



VILNIAUS UNIVERSITETAS  
MATEMATIKOS IR INFORMATIKOS FAKULTETAS  
KOMPIUTERIJOS KATEDRA

Kursinis darbas

## **Mobilioji akcijų kainų stebėjimo sistema**

Mobile stock market monitoring system

Atliko:

3 kurso, 2 grupės studentas,  
Tautvydas Milčiūnas

parašas

Vadovas:

dr. Joana Katina

Vilnius  
2017

# Turinys

<b>Santrauka</b>	<b>3</b>
<b>Summary</b>	<b>4</b>
<b>Ivydas</b>	<b>5</b>
<b>1. Panašių sistemų analizė</b>	<b>6</b>
1.1. „My Stocks Portfolio and Widget“ analizė . . . . .	6
1.2. „Stocks - Realtime Stock Quotes“ analizė . . . . .	6
1.3. Analizės išvados . . . . .	6
<b>2. Teorinis sistemos modelis</b>	<b>8</b>
2.1. Sistemos reikalavimai . . . . .	8
2.2. Vartotojo sąsaja . . . . .	8
<b>3. Praktinis programėlės įgyvendinimas</b>	<b>10</b>
3.1. Technologijų pasirinkimas . . . . .	10
3.1.1. „Expo SDK“ platforma . . . . .	10
3.1.2. „React Native ir Redux“ . . . . .	10
3.1.3. „Genymotion“ . . . . .	10
3.2. Išorinių duomenų šaltinių pasirinkimas . . . . .	11
3.3. Sistemos struktūra . . . . .	11
<b>4. Pagrindinio funkcionalumo įgyvendinimas</b>	<b>12</b>
4.1. Vartotojo akcijų aplanko kūrimas . . . . .	12
4.2. Akcijos išplėsto turinio lango kūrimas . . . . .	13
4.3. Valiutos keitimo lango kūrimas . . . . .	14
<b>Išvados ir rekomendacijos</b>	<b>16</b>
<b>Literatūros šaltiniai</b>	<b>17</b>
<b>Priedai</b>	<b>18</b>

## **Santrauka**

Šio kursinio darbo tikslas - sukurti automatizuotą akcijų stebėjimo sistemą, kuri gautų akcijų duomenis iš viešai platinamų tarnybų. Gauti duomenys bus atvaizduojami dvejais būdais - santraukos ir išplėstiniu. Santraukoje bus atvaizduojami pagrindiniai akcijos duomenys. Išplėstiniame apipavidalinime atvaizduojama informacija - detalūs akcijos duomenys kartu su istoriniais kainų pakitimais. Taip pat sistema turi turėti vartotojo asmeninį akcijų portfelį į kurį būtų galima pridėti norimas akcijas įvedus akcijos trumpinį į paiešką. Tam, kad būtų patogiau suprasti akcijų kainas visų šalių programėlės vartotojams, reikia sukurti valiutų keitimo funkcionalumą.

# Summary

## **Mobile stock market monitoring system**

The main objective of this work is to create an automated stock monitoring system. This system would fetch data from external providers. Fetched data then would be destructed and displayed as two formats: summarized and extended. In summarized format, common data is displayed. In extended format, full data is displayed along with historical price changes in graphical format. In addition, the system should provide user with ability to create personal stock portfolio which could be expanded by adding stock found with interactive search. To understand different stock prices, currency exchange functionality should be created.

## Ivyadas

Šiuo darbu siekama išsiaiškinti kintančios akcijų rinkos pateikiamus duomenis, juos išanalizuoti ir pateikti vartotojui patraukliu būdu. Sukurta programėlė bus prieinama vartotojams turintiems įrenginius naudojančius „Android“ operacinę sistemą, kadangi didžioji dalis mobiliųjų įrenginių vartotojų naudoja būtent šią operacinę sistemą. Šio kursinio darbo metu iškeltas uždavinys buvo sukurti sistemą, kuri leistų stebėti realiu laiku kintančias akcijas. Tam, kad galėčiau įgyvendinti tokį uždavinį norėjau išanalizuoti kuo daugiau skirtingų programėlių ir suprasti kuo jos skiriasi, bei ko trūksta, kad jos būtų geresnės ir pasinaudojus tokiais atradimais sukurti sistemą, kuri būtų paprasta naudoti, bet joje netrūktu informatyvaus turinio ir funkcionalumo. Buvo nuspręsta šią programėlę įgyvendinti sukuriant du akcijų atvaizdavimo būdus: programėlėje įkoduotų akcijų trumpinių atnaujinimą iš išorinių šaltinių ir vartotojų asmeniniuose portfeliuose pridėtų akcijų atnaujinimą ir gavimą. Du skirtingi duomenų atvaizdavimo tipai yra išskirti todėl, kad ne visi vartotojai nori matyti iš anksto nustatytų akcijų gavimą realiu laiku, o nori pridėti savo individualias akcijas ir matyti jų pasikeitimus asmeniniuose portfeliuose. Pagrindinė problema, su kuria susidūriau kurdamas sistemą - tai duomenų atvaizdavimas, kuris būtų intuityvus vartotojui, kuris nieko nežino apie akcijų rinką.

