity poskytuje zpětné volání, které umožňují aktivitě dát vědět, že se stav změnil. V rámci metod životního cyklu je možné deklarovat, jak se aktivita chová, když uživatel aktivitu opustí a znovu ji obnoví nebo například přijme hovor, přepne se do jiné aplikace, otočí obrazovku a při mnoho dalších situacích. K tomu jsou využívány níže vypsané zpětné volání. [15]

Třída Aktivity poskytuje šest základních zpětných zavolání (viz Obrázek 6):

- onCreate() – Zahájení aktivity.
- onStart() – Jakmile se aktivita dostane do stavu Zapnutá.
- onResume() - Dostává se do popředí.
- onPause() – Systém zavolá metodu při první indikaci, že uživatel opouští aktivitu.
- onStop() – Jakmile uživatel nevidí aktivitu.
- onDestroy() – Je zavolána předtím, než je aktivita zničena. [15]



Obrázek 6 – zjednodušená ilustrace životního cyklu aktivity [16]

2.2.2.3 Služby

Služba je obecný vstupní bod pro udržování aplikace spuštěné na pozadí Jedná se o komponentu, která také běží na pozadí. Služby byli vytvořeny za účelem provádění dlouhotrvajících operací nebo provádění práce pro vzdálené procesy. Služba může například přehrávat hudbu na pozadí, když je uživatel v jiné aplikaci, nebo může načítat data po síti, aniž by blokovala interakci uživatele s aktivitou. Služby můžou být spuštěny z jakékoliv další komponenty. existují dvě velmi odlišné služby, které systému říkají, jak spravovat aplikaci:

1. Spuštěné služby – tyto služby informují systém, aby je udržoval v provozu, dokud nebude dokončena jejich práce. Může to být synchronizace některých dat na pozadí nebo přehrávání hudby i poté, co uživatel aplikaci opustí.
2. Vázané služby – tyto služby běží, jelikož je využívá některá aplikace, která je na sebe naváže. Například pokud běží proces A, který je vázaný na procesu B systém neukončí proces B, dokud jej na sebe proces A má navázaný.[14]

2.2.2.4 Přijímač vysílání

Přijímač vysílání je komponenta, která umožňuje systému doručovat události i mimo běh aplikace. Například aplikace může naplánovat poplach třeba u aplikací monitorující cenu akcií, že pokud se cena akcie posune o určitou cenu, tak pošle poplach systému, který je poté zobrazen uživateli, i když nemá aplikaci spuštěnou. [14]

2.2.2.5 Poskytovatelé obsahu

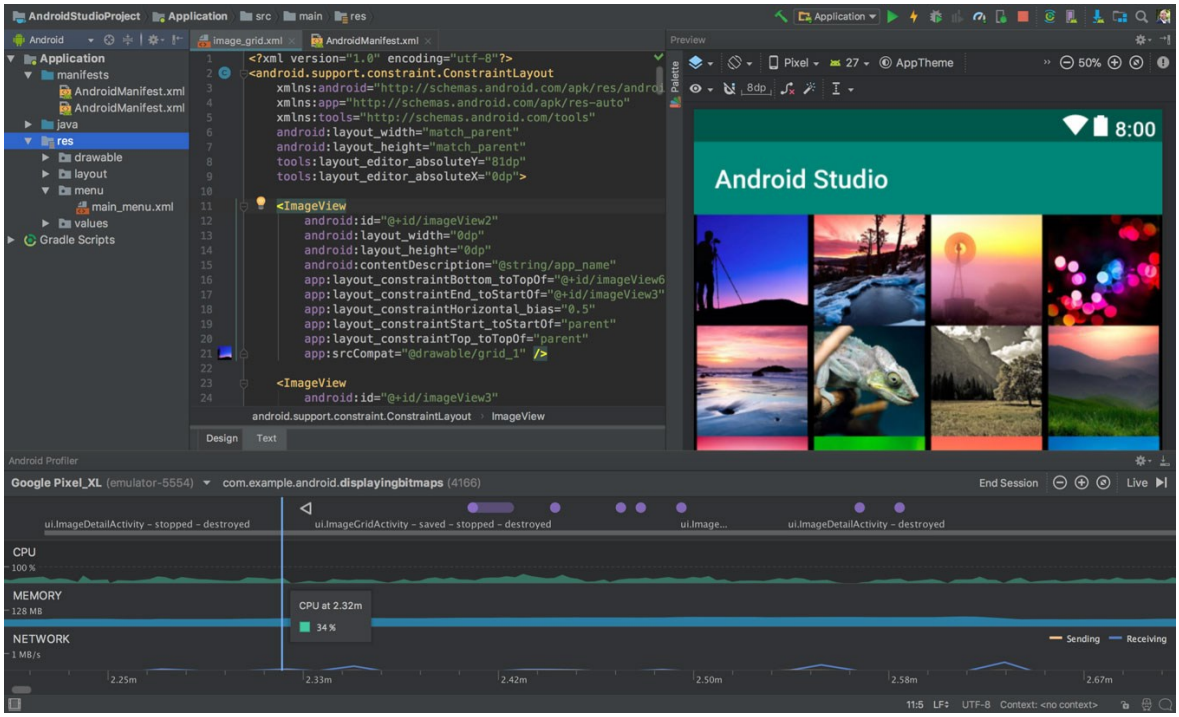
Komponent poskytovatel obsahu spravuje sdílená data aplikace. Pokud jsou tato data sdílena a aplikace má dostatečné povolení, může k nim přistoupit, a to i v databázi SQLite, na webu nebo na jakémkoli jiném trvalém umístění úložiště, ke kterému má aplikace přístup. Prostřednictvím poskytovatele obsahu mohou ostatní aplikace dotazovat nebo upravovat data, pokud to poskytovatel obsahu umožní. [14]

2.2.2.6 Manifest

V manifestu se deklarují komponenty aplikace a uživatelská oprávnění. Například v této bakalářské práci je ke korektní funkčnosti aplikace potřeba internet pro získávání informací o akciích, proto musí být v manifestu o povolení požádáno. Kromě této funkcionality se v manifestu deklaruje minimální úroveň API požadování aplikací, deklarují se knihovny API, a Hardwarové a softwarové vlastnosti, které aplikace používá.[14]

2.2.3 Android Studio

Android Studio je oficiální IDE pro vyvíjení aplikací pro operační systém Android. Je to plnohodnotné vývojové studio, ve kterém lze naprogramovat celou aplikaci od grafického návrhu, zdrojového kódu, až po ladění a kontrolování přiřazených zdrojů v reálném čase. Android Studio je podporováno pro Windows, Mac, Linux, Chrome OS. Je postaveno na vývojovém prostředí IntelliJ IDEA. S funkcemi cílenými pro vývoj nativních aplikací pro operační systém Android. Do Android Studia je zabudována přímo i kontrola verzí. [14]



Obrázek 7 – Náhled uživatelského rozhraní v Android Studiu [14]

2.2.3.1 Emulátor

Emulátor Androidu simuluje zařízení Androidu přímo v počítači, tím ulehčuje testování aplikace a umožňuje vývoj aplikací i bez fyzického zařízení, které má operační systém Android. Emulátor poskytuje téměř všechny funkce skutečného zařízení Android. Je možné simulovat telefonní hovory a textové zprávy, určit umístění zařízení, simulovat různé rychlosti sítě, simulovat rotaci a další hardwarové senzory, přistupovat k Obchodu Google Play a mnohem více. Je možné testovat, jak na mobilním zařízení, tak i na tabletu nebo chytrých hodinkách, a dokonce i na televizi. [17]

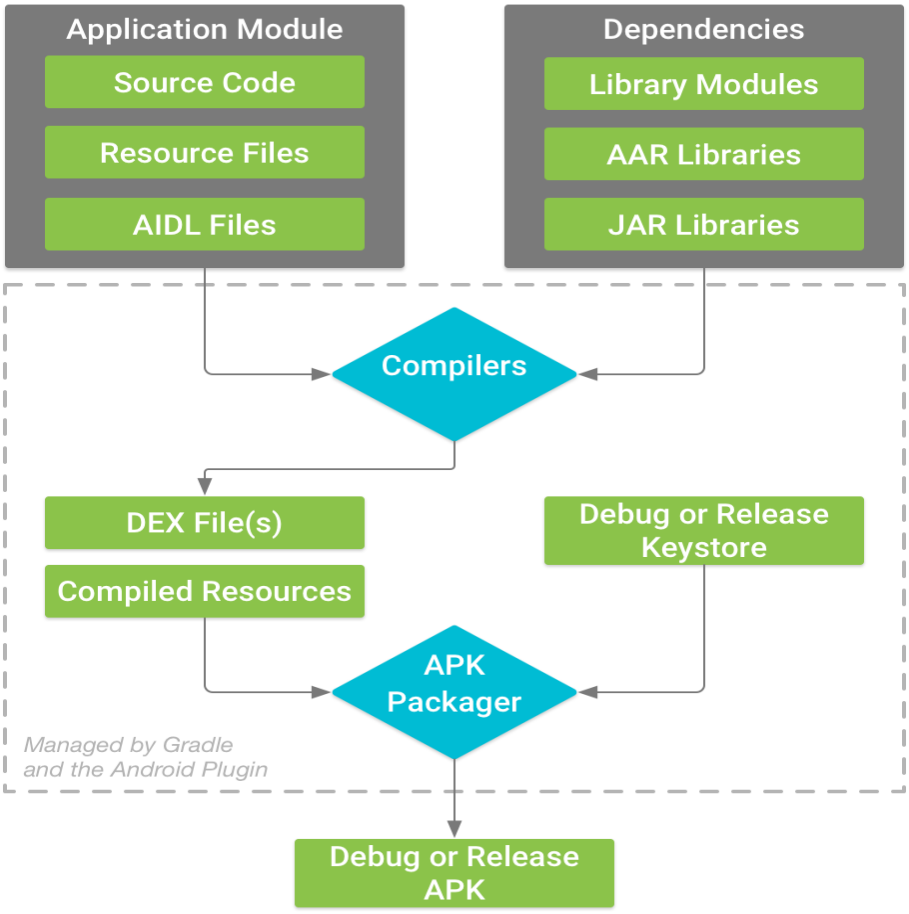
2.2.3.2 Gradle

Gradle je zodpovědný za kompilaci kódu, testování a převádí zdrojový kód na soubory DEX[1].

[1] DEX – soubor známý také jako .dex je optimalizovaný soubor třídy, který běží na virtuálním stroji Dalvik

Klasické sestavení vypadá následovně:

- Kompilátory převod zdrojového kódu do souborů DEX. Mezi sobory DEX patří byte-code, který běží na samotném zařízení se systémem Android, a vše ostatní převede do kompilovatelných zdrojů.
- Balíček APK [2] kombinuje soubory DEX a kompilované zdroje do jednoho souboru.
- Balíček APK podepíše APK.
- Před vygenerováním konečného souboru APK používá Balíček nástroj zipalign [3] k optimalizaci aplikace tak, aby jakmile poběží na zařízení Android využívala co nejméně paměti.



Obrázek 8 – Proces sestavování modulu aplikace pro Android [18]

[2]APK je formát souboru balíčku používaný operačním systémem Android.
[3] zipalign je nástroj pro zarovnání archivu zip, zajišťuje, že všechny nekomprimované soubory v archivu jsou zarovnány na začátek souboru.

Android Studio je dodáváno s předinstalovaným systémem Gradle, proto není nutné k sestavení projektu instalovat další runtime software. [18]

2.2.4 Programovací jazyky

Pro vývoj Android aplikací se používají hlavně dva jazyky Java a Kotlin, které jsou oficiálně podporovány Googlem. Aplikace pro operační systém je možné vytvářet i přes programovací jazyky, jako C++, C#, LUA nebo JavaScript a jiné.

