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

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 patogu 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 di erent stock prices, currency exchange functionality should be created.

Įvadas

Šiuo darbu siekama išsiaiškinti kintančios akcijų rinkos pateikiamus duomenis, juos išanalizuoti ir pateikti varotojui patraukliu būdu. Sukurta programėlė bus prieinama vartotojams, kurių mobilieji įrenginiai naudoja „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 būtų galima įgyvendinti tokį uždavinį norėjau išanalizuoti kuo daugiau skirtingų programėlių ir suprasti kuo jos skiriasi bei ko trūksta, kad jos būtų geresnės ir atsižvelgiant į išanalizuotas programėles sukurti sistemą, kuri būtų paprasta naudoti, bet joje netrūktų 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, kilusi kuriant šią sistemą - 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 įterptų įkoduotų akcijų trumpinių sąraše ir 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 platesnius akcijos duomenis. Taip pat sukurtas 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 sudarysiu savo programėlės dizainą ir funkcionalumų įgyvendinimą.

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 grafikais (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 atvaizduojamas langas su dar didesniu informacijos kiekiu kartu su istoriniais kainų kitimo grafikais, naujienomis apie konkrečią akciją, jos detalią apžvalgą ir finansiniais duomenimis. Ši aplikacija pasižymi dideliu ir konkrečiu duomenų kiekiu apie kiekvieną konkrečią akciją, tačiau nėra labai paprasta naudoti ir sukelia informacijos pertekliaus jausmą. Šioje aplikacijoje galima susikurti savo portfelius ir pirkti akcijas. Tam, kad būtų galima pirkti akcijas, privaloma prisijungti prie sistemos naudojant savo realius duomenis.

1.3. Analizės išvados

1 lentelė. Panašių programėlių analizė

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

Lentelėje nr. tabl:isvadu-lentele pavaizduoti abiejų nagrinėtų aplikacijų privalumai ir trūkumai. Aplikacija *„My Stocks Portfolio and Widget“* neturėjo prisijungimo galimybės, 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. Vartotojui, kuris sistemą naudoja 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

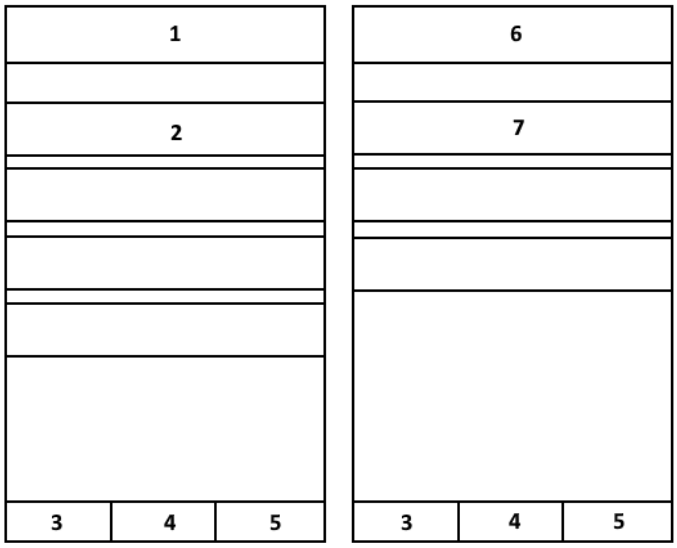
Funkciniai reikalavimai:

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:

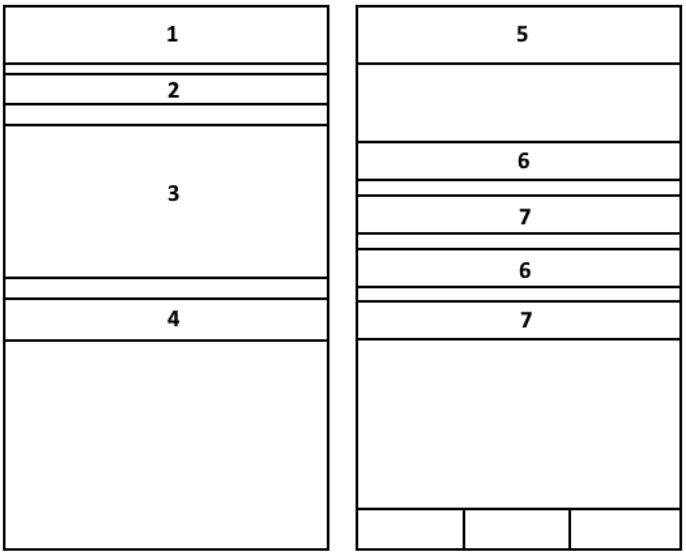
1. Aiškiai atvaizduojama informacija
2. Patogus ir intuityvus valdymas

