TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Deivid Drenkhan 135193IAPB

OSAKAALUDE PÕHISE PORTFELLIHALDUSE RAKENDUSE KAVANDAMINE JA ANALÜÜS

Bakalaureusetöö

Juhendaja: Enn Õunapuu
Dotsent

Tallinn 2018

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Deivid Drenkhan

21.05.2018

Annotatsioon

Investeerimine annab eraisikutele võimaluse firmade eduka käekäigu korral teenida täiendavat tulu, koguda raha eesmärkide täitmiseks või saavutada finantsvabadus. Finantsiliste eesmärkide täitmiseks on vaja investeeringuid aktiivselt jälgida ja hallata. Selleks on loodud mitmeid rakendusi, mis seda investorile lihtsamaks teeb.

Bakalaureusetöö põhieesmärk on luua terviklik portfellihalduse veebirakendus, mis võimaldab kasutajatel otsida erinevaid varaklasse, luua osakaaludel põhinevaid portfelle ja saada ülevaade oma investeeringute tootlikkusest.

Töö esimeses pooles püstitatakse eesmärgid, mida arendatav rakendus peab lahendama ja seejärel tutvustatakse, milliseid tehnoloogiaid ja tööriistu selle jaoks kasutatakse. Teises pooles on kirjeldatud veebirakenduse toimimist ja selle arendusprotsessi.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, 5 peatükki, 11 joonist, 2 tabelit.

Abstract

Percentage based portfolio management application design and analysis

Investing is a good opportunity to get part of company's good economic results to earn extra income, raise funds for financial goals or achieve financial freedom. For achieving these goals, one must actively keep track of the markets and manage investments. To make all this easier for novice investor, there are many portfolio management applications.

The purpose of this thesis is to develop an application, that enables its users to search all kinds of asset classes, create percentage based portfolios and get an overview of how one's investments are performing.

In the first part of the thesis an overview of the main problem is given and objectives are set that the application is going to be solving. Next, author describes all the technologies and tools that are used for developing this web application. In the second half of the thesis, author describes the process of developing this application and how does it work.

This web application consists of two parts: Node.js based back end and Angular front end. Both Node.js and Angular are JavaScript technologies that makes this application uniform in terms of programming language.

Thesis is considered successful and all the features were added to the application, allowing users to create and manage their investments.

The thesis is in Estonian and contains 33 pages of text, 5 chapters, 11 figures, 2 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendusliides
CSS	*Cascading Style Sheet*, kaskaadlaadistik
ETF	*Exchange-Traded Fund*, börsil kaubeldav fond
HTML	*HyperText Markup Language*, hüpertekst-märgistuskeel
HTTP	*HyperText Transfer Protocol*, hüperteksti edastusprotokoll
IDE	*Integrated Development Environment*, integreeritud programmeerimiskeskkond
JSON	*JavaScript Object Notation*, JavaScripti objekt notatsioon
NoSQL	*Non SQL*, mitte SQL ehk mitte relatsiooniline
PWA	*Progressive Web Apps*, progressiivsed veebirakendused
SEO	*Search Engine Optimization*, otsingumootoritele optimeerimine
URL	*Uniform Resource Locator*, universaalne ressursilokaator
WWW	*World Wide Web*, veeb

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Investeerimine on hea võimalus eraisikutel teenida kasu suuremate firmade tegevuselt. Näiliselt lihtne viis lisatasu teenida tõmbab ligi üha enam inimesi, kellel puuduvad vastavad teadmised finantsturgudel tegutsemiseks. Seetõttu kasutatakse ära kohalike pankade lihtsamat võimalust läbi nende investeerimisega tegeleda. Et saada paremat ülevaadet pankade abil realiseeritavate investeerimistehingute kohta, on vaja kindla funktsionaalsusega portfellihalduse tarkvara. Portfellihalduse veebirakendused annavad parema ülevaate investeeringutest, aga neis puudub üks oluline funktsionaalsus – portfelli osakute suurendamine eelmääratud osakaalude baasil.