2.2.4.1 Java

Java je objektově orientovaný programovací jazyk, který je navržen tak, aby měl co nejméně závislostí na implementaci. Již kompilovaný kód lze spustit na všech platformách, které podporují Javu. Aplikace Java se kompilují do bytového kódu, který se dá spustit na JVM.

Java byla vybrána jako hlavní jazyk pro vývoj této aplikace, a to z toho důvodu, že je stále podporovaná Googlem. Má bohatou historii projektů, kde jsou již vyřešené problémy, což usnadňuje ladění a vývoj aplikace.

2.2.4.2 Kotlin

Kotlin je staticky psaný programovací jazyk, který se zaměřuje na JVM, Android, JavaScript and Native. Kotlin je open-source a je vyvíjený společností JetBrains. Projekt byl zahájen v roce 2010 a první oficiální vydání bylo v roce 2016, je to relativně nový programovací jazyk. Kotlin je expresivní a stručný, jeho cílem je být bezpečnější pro vývojáře. V Kotlinu by například nemělo dojít k null pointer exception. Pomocí Kotlinu se dá volat kód založený na Javě a vice versa.

V roce 2019 Google oznámil na Google I/O (každoroční vývojářská konference organizovaná Googlem), že rozvoj operačního systému Android bude především prioritně v jazyku Kotlin. To neznamená, že Java nebude podporována, ale že již nemusí podporovat více platformní projekty a Jetpack Compose. Také vzorové kódy již nemusí být oficiálně dostupné v Javě. [19]

2.3 iOS

V této kapitole je krátce představena historie a současnost operačního systému iOS, představeno vývojové prostředí určené k vývoji nativních mobilních aplikací pro iOS a představeny

programovací jazyky, které aktuálně dominují vývoj nativních aplikací na tomto operačním systému.

2.3.1 Historie a současnost

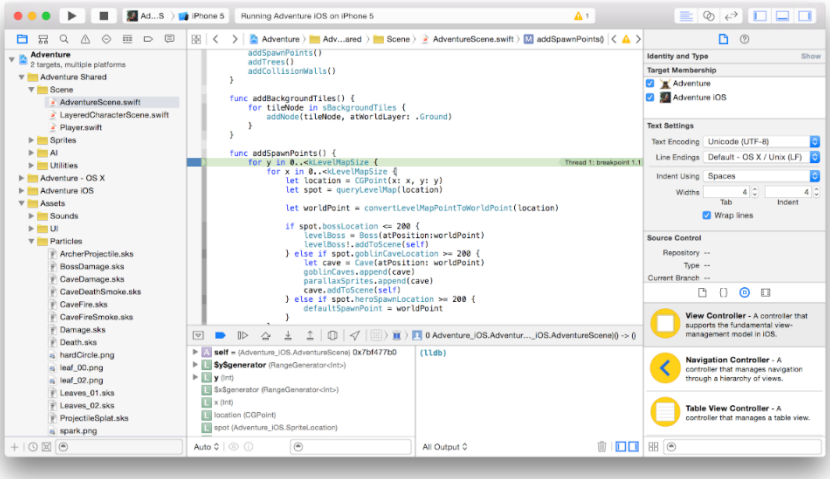
První mobilní zařízení od Apple bylo oficiálně vydáno v roce 2007. Fungovalo na operačním systému iPhone OS. Ze začátku nepodporoval iPhone OS aplikace třetích stran, takže měli uživatelé dostupných jenom pár aplikací, které byli vydány zároveň s mobilním zařízením. To se však změnilo v roce 2008, kdy byl spuštěn obchod s aplikacemi pro iPhone. [20]

iPhone OS nebyl schopen zvládnout mnoho programů, které běželi současně. To dovedlo Apple k vytvoření nového operačního systému a v roce 2010 upgraduje svůj „iPhone OS“ na „iOS“, který zvládal mít současně spuštěno velké množství programů. První iOS vyšel, jako iOS 4 a od té doby Apple konzistentně dodává jeden update ročně s novou funkcionalitou a redesignem operačního systému. Dnes je poslední oficiální vydání iOS 14. [20]

2.3.2 Xcode

Xcode je vývojové prostředí pro nativní aplikace pro iOS, společnosti Apple, v dnešní době je taky rozšířené pro vývoj aplikací pro iPad, Mac a Apple Watch. Bohužel pro instalaci a vývoj aplikací v IDE Xcode je nutné mít operační systém MacOS a Apple ID.

Xcode přirozeně podporuje všechny funkce a technologie aktuálně dostupné na iOS, pro které vytváří nativní aplikace. Xcode umožňuje tvořit grafické uživatelské rozhraní i kódování pro vytváření aplikací. Také stejně jako Android studio má vlastním emulátorem různých zařízení, které fungují na operačním systému iOS. [21]



Obrázek 9 – Náhled uživatelského rozhraní v Xcode [22]

2.3.3 Programovací jazyky

Pro nativní vývoj aplikací se v dnešní době nejvíce používá programovací jazyk Swift, který předběhl Obejctive-C v popularitě. Objective-C byl první oficiálně podporovaný jazyk pro vývoj nativních aplikací pro iOS. Aplikace se ale samozřejmě dají vyvíjet i v jiných jazycích například C#, Python, C++, HTML 5 a jiné.

2.3.3.1 Swift

Swift je hlavní programovací jazyk operačního systému iOS. Společnost Swift byla uvedena na trh společností Apple v roce 2014. V prosinci 2015 společnost Apple uvedla Swift pod licencí Apache 2.0. což znamená, že jejich kód je volně dostupný pro komerční použití, může se upravovat, distribuovat a používat k soukromému použití. Samozřejmě je Swift využíván i k vývoji pro macOS, watchOS, tvOS, Linux a z/OS. [23]

Swift je univerzální programovací jazyk vytvořený pomocí moderního přístupu k bezpečnosti, výkonu a návrhovým vzorům softwaru. Je vyvíjen jako open-source projekt a vývojáři tak mohou navrhovat změny a hlásit chyby v jeho implementaci podobně, jako například u operačního systému Linux. Swift je podporovaný, jak na macOS, tak i na Linuxu a nedávno byl dokonce vydán pro Windows 10. [23]

2.3.3.2 Obejctive-C

Je nadstavbou programovacím jazyku C a poskytuje objektově orientované schopnosti a dynamický runtime. „Objective-C dědí syntaxi, primitivní typy a příkazy řízení toku C a přidává syntaxi pro definování tříd a metod“. [24]

I když většina aplikací, které běží na iOS jsou aktuálně napsaná v Objective-C, je tento jazyk na ústupu, a to hlavně díky tomu, že Apple preferuje podporovat jejich nový jazyk Swift. I přesto se vyplatí mít znalosti tohoto jazyku. Hlavně proto, že většina dokumentací a návodů od Apple nebo i od třetích stran pro vývoj aplikací s tím počítá a nejsou mnohdy upravené pro Swift. [24]

II. PRAKTICKÁ ČÁST

3 POŽADAVKY NA APLIKACI

V této kapitole jsou popsány funkcionální a nefunkcionální požadavky pro tuto mobilní aplikaci.

3.1 Funkcionální požadavky

Funkcionální požadavky byly vybrány na základě zadání bakalářské práce, konkurence aplikací na trhu, autorových zkušenostech z mobilními aplikacemi a představou, jak bude uživatel interagovat s touto aplikací.

- Aplikace získává a zobrazuje základní informace o akciích podle jejich symbolu
- Aplikace získává historické ceny akcii a historii akcí
- Aplikace zpracovává a zobrazuje informace o akciích
- Zobrazování historické ceny akcií do sloupcových grafů
- Aplikace dokáže požádat server o předpověď ceny akcie
- Zobrazení předpovědi ceny akcie v aplikaci
- 3 rozdílné obrazovky pro možnost uživatele interagovat se získanými daty
- Vytvoření testovacího portfolia, které si uživatel může měnit podle záliby
- Umožnění nakupování a prodávání akcií (v testovacím portfoliu)
- Uchovávání dat o nákupu/prodeje akcií, vlastnictví akcií a stavu portfolia do lokálních souborů aplikace
- Výpočet zisku nebo ztráty z nákupu jednotlivých akcií
- Výpočet zisku nebo ztráty celkově ze všech uzavřených obchodů

3.2 Nefunkcionální požadavky

Nefunkcionální požadavky nespecifikují přímo funkcionalitu aplikace, ale vlastnosti a princip fungování aplikace, které uživatel nemůže přímo ovlivnit.

3.2.1 Kompatibilita

Aplikace je vyvíjena hlavně pro moderní mobilní zařízení, funkcionalita pro starší mobilní zařízení je dostupná, ale není testována.

3.2.1.1 API

Pro aplikaci bylo rozhodnuto, že bude vyvíjena pro nejnižší možná API level 21, což podle statistik (viz Obrázek 5) pokrývá 94,1% zařízení, které aktuálně využívají Android. Kde na level 21 API aplikace funguje bez problémů. Při normálním testování/ladění byla aplikace vyvíjena na API 28. A nejvyšší testovaná API je level 30, kde aplikace funguje normálně.

3.2.1.2 UI

Co se týká kompatibility Grafického rozhraní, tak nejnižší rozlišení, které bylo testováno je 480dp (480x800), kde tato aplikace je použitelná, avšak některý text například na tlačítkách může být těžce čitelný nebo nečitelný. Toto by se dalo řešit funkcionalitou, která je nabízená při vývoji, a to vytvořit speciálně layout pro menší a středně malé obrazovky. Při normálním testování a ladění byla aplikace testována na Pixel 4XL s rozlišením 1440x3040(560 dpi), Pixel 2 1080x1920 (420 dpi) a fyzickém zařízení Xiaomi Mi Mix 2 1080 x 2160 (~403 ppi), kde UI fungovalo podle očekávání. Největší grafické rozhraní bylo testováno na Android TV 1920x1080 (xhdpi), kde grafické rozhraní nemělo problém se přizpůsobit větší obrazovce.

3.2.2 Použitelnost

Pro použitelnost musí uživatel znát základní funkcionalitu mobilního zařízení a symboly akciových společností.

3.2.3 Lokalizace

Celá aplikace je lokalizovaná do anglického jazyka a ceny jsou uvedeny v dolarech. Datum je ukládán ve formátu YYYY-MM-DD HH:MM:SS.