Darbo tikslas - sukurti automatizuotą akcijų stebėjimo sistemą, kuri gautų akcijas iš viešai platinamų tarnybų ir jas tiek įkoduotų akcijų trumpinių sąrašė, tiek vartotojų asmeniniuose portfeliuose. Darbo uždaviniai - išanalizuoti panašias programėles, suprasti jų turinį, dizaino sprendimus, teikiamą naudą ir pagrindines sistemų savybes. Sudaryti teorinį sistemos modelį aprašant pradines sistemos dizaino ir funkcionalumo dalis. Aprašyti rastas tinkamiausias technologijas programėlės kūrimui, bei jas panaudoti sistemos įgyvendinimo procese.

Pasiekti rezultatai galutininėje programėlėje: sukurta funkcionuojanti akcijų stebėjimo sistema ir vartotojų akcijų portfeliai. Šiuose sąrašuose esančias akcijas galima praskleisti ir pažiūrėti akcijų istorinę kainų kitimo raidą grafiniu formatu, bei planetsnius akcijos duomenis. Taip pat sukurta valiutos keitimo funkcionalumas, leidžiantis vartotojams iš skirtingų valstybių išsiversti valiutų kainas į sau suprantamą valiutą.

# 1. Panašių sistemų analizė

Prieš sistemos kūrimą svarbu surasti ir išanalizuoti panašias programėles, kurios padėtų susidaryti dizaino, bei funkcionalumo įvaizdį. Pirmoji pasirinkta programėlė - „My Stocks Portfolio and Widget“. Antroji pasirinkta programėlė - „Stocks - Realtime Stock Quotes“. Abi aplikacijos yra skirtos akcijų rinkos stebėjimui, tad analizuodamas jas susidarysiu išspūdi, kaip turėtų būti tinkamai suformuluotas mano aplikacijos dizainas ir funkcionalumas.

## 1.1. „My Stocks Portfolio and Widget“ analizė

Ši programėlė (priedas A) skelbia naujienas iš akcijų rinkos ir leidžia vartotojui rankiniu būdu pridėti norimas stebėti akcijas. Bendras aplikacijos apipavidalinimas labai paprastas, tačiau informatyvus. Informacijos pagrindiniame lange nėra daug, tačiau paspaudus ant pasirinktos akcijos atvaizduojama visa reikalinga informacija kartu su istoriniais akcijos kainų kitimo grafikai (priedas B). Programėlė neturi automatinio akcijų atnaujinimo, reikia paspausti mygtuką, kad tai būtų įvykdyta. Taip pat yra galimybė nusipirkti akcijų ir matyti, ar uždirbama ar pinigai yra prarandami. Suteikiama galimybė kurti skirtingus akcijų portfelius, kur galima pridėti skirtingas grupes akcijų paketų ir juos taip grupuoti.

## 1.2. „Stocks - Realtime Stock Quotes“ analizė

Ši programėlė (priedas C) turi kelis skirtingus funkcionalumus. Informacijos atnaujinimas vyksta realiu metu, kai yra atidarytos akcijų rinkos. Pirmą kartą įjungus programėlę atvaizduojamas langas su daugybe skirtingų akcijų rinkų ir akcijų paketų. Yra atvaizduojama daug informacijos pagrindiniame lange ir yra sunku suprasti kokią informaciją skaičiai pristato. Paspaudus ant konkrečios akcijos susidūriau su dar didesniu informacijos kiekiu kartu su istoriniais kainų kitimo grafikai, naujienomis apie konkrečią akciją, jos detalią apžvalgą ir finansiniais duomenimis. Ši aplikacija pasižymi labai dideliu ir konkrečiu kiekiu apie kiekvieną konkrečią akciją, tačiau nėra labai intuityvi ir sukelia informacijos pertekliaus jausmą. Šioje aplikacijoje galima susikurti savo portfelius ir pirkti akcijas. Tam, kad galima būtų pirkti akcijas privaloma prisijungti prie sistemos naudojant savo realius duomenis.

## 1.3. Analizės išvados

Funckionalumas	„My Stocks Portfolio and Widget“	„Stocks - Realtime Stock Quotes“
Registracija	-	+
Akcijų rinkos	-	+
Aplanko kūrimas	+	+
Akcijų naujinimas	-	+
Akcijų pirkimas	+	+

1 lentelė. Panašių programėlių analizė

Lentelėje 1 pavaizduoti abiejų nagrinėtų aplikacijų privalumai ir trūkumai. Aplikacija „My Stocks Portfolio and Widget“ neturėjo prisijungimo galimybių, nebuvo galimas stebėti iš anksto

sukurtų akcijų rinkų, bei nebuvo automatinio akcijų atnaujinimo vos joms pasikeitus. Aplikacijoje „Stocks - Realtime Stock Quotes“ visi funkcionalumai įgyvendinti puikiai, tačiau pateikta informacija yra nepaaiškinta ir vartotojui, kuris sistemą mato pirmą kartą būtų sunku suprasti kas yra pavaizduota tam tikruose languose. Abi aplikacijos taikė vartotojo aplanko kūrimą, kuriame galima pridėti norimas akcijas pasinaudojus paieška.