1.1 Taust

Pikaajalistest investeeringutest on vaja saada ülevaade ning neid on vaja ka hallata (näiteks aktsiatehinguid ülesse märkida). Investeerimisteenust pakkuvad pangad võimaldavad teatud määral sellist funktsionaalsust oma veebirakendustes, samas jäävad need selgelt alla spetsiaalsetele rakendustele, mis on keerukamad ning kavandatud ja loodud konkreetselt investeerimist silmas pidades. Selliste rakenduste tuntumad esindajad on näiteks Google Finance, Yahoo Finance, Bloomberg jne.

Portfellihalduse veebirakendused annavad parema ülevaate investeeringutest, aga neis puudub üks oluline funktsionaalsus – portfelli osakute suurendamine eelmääratud osakaalude baasil.

LHV panga näitel on osakaaludega portfelli funktsionaalsus kasutusel „Kasvukonto” nime all. LHV Kasvukonto [1] võimaldab valida kuni kuus laiapõhjalist fondi ja määrata igale fondile osakaal selles portfellis. Kandes sellele kontole raha, jagatakse see summa osakaalude põhjal kasutaja valitud fondide vahel ja ostetakse fondi osakuid hetke turuhinnaga juurde.

1.2 Probleem

Nagu eelpool mainitud, on pankade veebirakendused investeeringutest ülevaate saamiseks ebamugavamad kui spetsiaalselt selleks loodud rakendused. Samas ei paku spetsiaalsed veebirakendused funktsionaalsust, mis jäljendaks teenust, mida kohalikud pangad pakutavat. Antud probleemid saab LHV Panga näitel välja tuua järgmiselt:

- fondide kohta täpsema info saamiseks suunatakse kasutaja välistele linkidele, tekitades kasutajas segadust, kuna info edastus on visuaalselt erinev;
- fondide ülddetailid ei eristu selgelt;
- luua saab ainult ühe osakaaludega portfelli („Kasvukonto”);
- alternatiivsetes veebirakendustes osakaaludega tüüpi portfellid üldse puuduvad.

Ülaltoodud punktides järeldub, et on võimalus portfellihaldus süsteem paremini kavandada ja selle bakalaureusetöö eesmärgiks on luua veebirakendus, mis need probleemid lahendaks.

1.3 Eesmärk

Antud bakalaureusetöö eesmärgiks on luua rakendus – Indic, mille abil saada ülevaade investeeringute tootlikusest ning luua võimalused osakaaludega ja tavalisete portfellide loomiseks ning haldamiseks. Veebirakendus võimaldab kasutajal luua antud veebikeskkonda isikliku konto, kuhu ta saab koostada endale meelepärased portfellid. Konto andmed salvestatakse andmebaasi, mida ei hoita kasutaja lokaalses seadmes. Uute portfellide loomisel saab valida portfelli tüübi ja lisada sinna erinevaid varaklasse (fondid, aktsiad, ETF). Rakendus on selgete parameetrite järgi hinnatav ja läbib Google Lighthouse olulistes kategooriates testid ehk saab hinnanguks vähemalt 75 punkti.

1.4 Ülevaade tööst

Bakalaureusetöö käigus tutvustatakse esmalt probleemi ja püstitatakse üldine eesmärk. Seejärel jagatakse rakenduse arendus etappideks ning tutvustatakse kasutatavaid tehnoloogiaid ja tööriistu.

Järgmiseks tehakse alternatiivsete rakenduste analüüs ja seatakse paika arendatava rakenduse nõutud funktsionaalsus, millele järgneb loodud veebirakenduse kirjeldus ja koodinäited. Autor annab ka ülevaate, kuidas toimub tehniline testimine ja optimeerimine.

Viimaks analüüsitakse valminud rakendust ja arenduse protsessi ning tehakse järeldused ja plaanitakse edasised arengud.