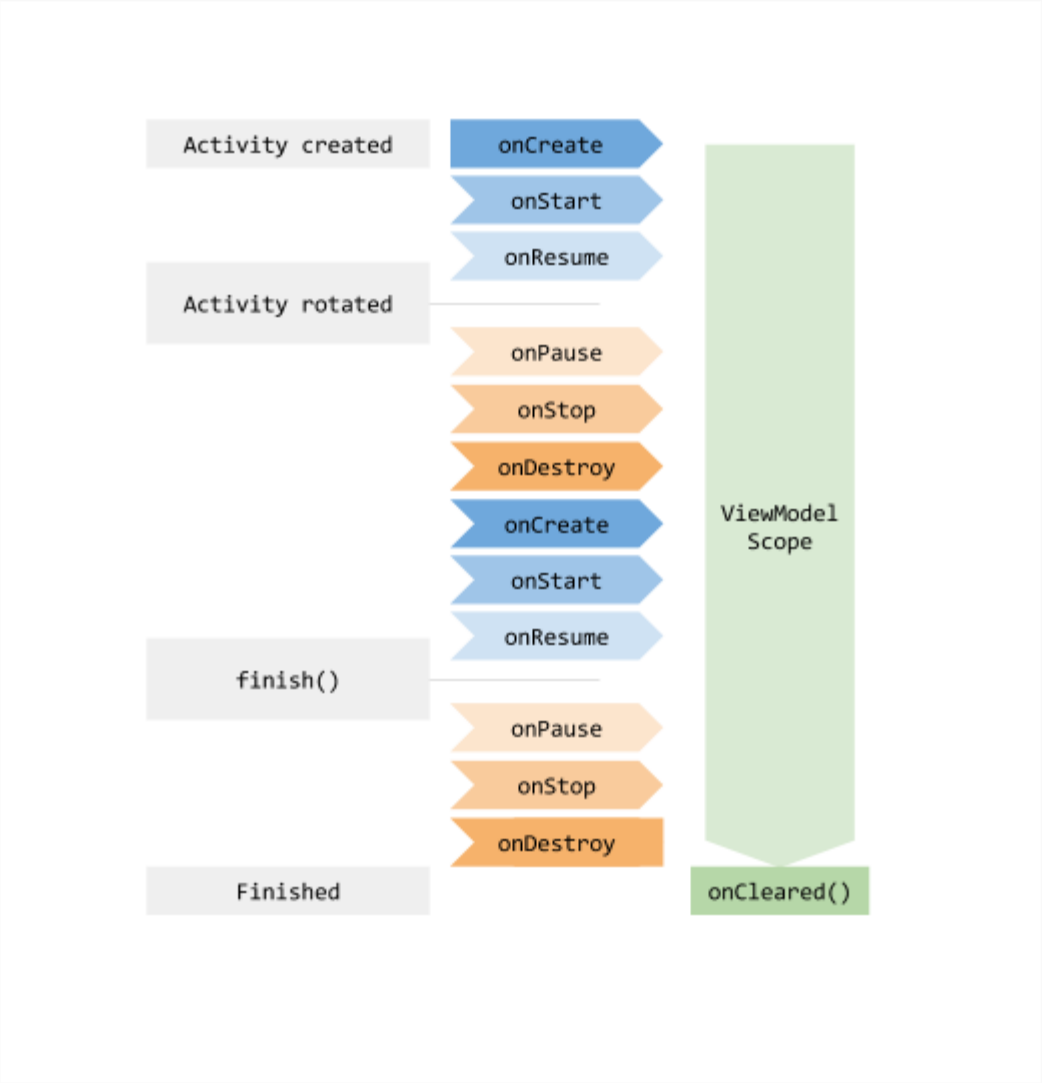
3.2.4 Dokumentace

Aplikace má popsané důležité funkce pomocí nástroje javadoc. Přístup k serveru je dokumentovaný pomocí swagger.io, kde je dokumentace automaticky vygenerovaná a je dostupná na adrese:

http://localhost:8080/edems_swag/stock_api/1.0.0/ui/ (příklad pro localhost)

4 MOBILNÍ APLIKACE

Mobilní aplikace je vyvíjena na platformě Android. Je designovaná pomocí architektury MVVM. To hlavně kvůli ViewModel třídě, která je v Androidu speciálně vytvořená pro ukládání a správu dat pro uživatelské rozhraní, [25] a to tak, že třída ViewModel umožňuje datům přežít změny konfigurace (viz Obrázek 10). Toto je kritické, protože ke změnám aktivity nebo fragmentu může dojít kdykoliv zvlášť na mobilním zařízení. Například při otočení obrazovky.



Obrázek 10 – Životní cyklus třídy ViewModel [REF 25]

4.1 Manifest

V manifestu [4]aplikace je požádáno o dvě povolení přístupu a to:

```
4   <uses-permission android:name="android.permission.INTERNET" />
```

- Povolení přístupu k internetu

Bez tohoto povolení je aplikace téměř nepoužitelná vzhledem k tomu, že všechny data o akciích se získávají ze serveru. Aplikace je použitelná, pokud již byly načteny data o akciích a poté byl internet přerušen.

```
5   <uses-permission android:name="android.permision.ACCESS_NETWORK_STATE"/>
```

- Povolení přístupu k informaci o síti

Toto povolení není nezbytné a v aplikaci je použito jenom tehdy, jako oznámení uživateli, že aktuálně jeho zařízení nemá přístup k internetu pomocí toast zprávy.

Dále je v manifestu nastavené:

```
8       <application
9           android:usesCleartextTraffic="true"
10          android:allowBackup="true"
11          android:icon="@mipmap/ic_launcher_stock"
12          android:label="@string/app_name"
13          android:roundIcon="@mipmap/ic_launcher_stock_round"
14          android:supportsRtl="true"
15          android:theme="@style/AppTheme">
16          <activity
17              android:windowSoftInputMode="adjustPan"
18              android:name=".ui.MainActivity"
19              android:label="@string/app_name"
20              android:theme="@style/splashScreenTheme">
21              <intent-filter>
22                  <action android:name="android.intent.action.MAIN" />
23                  <category android:name="android.intent.category.LAUNCHER"/>
24              </intent-filter>
25          </activity>
26          <activity
27              android:name=".ui.Detail.DetailActivity"
28              android:windowSoftInputMode="adjustPan">
29          </activity>
30      </application>
```

Řádek 9 – od API 23 a výše je nutno povolit přístup k adresám bez šifrování (http)

[4] Manifest je k nalezení v souboru *Stock_App\app\src\main\AndroidManifest.xml*

Řádek 17, 27 – nastavení chování aktivity při zobrazení virtuální klávesnice adjustPan – pouze posune obsah displeje nahoru aby se klávesnice vlezla do obrazovky. Další možnost je využít adjustResize – změní obsah stránky tak aby se do něj vlezla klávesnice.

Na ostatních řádcích jsou nastavené nezbytné věci jako nastavení popisku aplikace, stylu, ikony …

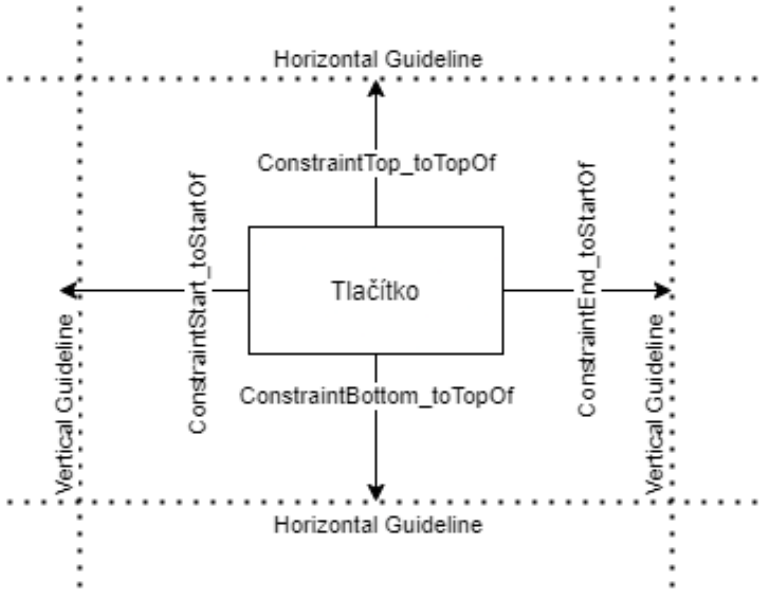
4.2 Grafické rozhraní

Hlavním cílem při vytváření grafického rozhraní bylo, aby rozhraní bylo příjemné pro používání, intuitivní a dynamické pro téměř všechna zařízení, která operují na operačním systému Android.

Grafické rozhraní pro aplikaci v operačním systému Android bylo vytvořené v Android Studiu pomocí rozložení ConstraintLayout, které umožňuje flexibilně umísťovat a měnit velikost widgetů. Widgety jsou v ConstraintLayoutu umísťovány tak, že musí být vždy upoutané na jiné widgety, které již v layout existují nebo na layout samotný.

V tomto grafickém rozhraní bylo využito nástroje guideline. Guideline je objekt, na který se mohou kotvit jiné objekty, ale neobjeví se uživateli na obrazovce. Guideline je vertikální nebo horizontální čára, která lze do layoutu vložit s pevnou pozicí nebo procentuálně vzhledem k velikosti rozložení.

Dynamické velikosti layoutu je docíleno nastavení guideline na procenta velikosti rozlišení a nastavení výšky a šířky objektu na 0dp. Tím, že je nastavený objekt na 0dp se objekt rozšíří na co největší velikost, jenž mu dovolují guidelines. Tím pádem není nutno se starat o velikost zařízení, které uživatel používá a je dynamicky rozložené. To neznamená, že layout bude čitelný například na hodinkách nebo na extrémně malých zařízeních. [26]



Obrázek 11 – Příklad designu tlačítka s ConstraintLayoutem

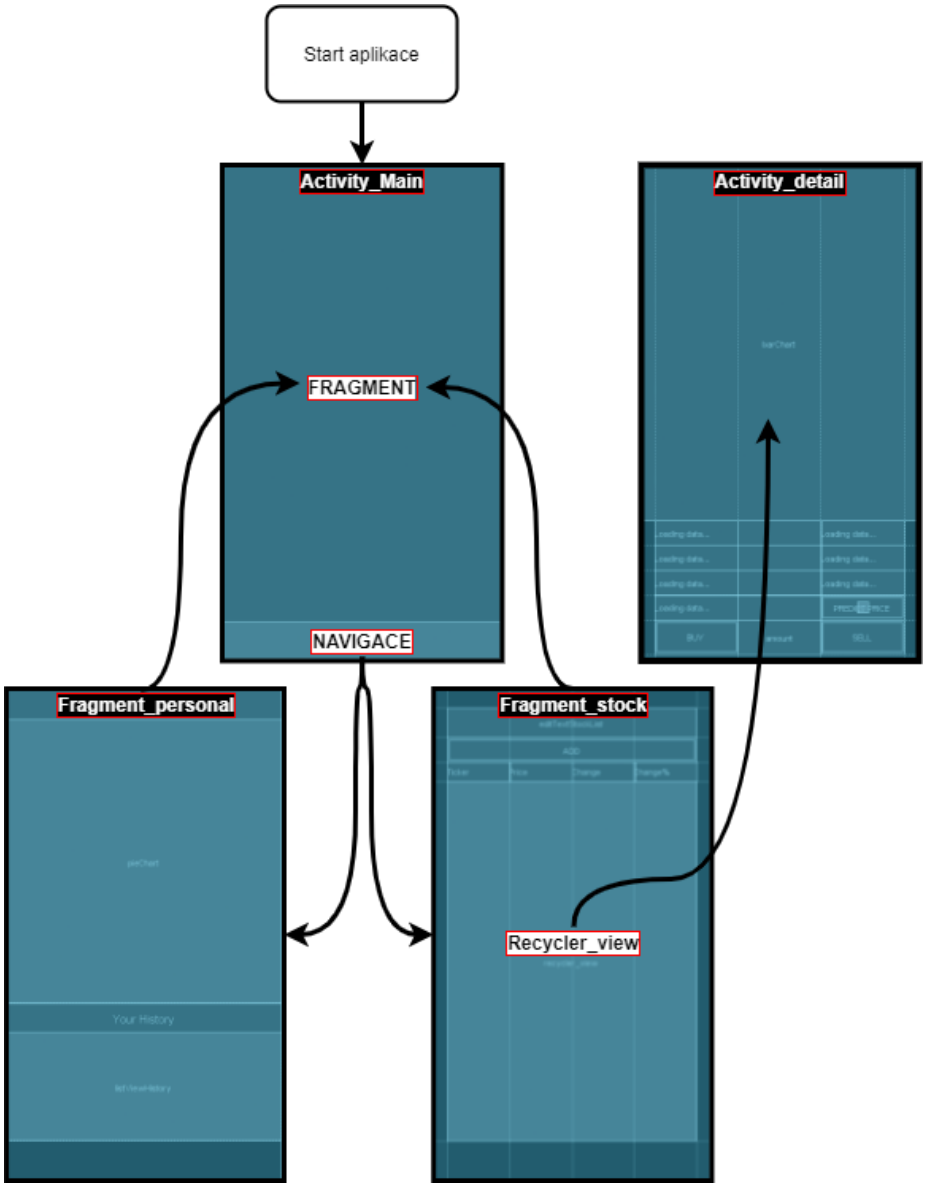
4.3 Popis obrazovek

Aplikace je rozložená do dvou aktivit MainActivity a DetailActivity. MainActivity je poté rozdělena do dvou fragmentů. Fragment_personal a Fragment_stock. V této kapitole je popsána navigace mezi jednotlivými obrazovkami, design a jejich funkcionalita.