## 2. Teorinis sistemos modelis

### 2.1. Sistemos reikalavimai

Funkciniai reikalavimai:

1. Turi būti įgyvendintas valiutų keitimo įrankis
2. Turi būti įgyvendintas grafikų atvaizdavimas ir keitimas
3. Turi būti įgyvendintas vartotojo individualus akcijų aplankas.
4. Turi būti įgyvendintas akcijų duomenų gavimas iš išorinių šaltinių.

Nefunkciniai reikalavimai:

1. Aiškiai atvaizduojama informacija
2. Patogus ir intuityvus valdymas

### 2.2. Vartotojo sąsaja

1		
2		
3	4	5

6		
7		
3	4	5

*1 pav. Akcijų kainų ir vartotojo aplanko atvaizdavimas*

Paveikslėlyje 1 yra pavaizduotas akcijų kainų ir vartotojo aplanko modelis, kuris yra suskirstytas į skirtingas dalis, kurios atstoja skirtingą funkcionalumą. Kairiajame paveikslėlyje yra atvaizduojamos akcijos, kurias įvedžiau pats, ir naujų akcijų pridėti į sąrašą yra neleidžiama. 1 - laukas yra skirtas lango pavadinimui ir sąrašo atnaujinimo funkcionalumui. 2 - sąrašo laukas skirtas sutrumpintiems akcijoms duomenims saugoti. Dešiniajame paveikslėlyje yra atvaizduojamas vartotojo aplankas į kurį vartotojas gali pridėti naujų akcijų. 6 - laukas skirtas lango pavadinimui, bei naujų akcijų pridėjimui į vartotojo akcijų sąrašą. 7 - vartotojo akcijų sąrašo laukas, kuris atvaizduoja sutrumpintus akcijos duomenis. 3 - naršymo juostos liečiamas laukas, skirtas pereiti iš anksto nustatytų akcijų sąrašą. 4 - naršymo juostos liečiamas laukas skirtas pereiti į vartotojo akcijų aplanko langą. 5 - naršymo juostos liečiamas laukas skirtas pereiti į valiutų keitimo langą.





### 3. Praktinis programėlės įgyvendinimas

Norint įgyvendinti mobilią akcijų stebėjimo sistemą man pirmiausia reikėjo išsiaiškinti, kur galiu gauti nemokamus ir patikimus akcijų duomenis.

#### 3.1. Technologijų pasirinkimas

Programėlei realizuoti pasirinkau „Facebook“ įmonės sukurtą „JavaScript“ programavimo kalbos „React Native“ biblioteką. Programos paleidimui, kompiliavimui ir versijavimui pasirinkau nepriklausomų programuotojų sukurtą nemokamą produktą „Expo SDK“. Savo programėlę testavau nemokama „Android“ modeliavimo programa – „Genymotion“. Naršymą programėlės viduje įgyvendinau pasitelkdamas „React Navigation“ įrankį. Įvairių komponentų dalių ir būsenų išsaugojimą ir pernešimą programėlės viduje įgyvendinti pasitelkiau „Redux“ įrankį.

##### 3.1.1. „Expo SDK“ platforma

Kuriant šią programėlę, man buvo labai svarbu kuo labiau supaprastinti programėlės kompiliavimo ir versijavimo procesą. Tam, kad nereikėtų nuolatos keisti kompiliavimo ir programėlės sudarymo proceso rankiniu būdu pasirinkau „Expo SDK“, kaip platformą savo programėlei. Ši platformą labai palengviną programėlės kompiliavimo ir kūrimo procesą automatiškai konfigūruodama kompiliavimą kiekvieną kartą pakeitus programinį kodą. Programuojant šioje platformoje man tapo prieinamas būdas naudoti „React Native“ biblioteką ir visą programą parašyti „JavaScript“ kalba be jokių sudėtingų kompiliatoriaus konfigūracijų. Ši platforma tapo labai naudinga tuo, kad ją įrašius iškart galima naudotis daugybe integruotų bibliotekų, tokių, kaip pastarasis „React Native“, įvairios dizaino ir funkcionalumo bibliotekos. Taip pat ši platforma labai patogi darbui su "Android" sistema dėl integruotų įrankių, kurie leidžia kompiliuotą kodą iškart atvaizduoti mobiliarioje sistemoje ir atnaujinti vos atnaujinus programinio kodo dalį.

##### 3.1.2. „React Native ir Redux“

Funkcionalumui įgyvendinti pasirinkau „React Native“ programavimo kalbą. Ši programavimo kalba yra paremta savaisiais(ang. Native) mobiliųjų įrenginių programiniais kodais ir subendrina programavimą kelioms skirtingoms platformoms iš karto. Ši programavimo kalba yra sukurta, kaip „JavaScript“ kalbos biblioteka. Kadangi mano programėlėje yra būtinas bendravimas tarp skirtingų komponentų ir informacijos dalijimasis nusprendžiau tai įgyvendinti naudodamasis „Redux“ biblioteka. Ši biblioteka leidžia lengvai valdyti aplikacijos atmintyje laikomas savybes ir komponentų parametrus ir juos perduoti kitiems komponentams esantiems ne būtinai tame pačiame lygmenyje ar tame pačiame navigacijos lygyje.