2 Metoodika

Esmalt on vaja kindlaks määrata, milliseid tehnoloogiad ja tööriistu veebirakenduse arendamiseks kasutatakse. Tagamaks eduka ja selge arendamise, on veebirakenduse arendusprotsess jagatud järgmisteks suurematеks etappideks:

- uurida ja analüüsida, millised lahendused püstitatud probleemidel juba turul olemas on;
- määrata minimaalne nõutud funktsionaalsus, mida antud bakalaureusetöö käigus arendatava veebirakendusega täidetakse;
- arendada rakendus, mis lahendab ülalmainitud probleemid ning täidab püstitatud eesmärgid ja nõuded;
- võrrelda loodud veebirakenduse tehnilisi hinnanguid alternatiivsete rakendustega;
- analüüsida esimeste testijate tagasisidet loodud veebirakendusele.

2.1 Kasutatud tehnoloogiad

Järgnevalt väljatoodud tehnoloogiad valiti rakenduse jaoks välja, pidades silmas seda, et uute tehnoloogiate õppimise peale oluliselt aega ei kuluks. Sellepärast baseerub enamik valitud tehnoloogiaid JavaScriptil, mida bakalaureusetöö autor TTÜs on õppinud ning omab selles head isikliku kogemust. Teine põhjus JavaScriptil baseeruvate tehnoloogiate kasutamiseks on see, et tulevikus on kergem rakendust hallata, kui suur osa selle realisatsioonist on kirjutatud ühes programmeerimiskeeles.

2.1.1 JavaScript

JavaScript [2] on objektorienteeritud programmeerimiskeel, mida peetakse üheks kolmest WWW alustehnoloogiaks [3]. Keel on dünaamiliste tüüpidega, mis tähendab seda, et programmi koodis ei pea ära defineerima muutujate liike, vaid need määratakse automaatselt vastavalt vajadusele.

Algselt oli JavaScript mõeldud kasutamiseks ainult kliendipoolsetes süsteemides, kus brauseritesse sisseehitatud spetsiaalsed käitusmootorid (*runtime engines*) suudavad selles keeles kirjutatud programme interpreteerida. Viimastel aastatel on keele populaarsus järjest kasvanud ning nüüd kasutatakse JavaScriptil põhinevaid tehnoloogiad ka serverites, mängumootorites jne. Ka selles bakalaureusetöös on JavaScript kasutusel nii kasutaja- kui ka serveripoolses liideses (*front end, back end*).

JavaScripti dünaamilised tüübid on suurte või lihtsalt keeruliste programmide korral takistavaks argumendiks. Arendajatel on raske meeles pidada kõiki muutujaid ja nende tüüpe. Veelgi enam, koodi lugedes ei ole arusaadav, mis tüüpi andmeid funktsioonid ja muutujad ootavad. Nii võivad tekkida küll oma loomuselt kerged süsteemi vead, millele aga on tihtipeale raske jälile jõuda. Selle probleemi lahenduseks on Microsoft loonud JavaScripti *superset*i - TypeScript.

TypeScript annab programmeerimiskeelele juurde staatilised tüübid [4], mis vähendab eelpoolmainitud vigade esinemist programmeerija töös. Kuna brauser ei oska interpreteerida TypeScripti faile, siis tuleb varasemalt need kompileerida JavaScripti failideks, mis on juba laialdaselt toetatud formaat.

2.1.2 Node.js

Nagu ülal kirjeldatud, on JavaScript kogumas populaarsust ka serveripoolsetes tehnoloogiates. Node.js [5] ongi multiplatvormne keskkond, kus JavaScripti programme käitada. Eelis teiste sarnaste tehnoloogiate ees (näiteks PHP) on see, et päringuid saab teha asünkroonselt ja seetõttu on Node.js väga skaleeruv ja kiire.

Node.js sisaldab endas mitmeid mooduleid, mille abil erinevaid ülesandeid lahendada. Vastavalt arendatava toote vajadustele, saab erinevate moodulite ja pakettidega muuta ning lisada rakendusele funktsioone, komponente ja palju muud. Moodulite lisamiseks ja haldamiseks on loodud „Node.js pakettide mänedžer” (*node package manager*) ehk lühidalt npm.

2.1.3 Npm

Npm [6] on suurim JavaScripti pakettide register, võimaldades neid arendajatel lihtsalt jagada ja kasutada. Hetkel sisaldab npm üle 600 000 erineva paketi. Npm teeb projektide ülesseadmise väga lihtsaks ja mugavaks.