4.3.1 Navigace mezi obrazovkami

Navigace mezi fragmenty v MainActivity je zprostředkovaná pomocí navigačního prvku navigation[5], který je umístěný na spodní časti této aktivity. Navigovat na DetailActivity je možné pouze kliknutím na akcii v recycler_view. (viz Obrázek 8)

[5] Design prvku navigation je k nalezení v *Stock_App\app\src\main\res\menu\bottom_nav_menu.xml* a kód funkčnosti navigace je k nalezení v *Stock_App\app\src\main\res\navigation\mobile_navigation.xml*



Obrázek 12 – Navigace aplikace

4.3.2 MainActivity

MainActivity[6] je první obrazovka, kterou uživatel uvidí po spuštění aplikace. Tato aktivita je v tomto projektu využita pouze jako prostředník, který v sobě drží fragmenty a navigační prvek, ve kterých je uchována logika a design.

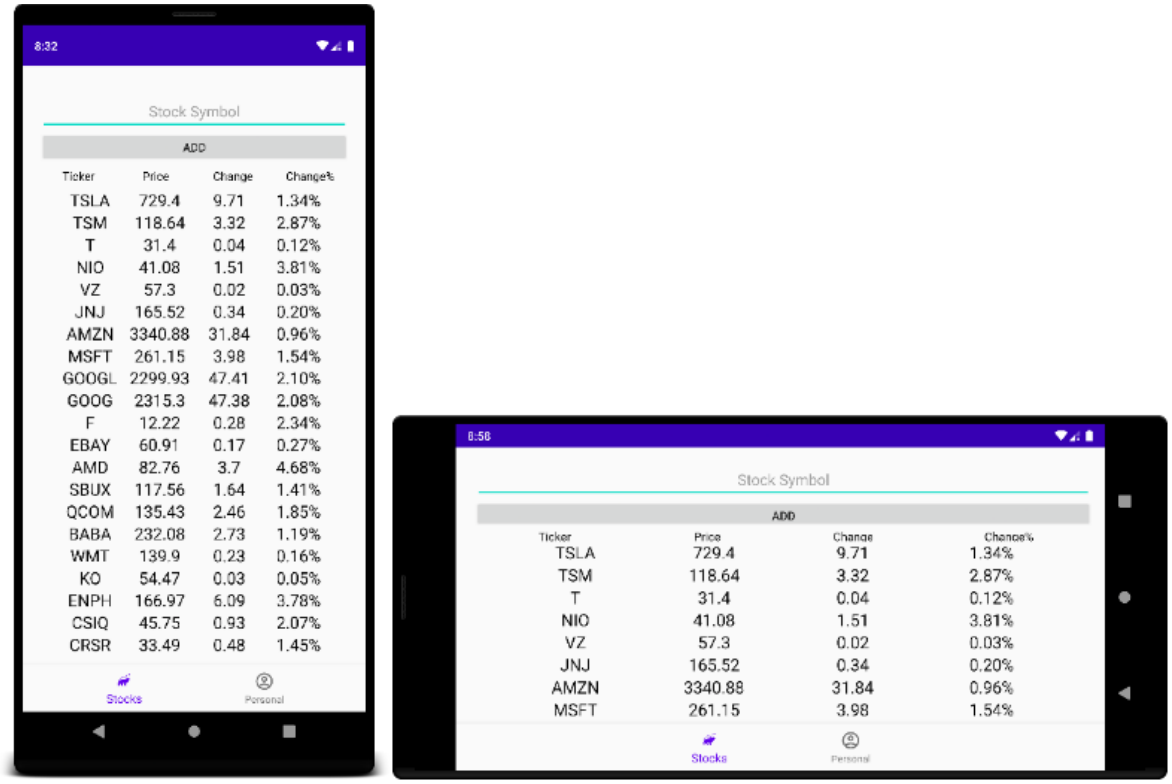
[6] MainActivity je k nalezení pod *Stock_App\app\src\main\java\cz\utb\fai\stock_app\ui\MainActivity.java*

4.3.2.1 Stock Fragment

Stock fragment je první věc, kterou uživatel uvidí, jelikož je při spuštění aplikace vykreslená do MainActivity. Uživatel zde má možnosti zadat symbol akcie do edit_text pole, pokud se symbol shoduje se symbolem existující akcie, tak po stisknutí buttonu ADD nebo stisknutí enter na virtuální klávesnici, je do recycler_view přidána akcie se základními informacemi, jehož implementace je popsána ve více detailu v podkapitole RecyclerView.

(viz Obrázek 9).

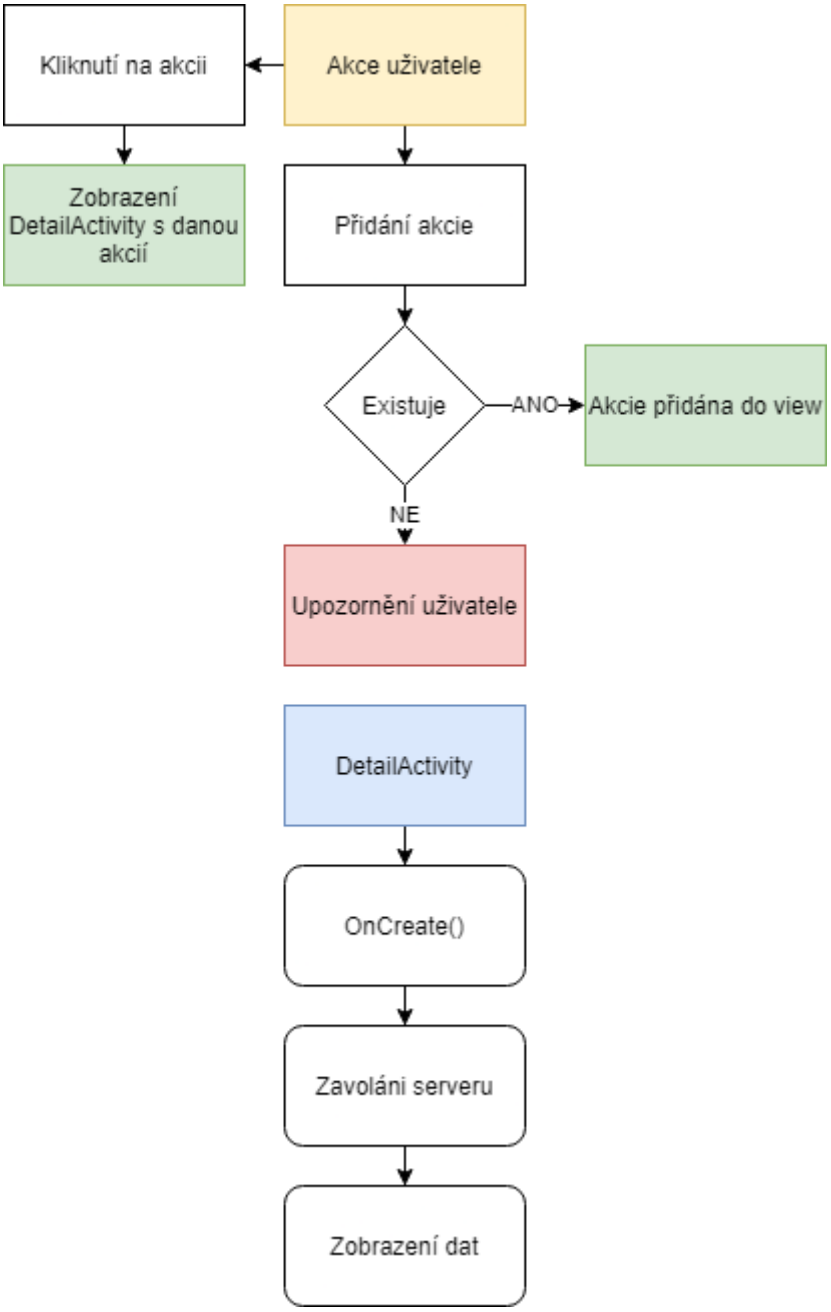
Při prvním načtení fragmentu se uživateli načtou do recycler_view data akcií, které aktuálně má v portfoliu s nejaktuálnější cenou, která je uložena na serveru



Obrázek 13 – MainActivity (Stock Fragment) s načtenými daty

Pro schéma stock fragmentu jsou vytvořeny dva layouty. Jeden pro umístění na výšku a jeden na šířku. [7]

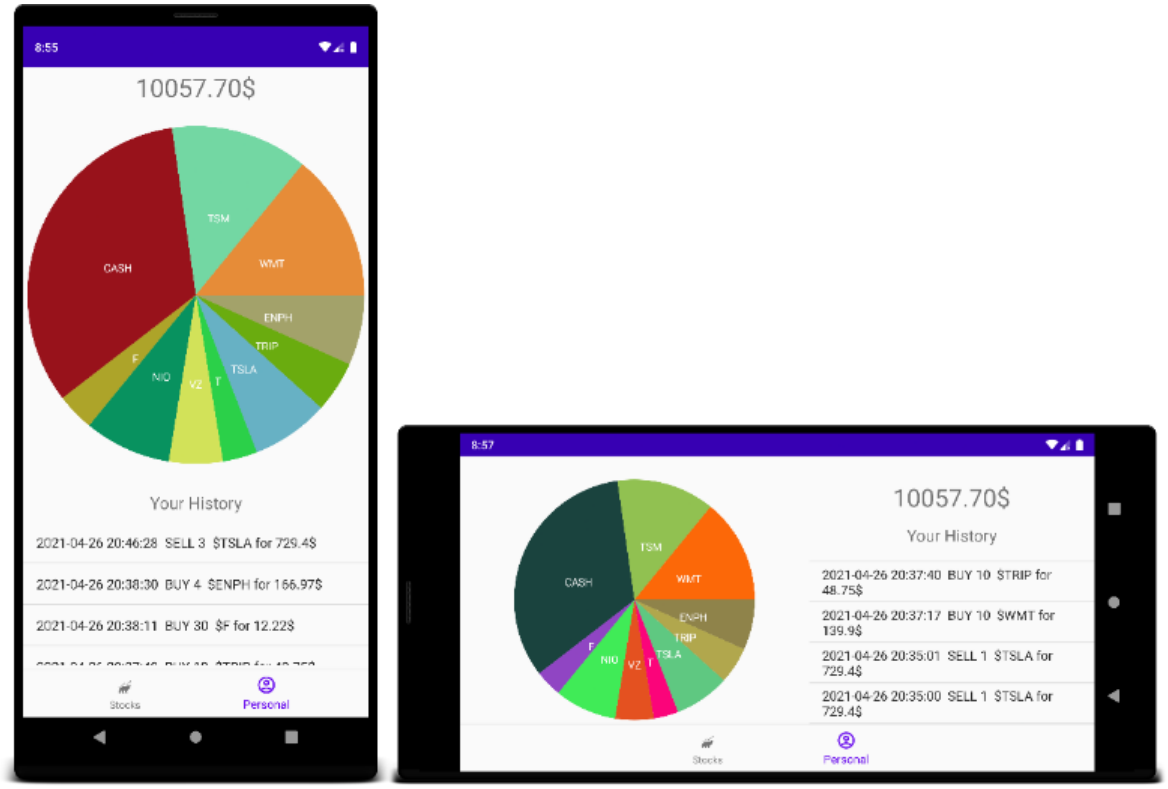
[7] layout je možné najít pod názvem: *fragment_stock.xml, fragment_stock.xml(land)*



Obrázek 14 - Diagram funkcionality fragmentu a uživatelské interakce ve stock fragmentu.