2.2. Vartotojo sąsaja



1 pav. Akcijų kainų ir vartotojo aplanko atvaizdavimas

Paveikslėlyje nr. 1 yra pavaizduotas akcijų kainų ir vartotojo aplanko modelis, kuris yra suskirstytas į skirtingas dalis, kurios atstoja skirtingą funkcionalumą. Kairiajame paveikslėlyje yra atvaizduojamos akcijos, kurių adresai yra įkoduoti į programėlės atmintį ir naujų akcijų pridėjimas į sarašą yra neleidžiamas. 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 į iš anksto nustatytų akcijų sarašą. 4 - naršymo juostos liečiamas laukas skirtas pereiti į vartotojo akcijų aplanko langą. 5 - naršymo juostos liečiamas laukas skirtas pereiti į valiutų keitimo langą.



2 pav. Akcijų kainų ir vartotojo aplanko atvaizdavimas

Paveikslėlyje nr. 2 kairėje yra pavaizduotas detalusis akcijos langas, į kurį patenkama paspaudus ant akcijos, o dešinėje pavaizduotas valiutos keitimo langas. Kairiajame paveikslėlyje 1 - lango pavadinimas ir grįžimas į bendrą akcijos peržiūros langą arba vartotojo aplanką priklausomai nuo to iš kur buvo įvykdyta naršymo komanda. 2 - valdymo juosta, kuri nurodo kuriam laikotarpiui atvaizduoti akcijos istorinį grafiką. 3 - akcijos istorinis grafikas, kuris atvaizduoja pasirinkto dienų laikotarpio akcijos kainas. 4 - pasirinktos akcijos detalesnių duomenų sąrašas. Dešiniajame paveikslėlyje 5 - valiutos keitimo lango pavadinimas. 6 - valiutos įvesties ir išvesties laukai. 7 - valiutos pasirinkimo langai, kurie atidaro naują langą su valiutų pasirinkimais.

3. Praktinis programėlės įgyvendinimas

Norint įgyvendinti mobilią akcijų stebėjimo sistemą pirmiausia reikėjo išanalizuoti, kur galiu gauti nemokamus ir patikimus akcijų duomenis. Taip pat reikėjo išsiaiškinti tinkamiausias technologijas, kurios padėtų įgyvendinti sistemą.

3.1. Technologijų pasirinkimas

Programėlei realizuoti buvo pasirinkta „Facebook“ įmonės sukurta „JavaScript“ programavimo kalbos „React Native“ biblioteka. Programos paleidimui, kompiliavimui ir versijavimui buvo pasirinktas nepriklausomų programuotojų sukurtas nemokamas programėlių kūrimo paketas „Expo SDK“. Programėlė buvo testuojama nemokama „Android“ modeliavimo programa – „Genymotion“. Naršymas programėlės viduje buvo įgyvendintas pasitelkiant „React Navigation“ įrankį. Įvairių komponentų dalių ir būsenų išsaugojimą ir pernešimą programėlės viduje realizuoti pasirinktas „Redux“ įrankis.

3.1.1. „Expo SDK“ platforma

Kuriant šią programėlę, buvo skirtas itin didelis dėmesys siekiant kuo labiau supaprastinti programėlės kompiliavimo ir versijavimo procesą. Tam, kad nereikėtų nuolatos keisti kompiliavimo ir programėlės sudarymo procesų rankiniu būdu, buvo pasirinkta „Expo SDK“ kaip platformą savo programėlei. Ši platforma labai palengvina programėlės kompiliavimo ir kūrimo procesą automatiškai konfigūruodama kompiliavimą kiekvieną kartą pakeitus programinį kodą. Programuojant šioje platformoje 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 iš karto galima naudotis daugybe integruotų bibliotekų, tokių, kaip minėtasis „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 mobilioje sistemoje ir atnaujinti vos atnaujinus programinio kodo dalį. Taip pat ši platforma įtraukia visus tėvinius prietaiso funkcionalumus, kaip kameros, mikrofono, jutiklių valdymas.