Projektis on package.json fail, kus on kirjas kõik moodulid ja seaded, mida rakendusel vaja läheb. Kasutades npm-i läbi terminali, tuleb käivitada käsk „npm install", mille peale installitakse vajalikud paketid, et projekt uues töökeskkonnas käima saada. Tänu lihtsale JSON tüüpi struktuurile, on lihtne failist lugeda välja, millised paketid on vajalikud lihtsalt rakenduse toimimiseks ja millised rakenduse arendamiseks. Näiteks ei ole mõtet TypeScripti kompilaatorit host serverisse lisada, kuna see on vajalik ainult arenduseks.

2.1.4 Express

Express on Node.js raamistik, mis võimaldab määrata, mis funktsioon välja kutsutakse vastavalt HTTP meetodile ja URL-i struktuurile [7]. Nii saab ühe sõlmpunkti suunas teha mitmeid erinevaid päringuid, ilma et API struktuur ise liiga keeruliseks muutuks.

2.1.5 MongoDB

MongoDB on NoSQLi esindav andmebaasi tehnoloogia [8]. Andmed salvestatakse JSON formaadis, võimaldades lihtsalt hoida rakenduse objektide infot, mis on muutuva või ebaselge struktuuriga.

Antud bakalaureusetöö jaoks vajalik andmebaas on hostitud mLab veebikeskkonnas, mis pakub tasuta [9] MongoDB andmebaasi kuni 0,5 gigabaidiste struktuuride jaoks. See veebikeskkond leiti aine „Veebipõhiste rakenduste arhitektuur, disain ja tehnoloogia" raames ülesannete andmebaaside lahendusteks.

2.1.6 Angular

Angular [10] on väga võimekas ja modernne komponentide põhine JavaScripti raamistik, mis võimaldab ehitada terviklikke veebirakendusi. Angulari kasuks räägib kindlasti selle kiirus, mis tuleneb eelkõige just sellest, et peale esmast veebilehe laadimist ei ole rakenduses navigeerides vaja enam serveriga suhelda. Kogu vajalik info on kliendil juba olemas ja rakendus vahetab navigeerimisel vastavad osad JavaScripti abil välja. Seega ei ole vaja kulutada väärtusliku aega ja ressursse, et kuvada mitte muutuvaid rakenduse osasid.

Angular jaguneb oma ehituselt [11] 3 suuremaks osaks - komponendid, vaated ja teenused.

- Komponendid hoiavad endas ühe konkreetse komponendi loogikat ja vastavat vaadet.
- Vaated defineerivad komponendi HTML struktuuri.
- Teenuseid kasutatakse komponentides, et vahendada infot erinevate komponentide ja/või mudelite vahel.

Angulari struktuur on küllaltki keerukas ja uute komponentide lisamine võib olla suuremate projektide korral ebavajalikult keeruline. Selle jaoks on välja töötatud terminali liides Angular CLI [12], mis soodustab selge ja loogilise rakenduse struktuuri loomist. Näiteks teeb see väga kergeks uute projektide loomise, kasutades selleks käsklust „*ng new*", ning uute komponentide ja teenuste loomise, vastavalt „*ng generate component*" ja „*ng generate service*".

2.1.7 ng-bootstrap

Bootstrap [13] on „mobiilid enne" (*mobile-first*) lähenemisega CSS raamistik kasutajaliideste arenduseks. Selles raamistikus on kogumik valmis komponente, mida saab kergelt oma projektides kasutada. Toetatud on kõiki tuntumaid brausereid, mis tähendab seda, et komponentide käitumine on kõigis erinevates brauserites samasugune. Tänu sellele saab kasutajaliideseid oluliselt kiiremini arendada, kuna arendaja ei pea pidevalt kontrollima, kas tema loodud lahendused toimivad kõigis brauserites, vaid võib eeldada, et Bootstrapi komponendid töötavad igal pool.