4.3.2.2 Personal Fragment

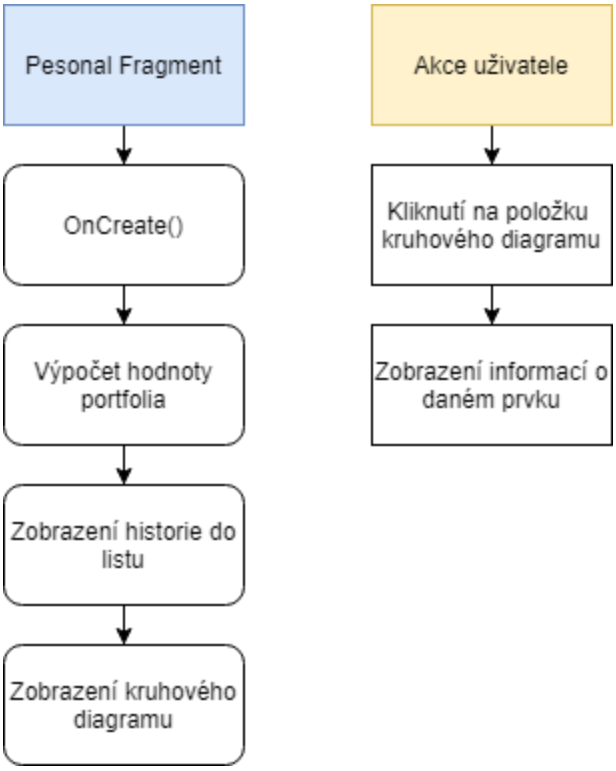
Personal fragment slouží jako osobní stránka uživatele, kde může vidět historii všech svých nákupů a prodejů akcií v listu. Je zde také vyobrazena cena portfolia číselně a v kruhovém diagramu, který je implementovaný z knihovny MPAndroidChart [27]. Po kliknutí na diagramu se zobrazí daná akcie s detaily o vlastnictví. Pomocí vyskakovacího okna implementovaného z knihovny android-simple-tooltip. [28]



Obrázek 15 – MainActivity (Personal Fragment) s načtenými daty

Pro Personal Fragment jsou vytvořené dva layouty jeden pro umístění na výšku a jeden na šířku.[8]

[8] layouty je možné najít pod názvem: *fragment_personal.xml, fragment_personal.xml(land)*



Obrázek 16 – Diagram funkcionality a uživatelské interakce v personal fragmentu.

Při vytvoření fragmentu se vypočte hodnota portfolia[9], která se zobrazuje v textovém poli v horní části obrazovky, dále je ze souboru načtena historie nákupů a prodejů[10], která je načtena do seznamu a zobrazena, a nakonec jsou načteny data do kruhového diagramu[11].

Uživatel na této stránce má možnost interagovat právě s kruhovým diagramem, a to tak, že se může dotknout části diagramu, kde se mu po vyobrazí tooltip s informacemi o dané akcii viz Obrázek 16. Zobrazuje se zde symbol akcie, počet akcií v portfoliu, průměrná hodnota nákupu akcie, aktuální cena akcie, procentuální rozdíl od stávající ceny, dosavadní výdělek z akcie, a nakonec celková hodnota všech akcií tohoto symbolu v portfoliu.[12]

---

[9]Funkce pro výpočet a zobrazení portfolia je k nalezení v Stock_App\app\src\main\java\cz\utb\fai\stock_app\ui\Personal\PersonalFragment.java na řádku 67

[10] Funkce pro zobrazení tohoto listu je k naleznutí v Stock_App\app\src\main\java\cz\utb\fai\stock_app\ui\Personal\PersonalFragment.java na řádku 85

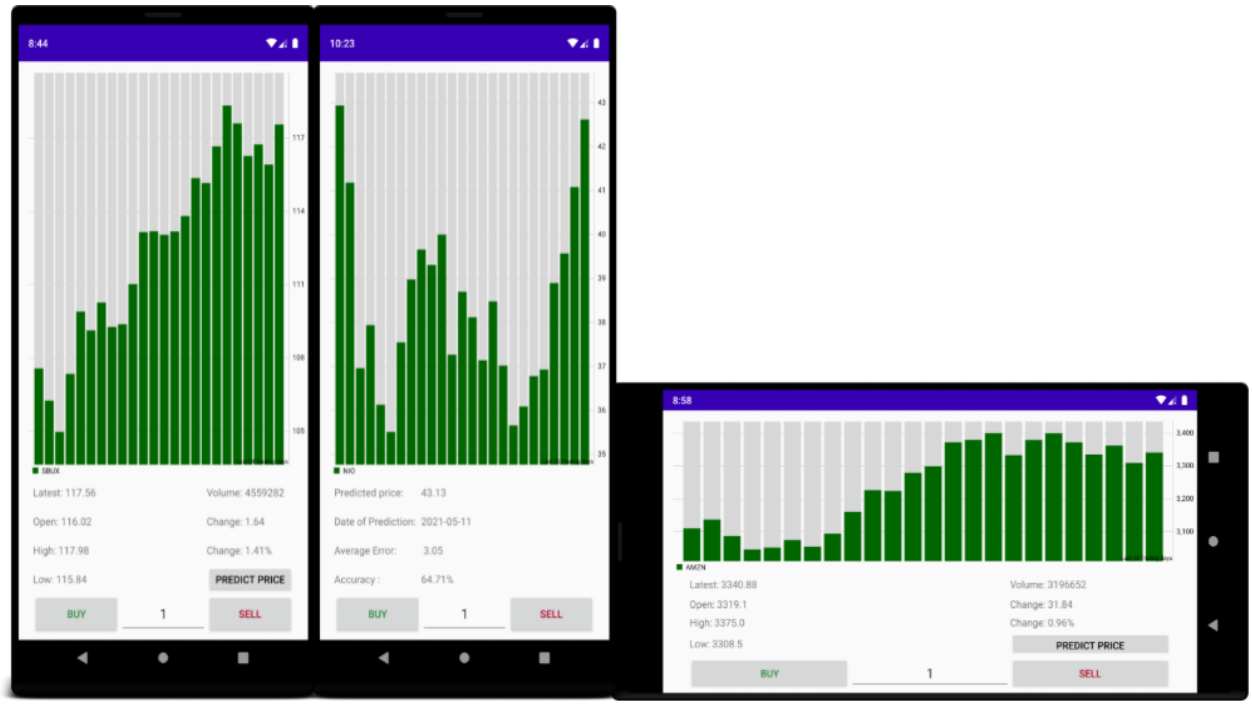
[11] Funkce pro a zobrazení kruhového diagramu je k nalezení v Stock_App\app\src\main\java\cz\utb\fai\stock_app\ui\Personal\PersonalFragment.java na řádku 137

[12] Funkce s výpočtem a zobrazením vyskakovacího okna je k nalezení v Stock_App\app\src\main\java\cz\utb\fai\stock_app\ui\Personal\PersonalFragment.java na řádku 100

Obrázek 17 – Příklad obsahu vyskakovacího okna pro akcii Tesla

4.3.3 DetailActivity

Tato aktivita se zobrazí uživateli poté, co klikne na vybranou akcii z recycler_view. V této aktivitě má uživatel možnost si nakoupit nebo prodat akcii a požádat o předpověď akcie, pokud je předpověď akcie dostupná budou mu vypsány data predikce (viz Obrázek 11 druhá obrazovka), jinak se na serveru začne předpovídat zadaná akcie. Graf v této aktivitě je implementovaný z knihovny MPAndoridChart [27].



Obrázek 18 – DetailActivity s načtenými daty

Pro aktivitu jsou vytvořené dva layouty pro umístění na výšku a šířku.[13]

[13] Layouty je možné najít pod názvem: *activity_detail_stock.xml, activity_detail_stock.xml (land)*

Obrázek 19 - Diagram funkcionality a uživatelské interakce v aktivitě DetailActivity

Při vytvoření aktivity je zavolán server, odkud je načtená historie akcie, která je poté zobrazena do grafu. Souběžně jsou taky nastavovány hodnoty pro textView. V této aktivitě má uživatel možnost interagovat s edit textem, do kterého se píše celé číslo kolik chce nakoupit[14]

[14] Funkce pro zakoupení akcie lze najít v Stock_App\app\src\main\java\cz\utb\fai\stock_app\ui\Detail\ DetailActivity.java na řádku 62

nebo prodat[15] akcií. Také je zde možné zavolat server o předpověď ceny akcie. Tyto interakce jsou popsány v Obrázku 19.

4.4 Komunikace se serverem

Komunikace se serverem je zprostředkována přes HTTP Google knihovnou Volley. Díky tomu, že API server je při vývoji používán jako localhost a není nikde hostovaný. Je defaultně IP adresa nastavená na 10.0.2.2, což je speciální alias v Android Studiu pro localhost. Pokud by aplikace byla spuštěná nebo testována v jiném prostředí než na emulátoru, například na mobilním zařízení, které je na stejné Wi-Fi je potřeba změnit IP adresu, a to na IP adresu, na které aktuálně běží server.[16]

4.5 RecyclerView

RecyclerView je takzvaný ViewGroup (seskupení náhledů). ViewGroup je speciální view (náhled), který může v sobě držet další views (náhledy). RecyclerView je v projektu využitý, jako dynamický seznam, do kterého se přidávají data o akciích a funguje jako propojení mezi DetailActivity a MainActivity. Design RecyclerView[17] je také dynamický, aby se přizpůsobil velikosti obrazovky stejně jako grafické rozhraní.

4.5.1 LiveData

LiveData je třída, která je využívaná pro držení specifických dat. LiveData lze v daném životním cyklu sledovat. To znamená, že pozorovatele (Observer) lze přidat zároveň s LifecycleOwner, a tento pozorovatel bude upozorněn na úpravy zabalených dat, pouze pokud je spárovaný LifecycleOwner v aktivním stavu[18] [29]

4.5.1.1 Observer

Observer neboli pozorovatel je interface, který je možné implementovat na třídu LiveData. Pokud se třída, na kterou je implementován, jakkoliv změní, observer je informován a na

[15] Funkce pro prodání akcie lze najít v Stock_App\app\src\main\java\cz\utb\fai\stock_app\ui\Detail\ DetailActivity.java na řádku 66

[16] Změnit IP adresu je umožněné v souboru *strings.xml*, kde je potřeba přepsat proměnou *IP_ADDRESS*.

[17] Design RecyclerView je možné najít pod jménem *layout_listitems.xml*

[18] LifecycleOwner je považován za aktivní, pokud je jeho stav spuštěn nebo obnoven.

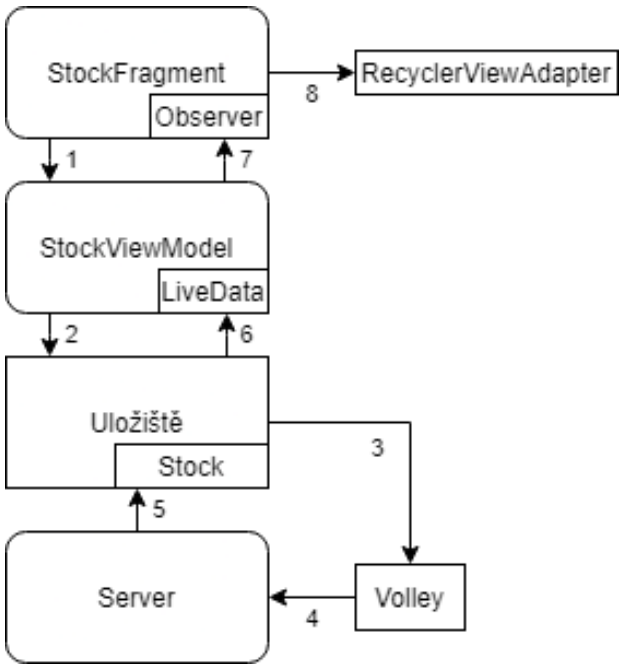
základě toho můžeme v programu na to reagovat. Díky tomu, že je pozorována třída LiveData, tak není nutno se starat o to, v jaké fázi je aktivita/fragment.