##### 3.1.3. „Genymotion“

Programėlės testavimui pasirinkau nemokamą modeliavimo įrankio „Genymotion“ versiją. Šis įrankis leidžia sukompiliuotą kodą atvaizduoti sumodeliuotame „Android“ įrenginyje kaskart atnaujinus programinį kodą. Taip smarkiai padidinamas programavimo darbo našumas, nes testavimas gali vykti ypatingai greitai ir nėra būtinybės programėlės testuoti tikrame įrenginyje.

### 3.2. Išorinių duomenų šaltinių pasirinkimas

Pasirinkau apjungti kelis skirtingus duomenų šaltinius, nes visi suteikė skirtingo funkcionalumo. „Yahoo Stocks API“ naudoju pagrindinių akcijų duomenų gavimui. Ši paslaugų tiekimo tarnyba ją kviečiant su norimu akcijos simboliu grąžina sąrašą duomenų apie pasirinktą akciją. Istoriniams akcijų pakitimų grafikams atvaizduoti naudoju „Quandl“ servisą, kuris grąžina akcijų kainas pasirinktam laikotarpiui. Gautiems istoriniams kainų pakitimų grafikams atvaizduoti naudojausi išorine paslauga „HighCharts“. Ši paslauga leidžia atvaizduoti norimus duomenis pasirinktais grafikais, kurie pasirenkami siunčiant sukurtą konfigūraciją į išorinį paslaugų tiekėją.

### 3.3. Sistemos struktūra

Kuriant mobilią aplikaciją su „React Native“ ir „Redux“ technologijomis yra labai svarbu tinkamai struktūrizuoti projektą. Toliau pateiksiu mano sistemos struktūros pseudokoda:

- komponentai
  - namų/
  - portfelis/
- papildiniai/
- konstantos/
  - isoriniai-servisai/
- redux/
  - reducers
  - store
- naršymo-maršrutai/
- aplikacijos-pradžios-failas

Programai pradėjus darbą pirmiausia yra kreipiamasi į failą *aplikacijos-pradžios-failas* kuris yra tėvinis projekto failas. Šis failas sukuria programoje naudojamus komponentus ir užregistruoja pradinę aplikacijos būseną, kuri vėliau yra naudojama saugoti parametrus, kurie yra perduodami skirtingose aplikacijos vietose esantiems komponentams. Pradinis sistemos būsenos registravimas vyksta aplanke *redux/store.js*, o atskirų komponentų būsenos surenkamos *redux/reducers.js*. Tuomet yra kreipiamasi į aplanką *naršymo-maršrutai/*, kuris nusako, koks turi būti pirmasis komponentas, kurį sistema priima ir atvaizduoja vartotojui. Pirmasis komponentas kurį įkelia mano sistema yra failas *komponentai/namų*. Tam, kad šis komponentas galėtų atvaizduoti mano norimą informaciją, buvo privaloma gauti pradinius naršymo maršrutų, programos būsenos ir išorinių servisų duomenis. Galiausiai namų komponente yra kviečiami išoriniai servais iš anksčiau nurodyto failo ir viskas yra saugoma programos globalioje būsenoje.

## 4. Pagrindinio funkcionalumo įgyvendinimas

Programėlės funkcionalumas susideda iš daugelio nedidelių funkcinių modulių, kurie yra paskirti atlikti tam tikrą veiksmų seką ir daugeliu atvejų grąžinti vienokį ar kitokį rezultatą. Funkcionalumai susiję su akcijų gavimu ir apipavidalinimu yra kuriami taikant teorines žinias perskaičius [1] ir [2] šaltinius, bei išanalizavus panašias programėles. Šiame skyriuje detaliau aptarsiu pagrindinius funkcionalumus ir kaip juos įgyvendinau.

### 4.1. Vartotojo akcijų aplanko kūrimas

Vienas pagrindinių programėlės funkcionalumų yra vartotojo akcijų aplanko kūrimas (pavaizduota priede E). Šis funkcionalumas yra pasiekiamas `/areas/portfolio/PortfolioScreen.js`. Šiame lange yra pateikiamas akcijų sąrašas, kurias vartotojas pridėjo naudodamasis pliuso mygtuką. Paspaudus mygtuką yra atidaroma funkcija `fetchSingleStocks(item)`. Ši funkcija priima vartotojo įvestą raktažodį ir vykdo pseudokodą (pavyzdys išeities kode 1). Pavyzdžio antroje eilutėje pavaizduotas išorinis paslaugų tiekėjas, kuriam pateikus norimos akcijos trumpinį yra grąžinamas duomenų apie akciją sąrašas. Šis sąrašas tuomet yra trečiojoje eilutėje interpretuojamas kaip objektas. Galiausiai ketvirtojoje eilutėje gautas objektas yra siunčiamas į aplikacijos atmintį (ang. state) ir ten yra išsaugomas vėlesniam panaudojimui.

```
1 fetchSingleStocks(akcijos – trumpinys) {  
2   const atsakas = gauti('http://d.yimg.com/aq/autoc?query=${akcijos –  
   trumpinys}&region=US&lang=en-US')  
3   const atsakasJson = atsakas.json();  
4   nustatomeGlobaliaBusena({paieskos – rezultatai: atsakasJson.akcija })  
5 }
```