3.1.2. „React Native ir Redux“

Funkcionalumui įgyvendinti buvo pasirinkta „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 šioje programėlėje yra būtinas bendravimas tarp skirtingų komponentų ir informacijos dalijimasis, buvo nuspręsta tai įgyvendinti naudojantis „Redux“ biblioteka. Ši biblioteka leidžia lengvai valdyti aplikacijos atmintyje laikomas savybes, komponentų parametrus ir juos perduoti kitiems komponentams, esantiems nebūtinai tame pačiame lygmenyje ar tame pačiame navigacijos lygyje.

3.1.3. „Genymotion“

Programėlės testavimui buvo pasirinkta nemokama modeliavimo įrankio „Genymotion“ versiją. Šis įrankis leidžia sukompiliuotą kodą atvaizduoti sumodeliuotame „Android“ įrenginyje kiek-

vieną kartą 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

Buvo nuspręsta apjungti kelis skirtingus duomenų šaltinius, nes visi suteikė skirtingo funkcionalumo. „Yahoo Stocks API“ buvo naudojama 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 buvo naudojamas „Quandl“ servisas, kuris grąžiną akcijų kainas pasirinktam laikotarpiui. Gautiems istoriniams kainų pakitimų grafikams atvaizduoti buvo naudojamasi 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. Žemiau yra pateikiamas mano sistemos struktūros pseudokodas:

- komponentai
  - namų/
  - portfelis/
- papildiniai/
- konstantos/
  - isoriniai-servisai/
- redux/
  - reducers
  - sore
- 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 parametrams, 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 kuris įkeliamas šioje sistemoje yra *komponentai/namų* komponentas. Tam, kad šis komponentas galėtų atvaizduoti norimą informaciją, buvo privaloma gauti pradinius naršymo maršrutų, programos būsenos ir išorinių servisų duomenis. Galiausiai namų komponente yra kviečiami išoriniai servisais iš anščiau nurodyto failo ir viskas yra saugoma programėlės 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 bus detaliau aptariami pagrindiniai funkcionalumai ir jų įgyvendinimas.

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 nr. 1). Pavyzdžio antrojoje eilutėje pavaizduotas išorinis paslaugų tiekėjas, kuriam pateikus norimos akcijos trumpinį yra grąžinamas duomenų apie akciją sąrašas. Šis sąrašas tuomet trečiojoje eilutėje yra interpretuojamas kaip objektas. Galiausiai ketvirtojoje eilutėje gautas objektas yra siunčiamas į aplikacijos atmintį (ang. state) ir ten yra išsaugomas vėlesniam panaudojimui.

1 išeities kodas. Funkcijos fetchSingleStocks() pseudokodas.

```
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 }
```

Įvykdžius funkciją *fetchSingleStocks(item)* programėlės atmintyje yra sukuriamas naujas akcijos objektas. Tam, kad šis objektas būtų atvaizduojamas, jis yra pridedamas į sarašą naudojantis „React Native“ komponentu *ListView*. Šis komponentas priima objektą ir perduoda šį objektą į funkciją *renderRow(item)*. Ši funkcija leidžia apdoroti vieną sarašo eilutę su gautais duomenimis ir grąžinti eilutę sąrašui. Tada yra kviečiama funkcija *fetchYahooSingle(pasirinktasSimbolis)*, kuri ieško pasirinkos akcijos *„Yahoo Finance“* duomenų bazėje (pavyzdys išeities kode nr. 2). Antrojoje eilutėje kviečiamas išorinis servisas, kuris radęs akciją ją grąžina kaip masyvą. Toliau šeštoje eilutėje vykdome 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 pridedame akciją į programėlės atmintį. Pats svarbiausias žingsnis yra septynioliktoje eilutėje, čia išsaugome rastą naują akciją į mobilaus prietaiso atmintį. Atlikus septynioliktą žingsnį ir perkrovus aplikaciją, vartotojas portfelio lange rastų išsaugotą akciją.