4.5.2 RecyclerViewAdapter

Adaptér poskytuje vazbu mezi specifickými daty aplikace a daty které jsou zobrazené ve view.

4.5.3 Princip zobrazování akcií

V následujícím Obrázku je představen princip, jak se data o akciích dostávají do RecyclerView



Obrázek 20 – Princip zobrazování akcií v RecyclerView

1. Ve fragmentu StockFragmentu je inicializován RecyclerView – nastaven adaptér pro RecyclerView a LayoutManager . Poté je ve fragmentu implementován Observer, který je nastaven na pozorování třídy MutableLiveData <List<Stock>>. Dále je implementovaná třída StockViewModel.
2. Ve ViewModelu StockViewModel je deklarovaná třída MutableLiveData<List<Stock>> a také implementována třída úložiště StockRepository.
3. Třída StockRepository (Uložiště pro akcie), jak už název napovídá, slouží jako úložiště pro všechny aktuálně načtené akcie ze serveru. Třída je navrhnutá podle návrhového vzoru Singleton. To z toho důvodu, že třída StockRepository je využívaná i při kalkulaci

ceny portfolia, proto je výhodné mít jenom jednu instanci této třídy v projektu. Dále se volá server pomocí knihovny Volley.

4. Knihovna Volley asynchronně volá server.
5. Server vrací požadovaná data s informacemi o akcii a data jsou vložena do třídy StockRepository
6. Data jsou vráceny zpět do StockViewModel, kde jsou uloženy do MutableLiveData třídy.
7. Vzhledem k tomu, že se data ve třídě MutableLiveData změnily, tak reaguje Observer, který v sobě má jedinou akci a to, že volá adaptér.
8. RecyclerViewAdapter oznamuje RecyclerView, že se data změnily – RecyclerView je zobrazen s novými daty.

4.6 Data aplikace

Data aplikace jsou ukládány do interního úložiště mobilního zařízení, a to z mnoha důvodů. Za prvé jsou tyto data zabezpečené více, než kdyby byly k přístupu v externím uložišti, kde k nim má přístup každá jiná aplikace. Také ne vždy každé zařízení má externí úložiště, ale hlavní důvod je, že není nutno žádat uživatele o povolení k zapisování do interního úložiště.

Některá data se neukládají a jsou v aplikaci jenom při běhu aplikace a jakmile je aplikace uzavřena, data nejsou nikde uloženy (například nejnovější údaje o akciích). Některá data jsou v aplikaci jenom po dobu života dané aktivity, například předpověď k dané akcie.

Při startu aplikace se prvně zkontroluje, zda existují adresář a soubory do kterých se zapisují lokální data. Pokud neexistují jsou vytvořeny nové prázdné soubory, kromě souboru money, do kterého je připsáno 10 000 $ (při startu aplikace je portfolio založené z 10 000$).

K udržování stejného formátu je využito Google knihovny Gson, která převádí modelovou třídu na JSON formát.

4.6.1.1 Money

Money je soubor, do kterého se zapisuje aktuální hodnota peněz v portfoliu.

Příklad obsahu souboru: {"amount":3179.6,"currency":"$"}

- amount [Double] – počet peněz v portfoliu

- currency [String] – symbol pro právě používanou měny

4.6.1.2 History

History je soubor, do kterého se ukládají záznamy o všech úspěšně prodaný a nakoupených akciích. Nákupy a prodeje jsou postupně oddělené pomocí čárky.

Příklad dvou transakcí zápasných v souboru:

{"amount":"1","date":"2021-04-13
22:49:00","name":"NIO","operation":"BUY","price":"37.14"},
{"amount":"1","date":"2021-04-26
20:34:58","name":"TSLA","operation":"SELL","price":"729.4"}

- amount [String] – počet nakoupených/prodaných akcií
- date [String] – datum nakoupení
- name [String] – název symbolu akcie
- operation [Trade[19]] – zda jde o nákup nebo prodej
- price [String] – cena akcie

4.6.1.3 Portfolio

Portfolio je soubor, do kterého se ukládá aktuální počet akcií v portfoliu.

Příklad akcie v souboru:

{"amount":10,"averagePrice":139.9,"ticker":"WMT"}

- amount [Integer] – počet vlastněných akcií
- avaragePrice [Double] – zprůměrovaná cena nakoupených akcií
- ticker [String] – symbol akcie

[19] Trade – enum z konstanty [BUY, SELL]

5 API SERVER

API server byl vytvořen ze dvou hlavních důvodů. První, aby bylo ušetřeno baterii a zdroje u mobilního zařízení, jelikož předpověď je operace, která zabírá čas a zdroje. Druhý představuje ušetření Api volání, jelikož data o akciích jsou přebírány z Alpha Vantage [30], kde verze API klíče, která je využívaná v tomto projektu je omezená na 5 API volání za minutu a 500 volání za den.

5.1 Swagger

Swagger je IDL pro popis rozhraní RESTful API pomocí formátu JSON. Swagger se používá společně se sadou open source nástrojů pro navrhování, vytváření, dokumentování a používání webových služeb RESTful.

Server byl implementován v Python Flask a byl vygenerován pomocí Swagger verze 3.0.0 [31]. Server je vygenerovaný pomocí YAML souboru[20]. YAML soubor v sobě obsahuje tři GET požadavky, které jsou popsány v kapitole HTTP GET požadavky. Rovnou při generování serveru je vytvořená i dokumentace pro API server podle YAML souboru. Server je po vygenerování plně funkční a je do něj potřeba jenom doplnit business logika pro GET požadavky a inicializaci.

5.2 Spuštění serveru

Pro korektní spuštění serveru je nutné mít nainstalované požadované knihovny od Swagger.io + knihovny, které jsou uvedeny níže v kapitole Použité knihovny.

Je možné jít do kořenové složky serveru a pomocí pipu nainstalovat požadavky, které jsou uvedené ve složce requirements.txt.

pip3 install -r requirements.txt

Nastartování serveru je poté taky inicializováno z kořenové složky, a to pomocí příkazu:

python3 -m swagger_server na Linuxu *nebo python -m swagger_server* na Windows.

[20] Soubor je k nalezení pod názvem *swagger.yaml*

5.3 Databáze

Databáze je vytvořena pro uchování již zpracovaných dat a ušetření API volání na Alpha Vatntage server [30]. Databáze je vytvořena pomocí SQLite, která je implementovaná v Pythonu pomocí modulu sqlite3. Tento modul se nemusí instalovat samostatně, jelikož je standartně dodáván s verzí Python 2.5 a výše. A server byl vyvíjen a testován na Python verzi Pythonu 3.8.9

Databáze je ukládána lokálně do kořenové složky[21]. Při spuštění serveru je prvně zkontrolováno[22], zda databáze existuje a pokud ne, je následovně vytvořena s níže zobrazenými tabulkami.

STOCK		
SYMBOL	**VARCHAR (10) NOT NULL**	**PK**
VALID	INTEGER NOT NULL	
JSON	TEXT NOT NULL	

STOCK_HISTORY		
SYMBOL	**VARCHAR (10) NOT NULL**	**PK**
LAST_CALLED	DATETIME NOT NULL	
JSON	TEXT NOT NULL	

PREDICTION		
SYMBOL	**VARCHAR (10) NOT NULL**	**PK**
STATE	TEXT NOT NULL	
PRICE	REAL	
MEAN_ABSOLUTE_ERROR	REAL	
ACCURACY	REAL	
PRICE_DATE	DATETIME	

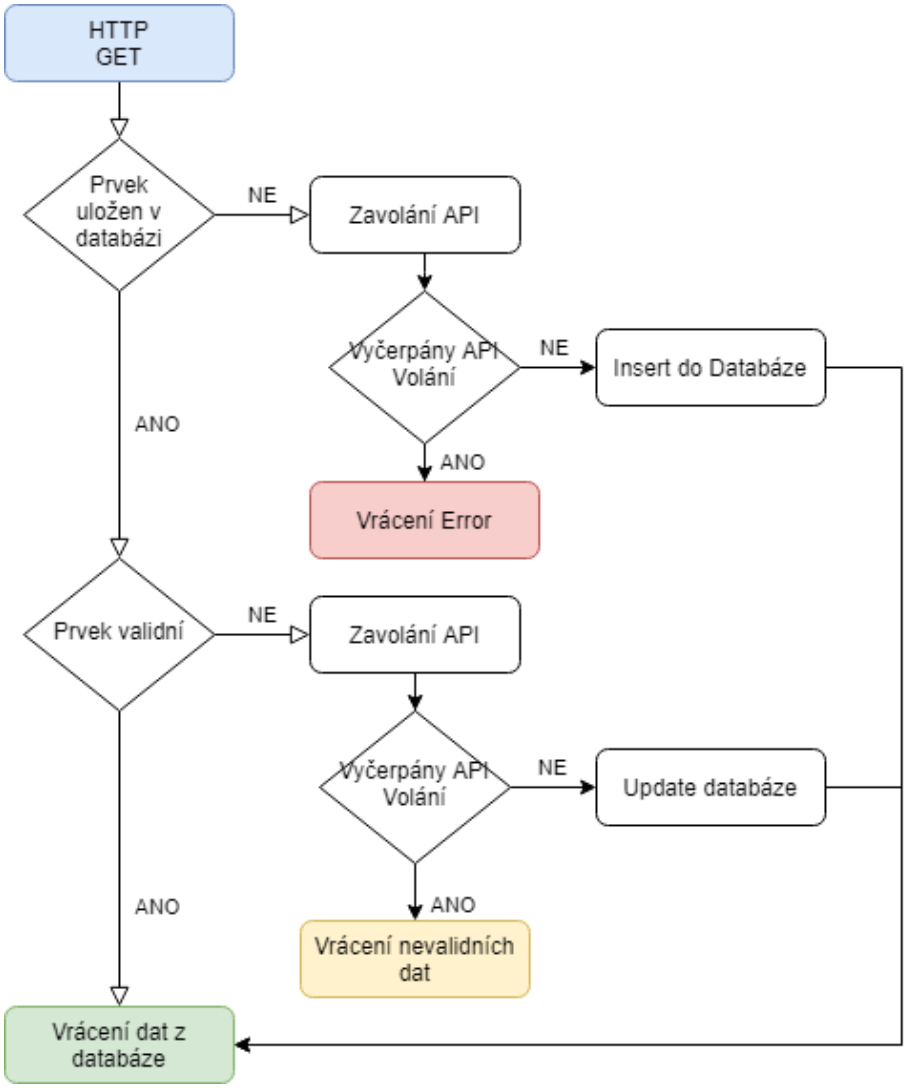
Obrázek 21 – Tabulky Databáze

5.4 HTTP GET požadavky

V API serveru jsou v provozu celkově tři GET požadavky. Požadavek /quote a /history se v logice, jak vrací data, diferencují pouze délkou jejich validity. Logika těchto dvou požadavků je popsána níže v Obrázku 18. Logika požadavku /predict je popsán v Obrázku 19.

[21] Databáze je uložena pod jménem *stock.db*

[22] Soubor s funkcemi vytvoření databáze je k nalezení: *swagger_server/controllers/__init__.py*



Obrázek 22 – Logika serveru pro požadavky GET /quote a GET /history

5.4.1 Požadavek GET /quote

GET /quote je HTTP požadavek, který vrací základní informace o akcii.

Volání quote:

http://IP_ADRESS/edems_swag/stock_api/1.0.0/quote?ticker= TICKER

- IP_ADRESS – IP adresa serveru.
- TICKER [string] – povinný parametr, do kterého se zadává symbol požadované akcie.