*1 išeities kodas. Funkcijos fetchSingleStocks() pseudokodas.*

Įvykdžius funkciją `fetchSingleStocks(item)` programėlės atmintyje turiu naują akcijos objektą. Tam, kad šį objektą atvaizduočiau pridėdau į sąrašą naudodamasis „React Native“ komponentu `ListView`. Šis komponentas priima objektą ir perduoda šį objektą į funkciją `renderRow(item)`. Ši funkcija leidžia apdoroti vieną sąrašo eilutę su gautais duomenimis ir grąžinti eilutę sąrašui. Tada yra kviečiama funkcija `fetchYahooSingle(pasirinktasSimbolis)`, kuri ieško pasirinktos akcijos „Yahoo Finance“ duomenų bazėje (pavyzdys išeities kode 2). Antroje eilutėje kviečiamas išorinis servisas, kuris radęs akciją ją grąžina kaip masyvą. Toliau šeštoje eilutėje vykdoma tikrinimą ar akcija yra grąžinta ir turi reikiamus duomenis. Jeigu akcija turi reikiamus duomenis aštuntoje eilutėje tikriname ar mūsų aplikacija jau turi tokią akciją savo būsenoje. Jeigu neturi, penkioliktoje eilutėje pridėdame akciją į programėlės atmintį. Pats svarbiausias žingsnis yra septynioliktoje eilutėje, čia išsaugome rastą naują akciją į mobilaus prietaiso atmintį. Atlikus septyniolikto žingsnį ir perkrovus aplikaciją, vartotojas portfelio lange rastų išsaugotą akciją.

```

1 fetchYahooSingle(pasirinktasSimbolis) {
2   const atsakas = gauti('https://query.yahooapis.com/v1/public/yql?q=
      select * from yahoo.finance.quotes where symbol in ({$
      pasirinktasSimbolis})&format=json&env=store/datatables.org/
      Falltableswithkeys&callback=')
3
4   const atsakasJson = atsakas.json();
5   const portfelis = this.state.portfolio;
6   if (atsakasJson nera tuscias ir turi duomenis 'open/ask') {
7     let rastasPortfelyje = [];
8     for(sukame cikla per 'portfelis' ir ieskome ar nera jau tokio
        simbolio) {
9       rastasPortefyle.push(atsakasJson);
10    }
11  }
12
13  if(rastasPortfelyje === 0) {
14    portfolio.push(atsakasJson)
15    nustatomeGlobaliaBusena({ portfelis: atsakasJson });
16
17    AsyncStorage.setItem('portfelis');
18  }
19  else {
20    Alert('Tokia akcija jau yra sarase!');
21  }
22 }

```

2 išeities kodas. Funkcijos *fetchYahooSingle()* pseudokodas.

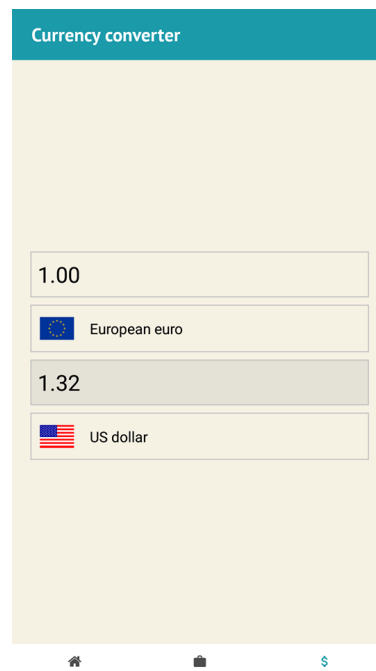
Norint atnaujinti vartotojo akcijų portfelio sarašą yra kviečiama funkcija *fetchYahooRefresh(portfelis)*. Ši funkcija kviečia išorinį servisą tiek kartų, kiek yra akcijų vartotojo portfelyje ir gauną atitinkamą kiekį masyvų. Kiekvieną masyvo elementas yra patikrinamas ar yra ne tuščias ir ar yra gauti atitinkami duomenys akcijai sėkmingai atvaizduoti. Jeigu duomenys yra gauti, senasis vartotojo portfelis yra ištrinamas iš programėlės atminties ir iš telefono atminties. Tuomet yra iš naujo išsaugojamas. Taip yra išvengiama pasikartojančių duomenų ir gaunami naujausi akcijų duomenys.