Bootstrap on siiski ainult CSS raamistik ja dünaamiliste komponentide jaoks on vaja JavaScripti. Antud projektis on kasutusel Bootstrap v4.0.0, mis vajab jQuery raamistikku. Angular ise aga on juba JavaScript raamistik ja pole mõistlik kahte sama ülesannet täitvat raamistikku projektis kasutada. Siinkohal võetakse kasutusele ng-bootstrap [14], mis on Bootstrapi realisatsioon Angulari jaoks.

Kõik ng-bootstrapi komponendid on samuti testitud erinevates brauserites ja arendajad lubavad 100% kattuvust Bootstrap raamistikuga. Neil kahel raamistikul on siiski üks suur erinevus: animatsioonid. Nimelt ei ole ng-bootstrapi kaasatud ühtegi animatsiooni, kuna Angular käsitleb animatsioone niivõrd erinevalt, ei ole nende ületoomine Bootstrapist üks-ühele võimalik.

2.2 Kasutatud tööriistad

Rakenduse kvaliteet on suuresti sõltuv tööriistadest ja nende sobilikkusest projekti arendamiseks. Järgnevalt toob autor välja selle projekti jaoks kasutusel olevad tööriistad, mis võimaldasid rakenduse arendamise, versioonihalduse ja testimise.

2.2.1 IntelliJ IDEA

IntelliJ IDEA [15] on JetBrainsi loodud IDE, mis on mõeldud peamiselt Java programmeerimiseks. Selle bakalaureusetöö juures kasutusel olnud IntelliJ IDEA *Ultimate Editionit* saab täiendada ametlike pluginatega, mis lisavad programmile funktsionaalsust. Nii saab IntelliJ IDEA'ga arendada väga edukalt ka veebi- ja serverirakendusi.

IDE sisaldab endas väga palju tööriistu, mis teevad arendamise oluliselt mugavamaks ja kiiremaks. Näiteks sisseehitatud käsurida (*terminal*) aitab mugavalt npm käskluseid käivitada ja Angulari komponente genereerida. Lisaks on implementeeritud ka kõikide peamiste versioonihaldustarkvarade funktsionaalsus.

IntelliJ IDEA on tavakasutajatele tasuline, makstes 14.90€/kuus. Tänu õpilastele ja õpetajatele suunatud kampaaniale [16], on võimalik JetBrainsi valitud tooteid tasuta kasutada. Selleks on vaja täita taotlusevorm, kus õpilase või õpetaja staatuse kinnitamiseks on vaja sisestada ISIC kaardilt andmed.

2.2.2 Git

GIT on populaarne versioonihaldussüsteem [17], mis aitab mitmetel arendajatel koos sama projekti arendada. Versioonihaldus jälgib koodis tehtud muudatusi ja tagab ühise versiooni kõikidele osapooltele.

Selles bakalaureusetöös on kasutatud Bitbucketi veebiteenust versioonihalduse hostimiseks. See on väikestele meeskondadele (kuni viis liiget) või üksikkasutajatele tasuta ning erinevalt peamisest konkurendist, GitHubist, võimaldab Bitbucketi tasuta pakett luua privaatseid repositooriumeid [18]. Kuna rakenduse kaugema tuleviku osas pole veel kindlat plaani, on mõistlik kood privaatsena hoida.

2.2.3 Lighthouse

Lighthouse [19] on tarkvarahiiglase Google'i loodud audiitortööriist, mis on loodud testimaks veebilehtede jõudlust ja kiirust. Lisaks jõudluse testimisele võimaldab tööriist testida ka PWA valmidust, juurdepääsetavust, parimate tavade järgimist ja SEO-d. Lighthouse annab igale testitud kategooriale hinnangu saja punkti skaalal, kusjuures „hea" tulemus hakkab 75-st. Lisaks hinnangule teeb Lighthouse ka asjalikke soovitusi, kuidas igas kategoorias parem tulemus saavutada.

Veebilehte testides simuleeritakse telefoni parameetreid, et saada ülevaade, kuidas veebileht kaasaskantavatel seadmetel töötab. Kuna pea pool kogu interneti liiklusest toimub just sellistel mobiilsetel seadmetel [20], on oluline tagada mugav toimimine selliste parameetrite korral.