2 išeities kodas. Funkcijos fetchYahooSingle() pseudokodas.

```
fetchYahooSingle(pasirinktasSimbolis) {
  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=')

  const atsakasJson = atsakas.json();
  const portfelis =  this.state.portfolio;
  if (atsakasJson nera tuscias ir turi duomenis 'open/ask') {
          let rastasPortfelyje = [];
          for(sukame cikla per 'portfelis' ir ieskome ar nera jau tokio
              simbolio) {
           rastasPortefyle.push(atsakasJson);
          }
  }

  if(rastasPortfelyje === 0) {
          portfolio.push(atsakasJson)
          nustatomeGlobaliaBusena({ portfelis: atsakasJson });

          AsyncStorage.setItem('portfelis');
  }
  else {
          Alert('Tokia␣akcija␣jau␣yra␣sarase!');
  }
}
```

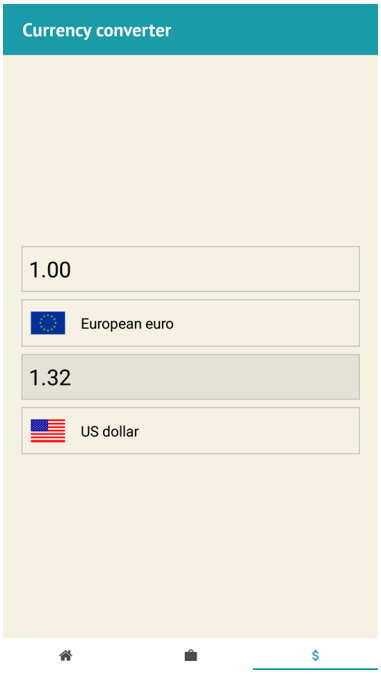
Norint atnaujinti vartotojo akcijų portfelio sarašą yra kviečiama funkcija *fetchYahooRefresh(portfelis)*. Ši funkcija kviečia išorinį servisą tiek kartų, kiek yra akcijų vartotojo portfelyje ir gauną atitinkamą kiekį masyvų. Kiekvienas masyvo elementas yra patikrinamas ar yra ne tuščias ir ar yra gauti atitinkami duomenys akcijai sėkmingai atvaizduoti. Jeigu duomenys yra gauti, senasis vartotojo portfelis yra ištrinamas iš programėlės atminties ir iš mobilaus prietaiso 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, grąžina 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 grafi-

kas gražintas sėkmingai šiame lange yra aprašomas grafiko aukštis ir plotis, kuris yra siunčiamas 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 buvo sukurtas grafiko datos imties keitimo funkcionalumas, 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 buvo būtina dalis, nes ne visi vartotojai naudoja vienodą valiutą akcijoms stebėti ir pirkti (lango pavyzdys paveikslėlyje nr. 3). Todėl buvo būtina užduotis sukurti būdą vartotojams išsiversti akcijų kainas į sau priimtiną valiutą. Šiame lange yra keturi laukai. Pirmasis laukas atlieka sumos įvedimo funkcionalumą (pavaizduota išeities kode nr. 3). Šioje funkcijoje gavus įvesties sumą, kursą ir išvesties kursą yra apskaičiuojama išvesties suma. Išvesties suma yra atvaizduojama *paveikslėlyje nr. 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.

3 išeities kodas. Funkcijos calculateRate() pseudokodas.

```
1 calculateRate(ivestis, ivestiesKursas, isvestiesKursas) {
2   const kursas = ivestiesKursas / isvestiesKursas;
3   return (ivestis * kursas);
4 }
```

Valiutų kursai yra apskaičiuojami naudojantis išoriniu *„Bank of Canada“* teikiamu, kasdien atnaujinamu duomenų sąrašu. Valiutų kursai yra gaunami iškviečiant funkciją *fetchRates* (pavaizduota išvestie kode nr. 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.

4 išeities kodas. Funkcijos fetchRates() pseudokodas.

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

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.

[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/Frameworkfortheeuroforeignexchangereferencerates.lt.pdf.