Quote je poté vrácena v tomto JSON formátu:

```
{
"Global Quote": {
    "01. symbol": "AMZN",
```

```
    "02. open": "3356.1900",
    "03. high": "3367.9750",
    "04. low": "3272.1300",
    "05. price": "3311.8700",
    "06. volume": "5439449",
    "07. latest trading day": "2021-05-04",
    "08. previous close": "3386.4900",
    "09. change": "-74.6200",
    "10. change percent": "-2.2035%"
  }
}
```


Pokud quote není uložená v databázi, nebo je uložená v databázi, ale data nejsou validní, tak se vkládají data do tabulky STOCK. Pří vkládání a updatu se vkládají data do tří sloupců:

- SYMBOL – ticker/symbol akcie, který také slouží jako primární klič
- JSON – obsahuje všechny data získaná s Alpha Vantage tento sloupec se také vrací při korektní validitě.
- VALID – do něj je zapisován aktuální UNIX čas + validní čas.[23]
  Validita je poté testována tak, že pokud je aktuální UNIX čas menší nebo rovný sloupci VALID je akcie ještě validní a nemusí se volat API pro obnovení dat.

5.4.2 Požadavek GET /history

GET /history je HTTP požadavek, který vrací data o akcii 100 dnů zpět.

Volání historie:

http://IP_ADRESS/edems_swag/stock_api/1.0.0/history?ticker= TICKER

- IP_ADRESS – IP adresa serveru.
- TICKER [string] – povinný parametr, do kterého se zadává symbol požadované akcie.

Historie je vypisována ve formátu:


```
{
  "Meta Data": {
    "1. Information": "Daily Time Series with Splits and Dividend
Events",
    "2. Symbol": "AMZN",
    "3. Last Refreshed": "2021-05-04",
    "4. Output Size": "Compact",
    "5. Time Zone": "US/Eastern"
  },
  "Time Series (Daily)": {
```


[23] Validní čas je nastaven na hodnotu 3600, validitu je možné změnit v *stock_ape_controller.py* proměnná má název time_valid

```
    "2020-12-09": {
      "1. open": "3167.89",
      "2. high": "3174.43",
      "3. low": "3088.0",
      "4. close": "3104.2",
      "5. adjusted close": "3104.2",
      "6. volume": "4100836",
      "7. dividend amount": "0.0000",
      "8. split coefficient": "1.0"
    },...
A následovně dalších 99 dnů zpět.
```

Pokud historie není uložená v databázi nebo je uložená v databázi, ale data nejsou validní tak se vkládají data do tabulky STOCK_HISTORY. Pří vkládání a obnovení tabulky se vkládají data do tří sloupců:

- SYMBOL – ticker/symbol akcie, která se volá, také slouží jako primární klič.
- JSON – obsahuje všechny data o historii získané s Alpha Vantage, tento sloupec se také vrací při korektní validitě.
- LAST_CALLED – obsahuje aktuální datum ve formátu YYYY-MM-DD
  Validita historie je jeden den. Pokud nebyl požadavek volán ve stejném dni musí být aktualizován.

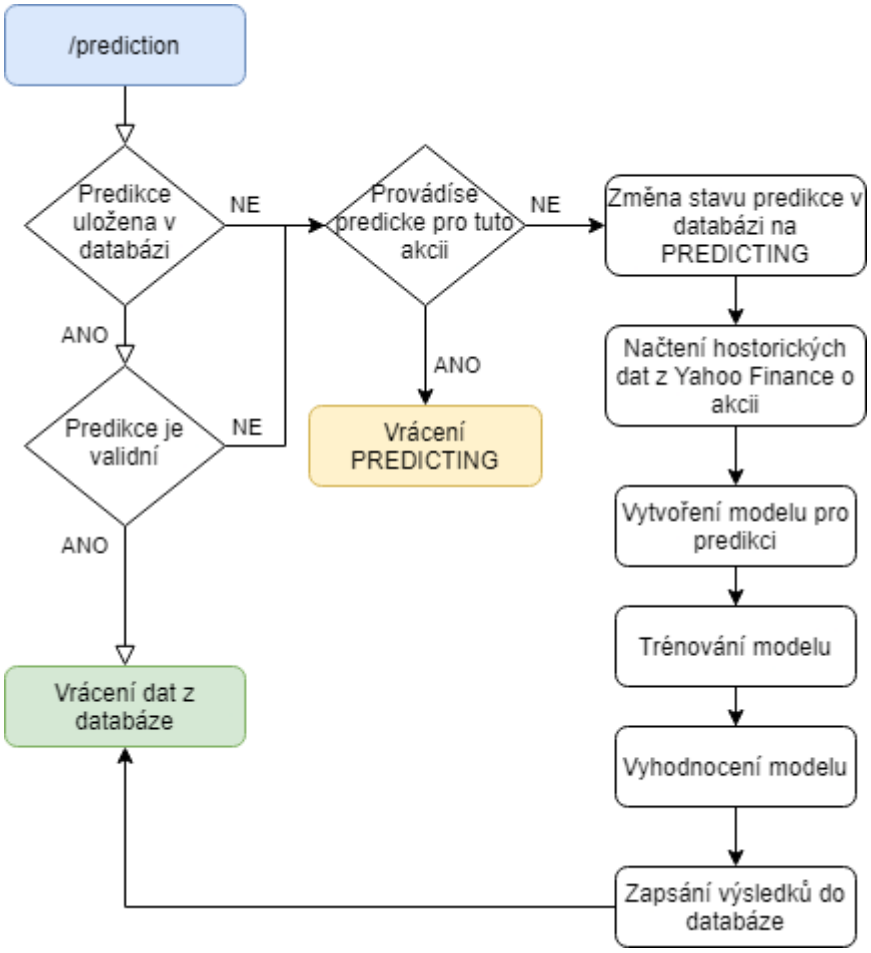
5.4.3 Požadavek GET /prediction

GET /prediction je HTTP požadavek, který vrací predikci akcie. Logika tohoto požadavku je popsána v Obrázku 19. Při příchodu požadavku je nejdřív nahlédnuto do databáze, pokud je predikce, již v databázi a je platná, je vrácena. Pokud není v databázi nebo není platná, je nastaven stav do databáze na PREDICTING a je započata predikce dané akcie.

Volání prediction:

http://IP_ADRESS/edems_swag/stock_api/1.0.0/prediction?ticker=TICKER

- IP_ADRESS – IP adresa serveru.
- TICKER [string] – povinný parametr, do kterého se zadává symbol požadované akcie.



Obrázek 23 - Logika serveru pro požadavek GET /prediction

Při zapisování do databáze se ukládají následující řádky do tabulky PREDICTION:

- SYMBOL [Varchar (10)] - ticker/symbol akcie, která se předpovídá, také slouží jako primární klič.
- PRICE [Real] – cena předpovědi akcie.
- MEAN_ABSOLUTE_ERROR [Real] – průměrná absolutní chyba.
- ACCURACY [Real] – přesnost předpovědi.
- PRICE_DATE [DateTime] – datum do kdy je predikce validní.

5.5 Predikce

Predikce je na serveru implementovaná z open source projektu [32], který byl vyvíjen pod MIT licencí. Projekt je vyvíjen v programovacím jazyku Python, takže implementace do serveru neuvedla žádný problém. Algoritmus je postaven na neuronové síti, která je

implementovaná pomocí knihovny TenserFlow. Pomocí této knihovny je sestavena rekurentní neuronová síť, pomocí buněk LSTM.

5.5.1 Použité knihovny

Pro korektní funkcionalitu je potřeba mít k dispozici následující knihovny

5.5.1.1 TensorFlow

TensorFlow byl vytvořen týmem Google Brain, je to open source knihovna pro numerické výpočty a strojové učení. TenserFlow spojuje dohromady modely strojového učení, neuronové sítě a algoritmy. TenserFlow používá Python pro front-end a pro back-end je kód spuštění ve vysoce výkonném C++.

5.5.1.2 Pandas

Poskytuje vysoce výkonné a snadno použitelné datové struktury a nástroje pro analýzu dat pro programovací jazyk Python.

5.5.1.3 NumPy

NumPy je knihovna pro programovací jazyk Python, která přidává podporu pro velká vícerozměrná pole a matice.

5.5.1.4 Yahoo_fin

Yahoo_fin je balíček Pythonu 3, který slouží k získávání historických údajů o cenách akcií také poskytuje aktuální informace o tržních stropech, dividendových výnosech.

V úvahu by mohlo přijít i využít Alpha Vantage API, ale vzhledem k výše nastíněnému problému s nedostatkem volání byla zachována tato knihovna.

5.5.1.5 Sklearn (Scikit-learn)

Scikit-learn je knihovna s otevřeným zdrojovým kódem pro strojové učení, která podporuje učení pod dohledem a bez dohledu. Poskytuje také různé nástroje pro přizpůsobení modelu, předzpracování dat, výběr a vyhodnocení modelu a mnoho dalších nástrojů.

5.5.2 Průběh predikce

Průběh predikce je již lehce nastíněn v Obrázku 19, ale v této kapitole je rozebrán do většího detailu.

1. Je nastaven náhodná hodnota seedu na statickou hodnotu, aby při trénování a testování zůstávaly stabilní výsledky.
2. Pomocí knihovny Yahoo_fin jsou stažené historické hodnoty akcie dostupné na Yahoo Finance, a to tyto následující hodnoty.
   Open – cena akcie při otevření trhu
   High – největší cena akcie v jednom dnu
   Low – nejnižší cena akcie v jednom dnu
   Volume – Počet nakoupených a prodaných akciích
   Adjusted close – cena akcie při zavření trhu + je tato cena upravena, pokud u akcie došlo k rozdělení.

   Dále jsou tyto data uspořádány a seřazeny podle parametrů zadaných do funkce[24]:

- Proměnná **n_steps** – kroky neboli délka okna označuje historickou délku sekvence, kterou chceme použít. Nastavena je proto, je používána opakující se neurální síť, a do této sítě musíme vkládat data, v tomto projektu je určena hodnota na 50. To znamená, že bude použito vždy 50 dní pro předpověď dalšího časového kroku.
- Proměnná **scale [**boolean**]** – udává, zda se mají škálovat ceny od 0 do 1. V tomto případě je nastavena hodnot na True, protože škálování vysokých hodnot od 0 do 1 pomůže neurální síti učit se mnohem rychleji a efektivněji.
- Proměnná **lookup_step** – udává budoucí vyhledávací krok k předpovědi, V tomto projektu na nastavena na 14 dnů. To znamená, že budoucí další vyhledávací krok bude za dva týdny.
- Proměnná **split_by_date [**boolean**]** - označuje, zda rozdělit tréninkové a testovací sady podle data.

[24] Funkce je k naleznutí v souboru *swagger_server/controllers/stock_ape_controller.py* na řádku 354.

False – znamená, že náhodně rozdělíme data na trénink a testování pomocí funkce knihovny sklearn train_test_split ().
True – rozdělení data podle chronologického pořadí.
V projektu je nastavena hodnota na True.

3. Dále je zavolána funkce pro vytváření modelu predikce [25]. V této funkci se vytvoří Rekurentní neuronová síť, která má, jako výstup hustou vrstvu [26]s jedním neuronem a její výsledek je hodnota, která je cena dalšího časového kroku.