3 Rakenduse arendus

Veebirakenduse arendamine on protsess, milles eesmärkide saavutamine on jagatud väiksemateks ülesanneteks. Selline arendusviis aitab hoida fookust eesmärkide saavutamisel ja üldiselt lihtsustab ning kiirendab arendust. Arenduse nõutud funktsionaalsuse määramiseks on vaja põhjalikult planeerida eelseisev töö, mis on keerulisem kui esmapilgul tunduda võib. Nimelt tekib arenduses mitmeid ettearvamatuid olukordi, millele tuleb jooksvalt lahendusi leida.

Ülesannete (*tasks*) haldamiseks on mitmeid rakendusi, mis on mõeldud suuremates meeskondades kasutamiseks. Selle bakalaureusetöö käigus aga piisab lihtsast ülesannete nimekirjast, kuna arendajaks on ainult üks inimene.

Enne nõutud funktsionaalsuse määramist uuritakse juba turul olevate alternatiivsete rakenduste omadusi, et saada ülevaade, kuidas portfellihaldus üldiselt erinevate firmade ja arendusmeeskondade poolt lahendatud on.

3.1 Sarnaste rakenduste analüüs

Finantsturgudel on majanduses väga suur roll ning investeeringute haldamise platvorme ja rakendusi on turul sellepärast palju. Portfelli halduse funktsionaalsust pakuvad mitmed ettevõtted ja üldiselt võib jagada need kaheks: tasulised rakendused, mis pakuvad väga palju võimalusi ja infot, ning tasuta rakendused. Järgnevalt keskendutakse ainult tasuta programmidele, et olla kättesaadav võimalikult suurele kasutajaskonnale.

Analüüsiks valitakse välja tuntumad firmad, kelle rakenduste juures uuritakse järgnevaid omadusi:

- üldine disain ja rakenduses navigeerimise keerukus;
- varaklasside otsimine;
- portfelli loomine ja selle ülevaade.

3.1.1 Google Finance

Antud bakalaureusetöö kirjutamise ajal muutis Google Finance rakenduse disaini ja funktsionaalsust, kaotades ära portfellide loomise võimaluse. Järgnev analüüs on tehtud enne muudatuste sisse viimist.

Rakenduse üldine välimus on lakooniline ja see teeb lehel orienteerumise keeruliseks. Avalehel on esikohal tänased olulisemad uudised, millele järgneb trendide nimekiri. Sellel on neli erinevat võimalust, mille alusel suuremad varaklasside muutujad välja kuvatakse. Paremal ääres on toodud ülevaade kasutaja poolt loodud portfelli tootlikusest. Lisaks on eraldi vaade uudistele, portfellidele, *stock screener*ile ehk varaklasside otsingu vaatele ja viimaks Google Domestic Trends. Viimane mõõdab Google otsingumootoris tehtud päringute arvu ajas, liigitades need erinevate majandusharude põhjal omavahel kokku.

Google Finance on kindlasti mõeldud kogemustega kauplejale, kes tahab end oluliste uudistega kursis hoida. Üllatav on, et disainile pole selle lehe juures erilist rõhku pandud.

3.1.2 Yahoo Finance

Esmamulje rakenduse disainist on modernne ja infotihe. Rakenduse avaleht paneb rõhku viimastele uudistele, mis võtavad enda alla suurema osa lehest. Uudistest paremale tulpa jäävad kasutaja loodud jälgimisnimekiri, portfellid, viimati vaadatud varaklassid, olulisemate valuutapaaride suhted, indeksfondide hinnad jne.

Aktsiate otsimine on tervel lehel püsival kohal - see on alati nähtav päises. Alustades otsingu välja kirjutamist, hakkab programm välja tooma otsingule seotud vasteid.

Portfelli loomise vaatesse jõudmine on aga keerukam, kuna vastav nupp on teistest eristumatu ja ei hakka koheselt silma. Portfelli loomine on ise tabeli põhine, kus uue varaklassi lisamisel täidetakse tabeli veerud koheselt rohke tehnilise infoga. Selle alla kuvatakse uudised, mis on seotud valitud varaklassidega. Portfelli ülevaade on täpselt seesama tabel, mis portfelli loomise ajal kuvati.