## 4.2. Akcijos išplėsto turinio lango kūrimas

Kiekviena akcija turi daugiau duomenų negu yra pavaizduota pagrindiniame ar vartotojo portfelio lange. Paspaudus ant vienos iš šių akcijų vartotojas yra nukreipiamas į išplėstą akcijos turinio langą (pavyzdys priede H). Perėjus į šį langą yra kviečiama funkcija *fetchHistoricalData(akcijosSimbolis, dataIki, siandienosData)*. Ši funkcija kviečia išorinį servisą „QUANDL“, kuris pagal pateiktus duomenis: akcijos simbolį, pradžios datą ir pabaigos datą, gražina istorinius akcijos duomenis. Šie duomenys yra tokie, kaip kiekvienos dienos akcijos kaina rinkos atsidarymo, užsidarymo metu. Gauti duomenys yra įrašomi į programėlės atmintį ir yra kviečiamas kitas modalinis langas *ChartScreen*. Šis langas atlieka grafiko atvaizdavimo funkcionalumą su duomenimis gautais iš *fetchHistoricalData* funkcijos. Šiame lange yra sudaroma grafiko konfigūracija, nustatomos ašys: x - datų masyvas, y - didžiausios ir mažiausios kainos masyvai. Tuomet yra kviečiama funkcija *highCharts*, kuri iš išorinio paslaugų tiekėjo gražina istorinį akcijos grafiką. Jeigu grafikas gražintas sėkmingai šiame lange aprašau grafiko aukštį ir plotį ir siunčiu atgal į išplėsto

akcijos turinio langą. Grįžus į minėtąjį langą yra atvaizduojamas gautas grafikas ir iš pagrindinio arba vartotojo profilio lango gauti išplėsti akcijos duomenys. Taip pat sukūriau grafiko datos imties keitimo funkcionalumą, kuris yra išdėstytas keturiais mygtukais virš grafiko. Kai yra paspaudžiamas vienas iš mygtukų, programėlės atmintyje pasikeičia laikinasis parametras nusakantis aktyvią grafiko datą, ir įvedant naująją vėl yra kreipiamasi į *ChartScreen* langą ir yra atnaujinamas išplėstinis akcijos duomenų langas nauju grafiku. Kadangi yra kviečiami skirtingi duomenų tiekėjai grafikams ir akcijos duomenims atvaizduoti, būna atvejų kai yra negaunami vieni ar kiti duomenys.

### 4.3. Valiutos keitimo lango kūrimas



3 pav. Valiutų keitimo langas

Valiutos keitimo funkcionalumo įgyvendinimas man atrodė būtina dalis, nes ne visi vartotojai naudoja vienodą valiutą akcijoms stebėti ir pirkti (lango pavyzdys paveikslėlyje 3). Todėl buvo būtina užduotis sukurti būdą vartotojams išsiversti akcijų kainas į sau priimtina valiutą. Šiame lange yra keturi laukai. Pirmasis laukas atlieka sumos įvedimo funkcionalumą (pavaizduota išeities kode 3). Šioje funkcijoje gavęs įvesties sumą, kursą ir išvesties kursą paskaičiuoju išvesties sumą. Išvesties suma yra atvaizduojama *pav. 3* trečiame laukelyje ir šis laukelis yra skirtas tik išvesties atvaizdavimui. Taip pat yra galimybė keisti valiutas paspaudus ant laukų kurie pažymėti valiutų vėliavomis. Valiutos iš kurių sąrašo galima rinktis yra įkoduotos aplikacijoje.

```

1 calculateRate(investis , investiesKursas , isvestiesKursas) {
2   const kursas = investiesKursas / isvestiesKursas ;
3   return (investis * kursas);
4 }

```

3 išeities kodas. Funkcijos *calculateRate()* pseudokodas.

Valiutų kursai yra apskaičiuojami naudojantis išoriniu „*Bank of Canada*“ teikiamu, kasdien

atnaujinamu duomenų sąrašu. Valiutų kursams gauti sukūriau funkciją *fetchRates* (pavaizduota išvestie kode 4). Šioje funkcijoje yra kviečiamas išorinis duomenų tiekėjas ir gauti duomenys yra grąžinami masyvo formatu ir išsaugomi programėlės atmintyje iki kol valiutų keitimo langas yra atidaromas iš naujo.

```
1 fetchRates() {  
2   const valiutuKursai = fetch('http://www.bankofcanada.ca/valuet/  
   observations/group/FX_RATES_DAILY');  
3   const valiutuKursaiJson => {  
4     valiutuKursai.forEach((elementas) => {  
5       return elementas.valiutosKodas;  
6     });  
7   }  
8 }
```

*4 išeities kodas. Funkcijos fetchRates() pseudokodas.*

## Išvados ir rekomendacijos

Šiame darbe buvo siekiama sukurti programėlę, kuri vartotojams suteiktų informaciją apie naujausius akcijos rinkų kainų pokyčius. Pagrindinis programėlės tikslas buvo suprogramuoti akcijų kainų stebėjimo sistemą, kur vartotojai patys galėtų pasirinkti norimas akcijas pateikę akcijos simbolį. Taip pat buvo siekiama, kad vartotojai turėtų papildomą valiutų keitimo funkcionalumą, bei galėtų detaliau apžiūrėti kiekvienos akcijos duomenis ir grafinius akcijos kainų pakitimus. Sukurtos programėlės funkcionalumai yra:

- Vartotojo akcijų portfelio kūrimas pasirenkant norimą akcijos simbolį.
- Programėlėje įkoduotų akcijų simbolių atvaizdavimas.
- Akcijų sąrašų atnaujinimas ir išsaugojimas į mobilaus įrenginio atmintį.
- Akcijos kainos pasikeitimų per nustatytą laiką grafinis atvaizdavimas ir detalus akcijos duomenų atvaizdavimas.
- Valiutų keitimas.