U této funkce se zadávají následující proměnné:

- Proměnná **sequence_length** – počet dnů v datovém setu
- Proměnná **n_features –** počet funkcí, které jsou zadávány každé sekvenci.
- Proměnná **units –** počet buněk.
- Proměnná **cell** – nastavuje typ buněk v tomto případě byly zvoleny LSTM, ale mezi další možností může patřit například SimpleRNN nebo GRU
- Proměnná **n_layers** – nastavuje počet vrstev LSTM
- Proměnná **dropout** – nastavuje míru ukončení pro každou vrstvu rekurentní neuronové sítě. Hodnota je od 0 (0 %), až po 1 (100 %).
- Proměnná **loss** – je nastavení ztrátové funkce z knihovny TensorFlow,
  Ztrátová funkce je nastavena na hodnotu "huber_loss" s defaultně nastavenými parametry. Samozřejmě se dají využít jiné Účelové funkce z knihovny TensorFlow.
- Proměnná **optimizer –** použitý optimalizační algoritmus.
- Proměnná **bidirectional** – nastavení, zda v tomto neuronová síť je oboustranná.
  Tato proměnná je nastavená na False, a tudíž není využitá.

4. Dále se začne tento model trénovat na datech získaných v prvním bodě. Při trénování se model díky zpětnému volání v knihovně TensorFlow Model tf.keras.callbacks.ModelCheckpoint ukládá. A také se ukládají i nastavené váhy modelu a to, pokud byl při epoše dosažen optimálnější výsledek.

[25] Funkce je k naleznutí v souboru *swagger_server/controllers/stock_ape_controller.py* na řádku 469
[26] Hustá vrstva je pravidelná hluboce spojená vrstva neurální sítě.

5. Po doběhnutí nastavených počtu epoch je model natrénovaný a následuje testování[27]. Ze kterého nám vychází finální data pro předpovězenou cenu, průměrnou absolutní chybu, přesnost predikce a datum predikce podle stanoveného kroku.

[27] Funkce je k naleznutí v souboru *swagger_server/controllers/stock_ape_controller.py* na řádku 587

ZÁVĚR

Cílem této bakalářské práce bylo navrhnout a vytvořit aplikaci pro investory, kteří by si chtěli vyzkoušet investování bez rizika ztracení vlastní kapitál nebo jen získat informace o akcii s možností pro předpovídání individuální akcie nebo indexu.

V teoretické části byl představen koncept akciových trhů. Byl vysvětlen všeobecný pojem akcie. Dále byly objasněny faktory, které potenciálně pohybují cenou akcie, finanční ukazatele, které se využívají při porovnávání akcií. Následuje představení možností predikce akcií, které byly zváženy pro implementaci v této bakalářské práci. Poté je představen vývoj nativních aplikací, představeny designové architektury, které se využívají pro jejich vývoj. Následuje popis operačního systému Android, jeho historie a význam API levelů, představeny základní komponenty, ze kterých se skládají aplikace na této platformě. Představeno IDE Android Studio a popsány nejvýznamnější programovací jazyky využívané pro vývoj aplikací pro Android. Následně je popis operačního systému iOS, ve kterém je popsána historie a současnost, IDE Xcode a programovací jazyky využívané pro vývoj aplikací na tomto operačním systému.

V praktické části jsou popsány funkcionální a nefunkcionální požadavky pro mobilní aplikaci. Následuje popis vývoje aplikace pro Android v programovacím jazyku Java, dále jsou popsány obrazovky aplikace a jakou funkcionalitu od nich uživatel může očekávat. Dále je popsán způsob komunikace se serverem a detailně popsán centrální prvek aplikace recycler view, do kterého se ukládají data o akciích. Nakonec této kapitoly jsou popsány data, které se ukládají do mobilního zařízení a jak se s nimi nakládá. V poslední kapitole je popsán implementovaný API server, jeho funkcionalita GET požadavky využívané v mobilní aplikaci. Na konci této bakalářské práce je popsána predikce akcií pomocí neuronové sítě, která byla implementována do serveru.

Pro budoucí vývoj tohoto projektu je dále plánováno doplnit do serveru další zdroj informací, pokud by první zdroj selhal. Implementovat další metodu predikce akcii nebo potenciálně vytvořit vlastní. Do aplikace je plánováno přidání nastavení části parametrů predikce. A nakonec rozšířit funkčnost i na trh s kryptoměnami.

SEZNAM POUŽITÉ LITERATURY

[1] SOBEL, Robert. *The Big Board: A History of the New York Stock Market*. 1. Beard Books, May 2000. ISBN 1893122662.

[2] BARBERIS, Nicholas a Andrei SHLEIFER. *Style investing: Journal of Financial Economics* [online]. science direct, 2003 [cit. 2021-4-28]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0304405X03000643?casa_token=zdn1QC9rQJEAAAAA:U_oxshOw19qDIuD3WMsoRJOsHKZSUpSCeKs_uhhpFRSTDHz7b3RnPq0HnI03ukzXx_guI6A4#SEC4

[3] KELLY, ROBERT C. *Company Earnings and EPS: Everything Investors Need to Know. Investopedia* [online]. 2020 [cit. 2021-4-28]. Dostupné z: https://www.investopedia.com/articles/basics/03/052303.asp

[4] FERNANDO, JASON. Earnings Per Share (EPS): What Is Earnings Per Share – EPS? *Investopedia* [online]. 2020 [cit. 2021-5-3]. Dostupné z: https://www.investopedia.com/terms/e/eps.asp

[5] FERNANDO, JASON. Price-to-Earnings Ratio – P/E Ratio. *Investopedia* [online]. 2021 [cit. 2021-5-3]. Dostupné z: https://www.investopedia.com/terms/p/price-earningsratio.asp

[6] FERNANDO, JASON. Debt-To-Equity Ratio (D/E). *Investopedia* [online]. 2021 [cit. 2021-5-3]. Dostupné z: https://www.investopedia.com/terms/d/debtequityratio.asp

[7] KENTON, WILL. Price/Earnings-to-Growth (PEG) Ratio: What Is the Price/Earnings-to-Growth (PEG) Ratio? *Investopedia* [online]. 2021 [cit. 2021-5-3]. Dostupné z: https://www.investopedia.com/terms/p/pegratio.asp

[8] Mobile Operating System Market Share Worldwide. *Statcounter* [online]. [cit. 2021-04-18]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide

[9] MVC [online]. Cambridge, 2018 [cit. 2021-04-20]. Dostupné z: https://cs50.harvard.edu/ap/2021/curriculum/x/references/mvc.pdf. Harvard.

[10] BOODHOO, Jean-Paul. MICROSOFT. *Design Patterns: Following the MVP* [online]. MSDN Magazine Issues, 2006 [cit. 2021-04-23]. Dostupné z: https://docs.microsoft.com/en-us/archive/msdn-magazine/2006/august/design-patterns-model-view-presenter

[11] SMITH, Josh. *Patterns – WPF Apps With The Model-View-ViewModel Design Pattern* [online]. 2009, **24**(02) [cit. 2021-04-20]. Dostupné z: https://docs.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern

[12] Chinetha, K., et al. "An Evolution of Android Operating System and Its Version." *International Journal of Engineering and Applied Sciences*, vol. 2, no. 2, Feb. 2015.

[13] <uses-sdk>: What is API Level? *Developer.android* [online]. 18. 11. 2020 [cit. 2021-4-29]. Dostupné z: https://developer.android.com/guide/topics/manifest/uses-sdk-element#ApiLevels

[14] Application Fundamentals. *Developer.android.com* [online]. 2021-02-23 [cit. 2021-04-23]. Dostupné z: https://developer.android.com/guide/components/fundamentals

[15] Understand the Activity Lifecycle: Lifecycle callbacks. *Developer.android.com* [online]. Google Developers, 2020 [cit. 2021-4-28]. Dostupné z: https://developer.android.com/guide/components/activities/activity-lifecycle

[16] A simplified illustration of the activity lifecycle. *Developer.android.com* [online]. Google Developers, 2020 [cit. 2021-4-28]. Dostupné z: https://developer.android.com/guide/components/images/activity_lifecycle.png

[17] *Run apps on the Android Emulator* [online]. 2020-10-12. [cit. 2021-04-23]. Dostupné z: https://developer.android.com/studio/run/emulator#avd

[18] *Android's Kotlin-first approach: Why is Android development Kotlin-first* [online]. 2021-01-22. [cit. 2021-4-25]. Dostupné z: https://developer.android.com/kotlin/first#why

[19] *Android's Kotlin-first approach: Why is Android development Kotlin-first* [online]. 2021-01-22. [cit. 2021-4-25]. Dostupné z: https://developer.android.com/kotlin/first#why

[20] Verma, Nishkarsh and SAMBHAV, SAURABH, Development of iOS: A Revolutionary Transformation and the Future (2020). International Journal of Advanced Research in Engineering and Technology, 11(6), 2020, pp. 445-454, Available at SSRN: https://ssrn.com/abstract=3656930

[21] Xcode: Overview. *Developer.apple* [online]. Apple, 2021 [cit. 2021-4-30]. Dostupné z: https://developer.apple.com/documentation/xcode/#Overview

[22] At a Glance: Single-Window Interface. *Developer.apple* [online]. Apple, 2018 [cit. 2021-4-30]. Dostupné z: https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/Art/XC_O_WrkspaceWindow_nocallout_2x.png

[23] About Swift. *Swift.org* [online]. Apple, 2021 [cit. 2021-4-30]. Dostupné z: https://swift.org/about/

[24] Programming with Objective-C: About Objective-C. *Developer.apple* [online]. Apple, 2014-09-17 [cit. 2021-4-30]. Dostupné z: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html

[25] *ViewModel Overview: The lifecycle of a ViewModel* [online]. 2021-04-28 [cit. 2021-5-2]. Dostupné z: https://developer.android.com/topic/libraries/architecture/viewmodel#lifecycle

[26] *ConstraintLayout: Developer Guide* [online]. 2021-02-24. [cit. 2021-4-25]. Dostupné z: https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout#developer-guide

[27] MPAndroidChart. JAHODA, Philipp. *Github.com* [online]. [cit. 2021-4-28]. Dostupné z: https://github.com/PhilJay/MPAndroidChart

[28] Android Simple Tooltip. JUNIOR, Douglas Nassif Roma. *Github.com* [online]. 2019 [cit. 2021-4-28]. Dostupné z: https://github.com/douglasjunior/android-simple-tooltip

[29] *LiveData* [online]. 2021-04-28 [cit. 2021-5-2]. Dostupné z: https://developer.android.com/reference/android/arch/lifecycle/LiveData

[30] Alpha Vantage API Documentation. *Www.alphavantage.co* [online]. 2021 [cit. 2021-4-28]. Dostupné z: https://www.alphavantage.co/documentation/

[31] OpenAPI Specification. *Swagger.io* [online]. SmartBear Software., 2021 [cit. 2021-4-28]. Dostupné z: https://swagger.io/specification/

[32] ROCKIKZ, Abdou R. How to Predict Stock Prices in Python using TensorFlow 2 and Keras. *Github* [online]. 2021 [cit. 2021-5-3]. Dostupné z:

https://github.com/x4nth055/pythoncode-tutorials/tree/master/machine-learning/stock-prediction

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AOSP	Android Open Source Project
API	Application Programming Interface
APK	Android Package
DEX	Dalvik Executable
D\E	Debt to Equity ratio
EPS	Earnings Per Share
FCF	Free Cash Flow
GRU	Gated Recurrent Unit
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IDL	Interface Description Language
I/O	Input Output
iOS	iPhone Operating System
IP	Internet Protocol
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
LSTM	Long Short-Term Memory
MVC	Model View Controller
MVP	Model View Presenter
MVVM	Model View ViewModel
NLP	Natural Language Processing
OS	Operating System
PEG	Price/Earnings to Growth

P/E	Price to Earnings
REST	Representational State Transfer
RNN	Random Neural Network
SQL	Structured Query Language
TBA	To Be Announced
UI	User Interface
YAML	YAML Ain’t Markup Language

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: SEZNAM PŘÍLOH NA CD-ROM

- Stock_apk – adresář obsahující instalační soubor aplikace
- Stock_app – adresář obsahující projektovou složku aplikace
- Stock_server – adresář obsahující projektovou složku serveru