3.1.3 Bloomberg

Bloombergi veebirakenduse disain on samuti modernne, kuid eripäraks on selle jõuliselt lihtne disain. Erinevate sektsioonide pealkirjad on suured ja silmapaistvad, aga info tihedus ei ole nii kokkupressitud nagu Yahoo Finance'il. Kõik lehe navigeerimisvalikud on peidetud päises asuvasse rippmenüüsse, mis teeb üldise navigeerimise veidi keeruliseks.

Otsing, mille kaudu varaklasse leida, on samuti päises. Tulemuste seas kuvatakse ka otsingule vastavaid uudiseid, mis võib algajale investorile asja segaseks teha.

Portfelli loomine on taaskord keeruline ja vastavasse vaatesse jõudmine nõuab esimesel korral päris palju otsimist. Portfelli vaade on jagatud kolmeks suuremaks osaks - ülevaade portfellist, portfellis olevate aktsiate infotabel ja portfelliga seotud uudised. Märkimisväärne on Bloombergi portfelli vaate juures ülevaate sektsioon, kus on interaktiivne graafik. Vastavalt valikule näitab graafik portfelli jaotust varaklassi, tööstuse või geograafilise asukoha järgi.

3.1.4 Sarnaste rakenduste analüüsi kokkuvõte

Üldine disain on analüüsitud rakendustel modernne, aga liialt infotihe, et alustavale investorile selget ülevaadet anda. Otsingu funktsionaalsus on kõigil mugav ja kasutajasõbralik, mida oligi oodata, sest automaatselt otsinguid lõpetavad süsteemid (*autocomplete*) on ammu standardiks saanud.

Kõikides rakendustes on portfelli üldinfo toodud järgmiste parameetritega:

- portfelli hetke turuväärtus;
- portfelli väärtuse muutus tänase päeva jooksul;
- portfelli kogu väärtuse muutus.

Selle bakalaureusetöö käigus loodava rakenduse navigatsioon peaks olema analüüsitud rakendustest lihtsam ja olema pidevalt nähtaval. Samuti peaks portfelli loomine olema selgesti ligipääsetav. Implementeeritakse ka andmete esitlus tabeli kujul, tähelepanu tuleb aga pöörata sellele, et liiga palju andmeid kokku ei pressita.

Tabelis 1 toodud portfelli tüüpide võrdlusest selgub, et osakaaludel põhinevat portfelli ei saa üheski võrreldud rakenduses luua.

Tabel 1. Alternatiivsete rakenduste portfelli tüüpide võrdlus.

	Google Finance	**Yahoo Finance**	**Bloomberg**
Tavaliste portfellide loomine	Ei	Jah	Jah
Osakaaludega portfellide loomine	Ei	Ei	Ei

3.2 Nõutud funktsionaalsus

Tehtud alternatiivsete rakenduste analüüsile ja töö alguses seatud eesmärkidele põhinedes seatakse paika nõutud funktsionaalsus, mille alusel hakatakse rakendust arendama.

- Rakenduses peab saama luua kasutajakonto, millega seotud info salvestatakse andmebaasi.
- Rakendus peab võimaldama varaklasside otsimist ja implementeerib otsingut lõpetava süsteemi.
- Kasutaja saab luua kahte tüüpi portfelle:
    - tavaline portfell;
    - osakaaludega portfell.
- Mugava kasutajakogemuse nimel peab rakendus olema piisavalt kiire.

3.3 Serveripoolne liides

Serverisse on loodud Node.js'il põhinev API, mis tegeleb kõigi andmebaasi ja varaklasside päringutega. Kõik päringud tehakse ühe lõppsõlme (*endpoint*) suunas, kus Express rakendus need vastu võtab. Vastavalt HTTP meetodile ja URL-i struktuurile käivitub vastav API funktsioon (Joonis 1).

```
router.get('/user/:username', function (req, res) {
    getUser( … );
});
router.put('/user', function (req, res) {
    updateUserData( … );
});
```

Joonis 1. Express funktsioonid kasutaja andmete pärimiseks ja uuendamiseks.