Rekomendacijos tolesniam darbo vystymui:

- Apjungti išorinius duomenų tiekėjus į vieną, tam, kad nebūtų duomenų paklaidų ir būtų optimizuotas paieškos, bei atvaizdavimo greitis.
- Sukurti interaktyvų akcijų pirkimo ir pardavimo žaidimą.

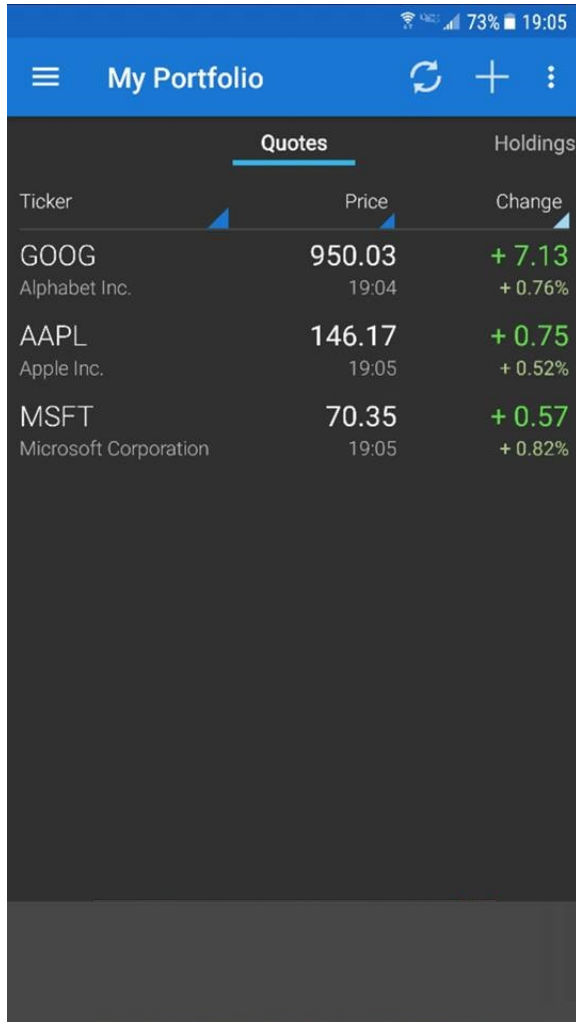


## Literatūros šaltiniai

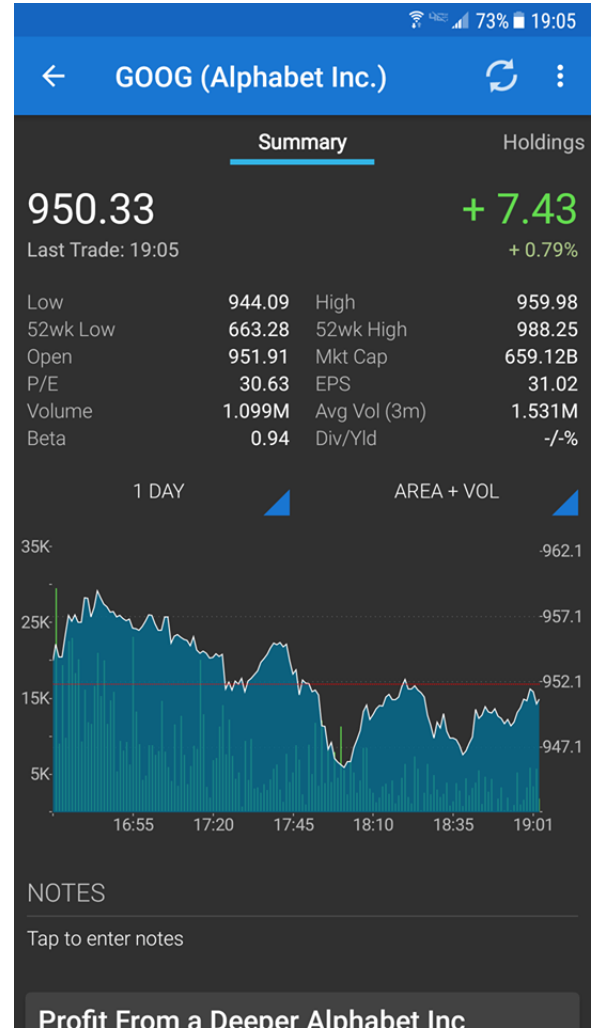
- [1] Igor Katin. On development and investigation of stock exchange model. Vilniaus universitetas. Matematikos ir informatikos institutas, 2014.  
[http://old.mii.lt/files/mii\\_dis\\_2014\\_katin.pdf](http://old.mii.lt/files/mii_dis_2014_katin.pdf).
- [2] Joana Katina. Prognozavimo problemų tyrimas virtualioje akcijų biržoje. Vilniaus universitetas. Matematikos ir informatikos institutas, 2015.  
<http://gs.elaba.lt/object/elaba:11697674/11697674.pdf>.
- [3] Orientacinių euro ir užsienio valiutų santykių nustatymo ir skelbimo tvarka.  
<https://www.ecb.europa.eu/stats/pdf/exchange/Frameworkfortheeuroforeignexchangerates.pdf>.
- [4] Akcijų kainos ir jų pokyčiai.  
<https://finance.yahoo.com/>.
- [5] Istoriniai akcijų kainų pokyčiai.  
<https://www.quandl.com/product/WIKIP/WIKI/PRICES-Quandl-End-Of-Day-Stocks-Info>.
- [6] Kasdieniai valiutų kursų pokyčiai.  
<http://www.bankofcanada.ca/rates/exchange/daily-exchange-rates/>.