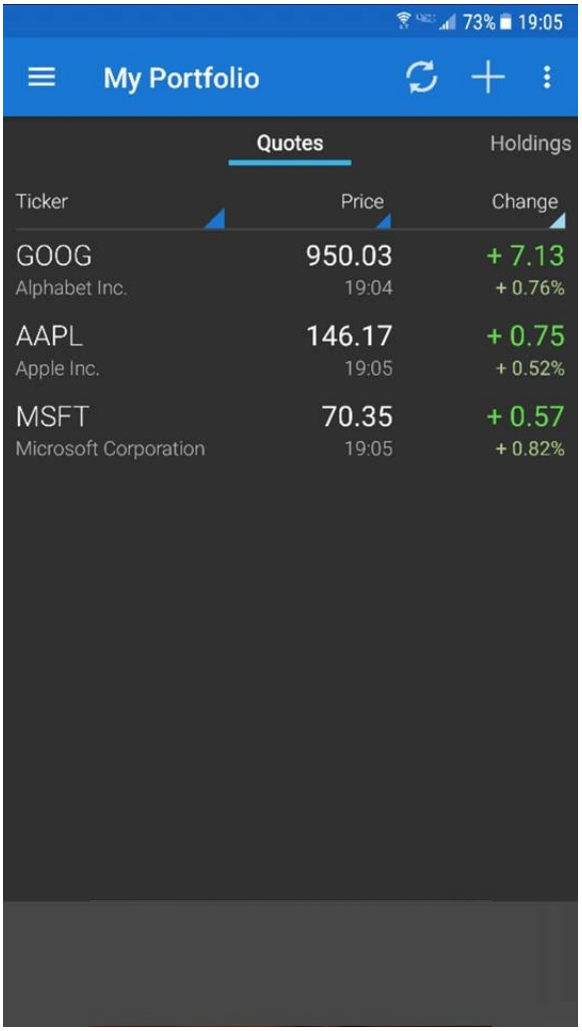
[4] Akcijų kainos ir jų pokyčiai.
https://finance.yahoo.com/.

[5] Istoriniai akcijų kainų pokyčiai.
https://www.quandl.com/product/WIKIP/WIKI/PRICES-Quandl-End-Of-Day-Stocks-Info.

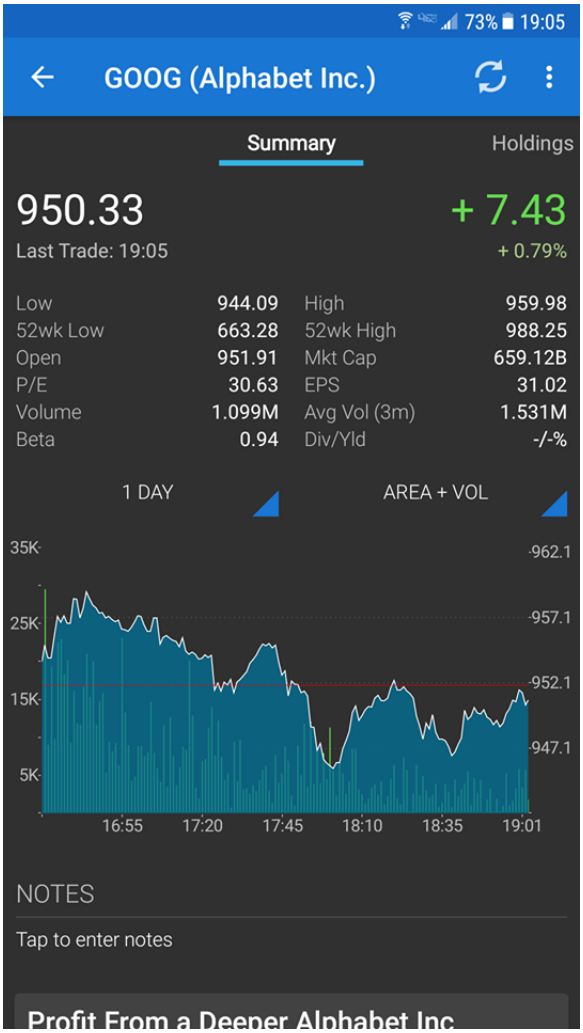
[6] Kasdieniai valiutų kursų pokyčiai.
http://www.bankofcanada.ca/rates/exchange/daily-exchange-rates/.