# Priedai

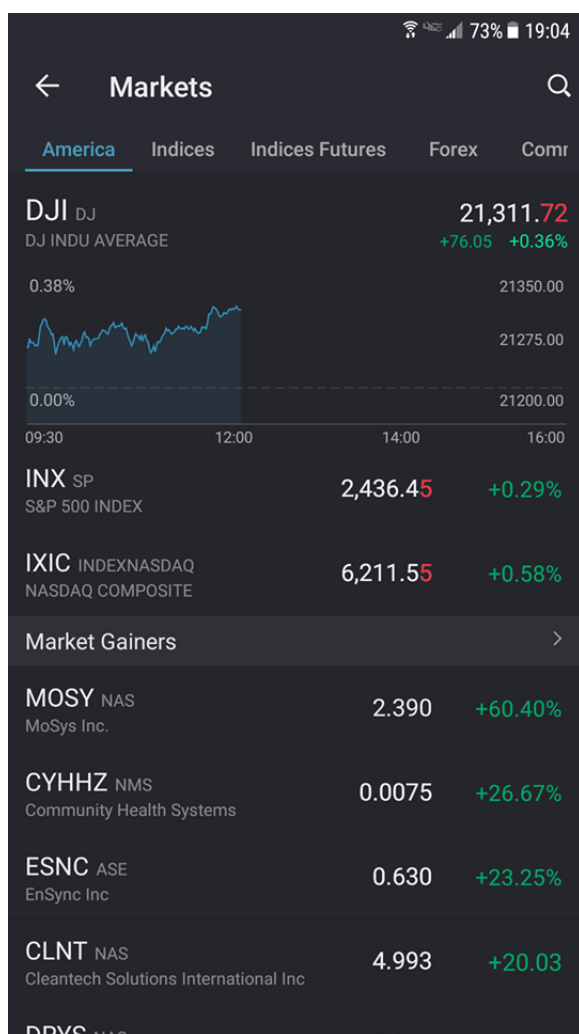
## A. Programėlė „My Stocks Portfolio and Widget“



## B. Programėlė „My Stocks Portfolio and Widget“



## C. Antroji analizuota aplikacija



## D. Antroji analizuota aplikacija



## E. Vartotojo akcijų portfelio langas

Portfolio +	
YAHOO INC 🕒 4:00pm	Price: 52.58 +0.00(+0.00%)
Alphabet Inc. 🕒 4:00pm	Price: 958.62 -1.56(-0.16%)
Alphabet Inc. 🕒 4:00pm	Price: 939.78 -2.53(-0.27%)
NVIDIA Corporation 🕒 4:00pm	Price: 151.62 -0.75(-0.49%)
Apple Inc. 🕒 4:00pm	Price: 142.27 -2.02(-1.40%)

## F. Akcijos paieškos laukas portfelio lange

← god | ×

GOOG

GOOGL

GT

a č ę è j š ų ū ž  
 q w e r t y u i o p  
 a s d f g h j k l  
 ↑ z x c v b n m ↵  
 !#☺ 🔊 , « Lietuvių » . 🔍

## G. Pradinis programėlės akcijų langas

Stocks Observation		
Yahoo! Inc. 🕒 4:00pm	Price: 52.58 +0.00(+0.00%)	
Apple Inc. 🕒 4:00pm	Price: 142.27 -2.02(-1.40%)	
Microsoft Corporation 🕒 4:00pm	Price: 70.00 +0.10(+0.14%)	
Alphabet Inc. 🕒 4:00pm	Price: 958.62 -1.56(-0.16%)	
Oracle Corporation Common Stock 🕒 4:01pm	Price: 45.09 +0.41(+0.92%)	
eBay Inc. 🕒 4:00pm	Price: 33.96 -0.13(-0.38%)	
NVIDIA Corporation 🕒 4:00pm	Price: 151.62 -0.75(-0.49%)	
United States Oil Fund 🕒 8:00pm	Price: 9.24 +0.06(+0.65%)	
Bank of America Corporation Com 🕒 4:00pm	Price: 23.43 -0.11(-0.47%)	

## H. Akcijos duomenų detalaus atvaizdavimo langas