Priedai

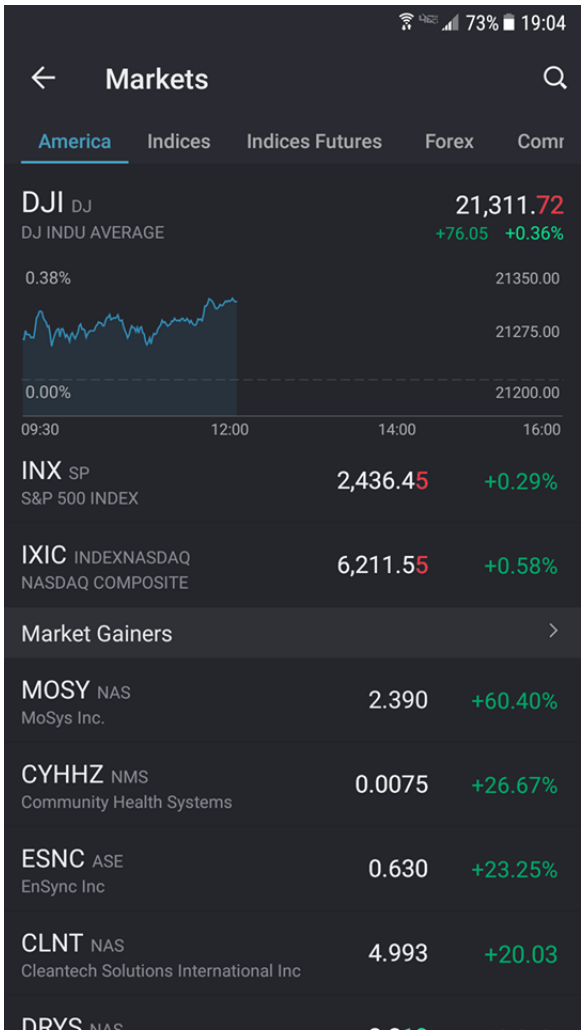
A. Programėlė „My Stocks Portfolio and Widget“



B. Programėlė „My Stocks Portfolio and Widget“

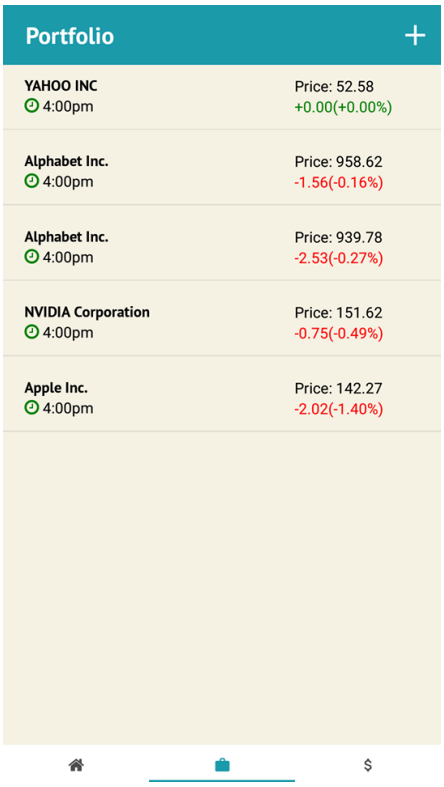


C. Antroji analizuota aplikacija

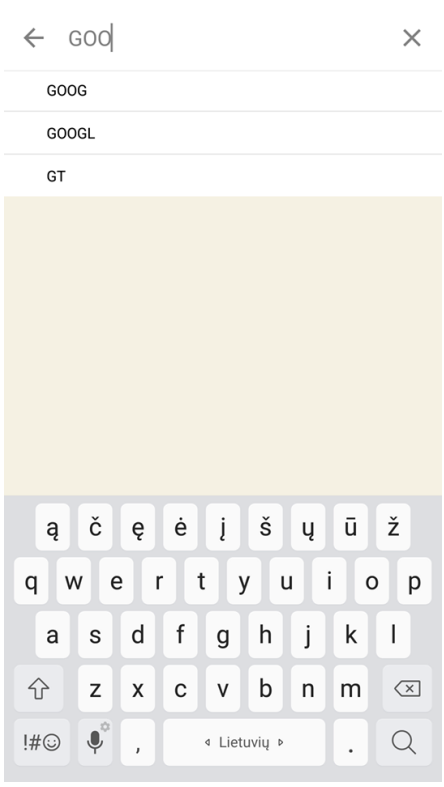


D. Antroji analizuota aplikacija

E. Vartotojo akcijų portfelio langas



F. Akcijos paieškos laukas portfelio lange



G. Pradinis programėlės akcijų langas

H. Akcijos duomenų detalaus atvaizdavimo langas