Esimene funktsioon, getUser(), kutsutakse välja, kui päringu meetodiks on „GET" ning URL struktuuris on parameeter „:username". Andmebaasis kasutaja andmete uuendamiseks mõeldud päring tehakse sama URL raja suunas, aga kuna HTTP meetod on teine, siis käivitub funktsioon updateUserData().

3.3.1 Varaklasside andmed

Kogu varaklassi põhine info tuleb yahoo-finance [21] npm paketist. Pakett võimaldab kasutada lihtsat API-t varaklasside kohta päringute tegemiseks. Paketis saab kasutada funktsioone yahooFinance.quote() ja yahooFinance.historical(). Esimene võtab sisendargumendiks varaklassi sümboli (nt AAPL – Apple Inc) ja tagastab objekti hetkeinfoga, milles on toodud turuhind, kauplemismaht, üldine kirjeldus jne. Teine funktsioon võtab sisendiks sümboli, huvipakkuva ajavahemiku ja ajavahemiku sageduse ning tagastab objekti varaklassi ajalooliste hindadega.

3.3.2 Turvalisus

Arvestades seda, et praegusel kujul rakendust suuremale avalikkusele avaldada ei plaanita, on turvalisus tagatud ainult kasutajanime ja parooli abil. Lisaks tuleb silmas pidada, et küberturvalisus on modernsete veebilahenduste puhul suur ja keerukas teema, mis ei ole antud ülesande eesmärgiks.

Kasutaja andmed päritakse andmebaasiga ühenduses olevast serveris, kus MongoDB funktsioon otsib välja kasutajanimele vastava andmebaasi dokumendi (Joonis 2). Kui kasutajanimele vastav andmebaasi dokument on leitud, tagastatakse see Angulari rakendusele.

```
db.user_data.findOne({username: req.params.username}, function (err, user) {
    if (err) {
        res.send(err);
    }
    if (user) {
        res.json(user);
    } else {
        res.json('-1');
        console.log('User not found.');
    }
    res.end();
});
```

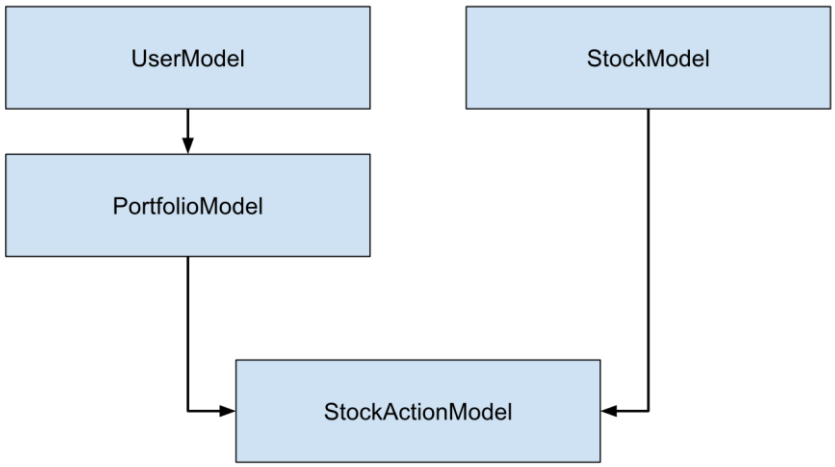
Joonis 2. Kasutaja andmete pärimine findOne() funktsiooni abil.

3.4 Andmemudelid

Andmemudelid on kasutusel, et siduda kokku omavahel relatsioonis olevad andmed. Kõik mudelid on oma olemuselt siiski JavaScript objektid, mis teeb nendes talletatud info tänu punkt notatsioonile (*dot notation*) [22] mugavalt kasutatavaks.

Rakenduses on kasutusel neli omavahel seotud andmemudelit (Joonis 3):

- kasutaja mudel (UserModel);
- portfelli mudel (PortfolioModel);
- tehingu mudel (Stock-ActionModel);
- varaklassi mudel (StockModel).



Joonis 3. Andmemudelid ja nende seosed

Andmeid hoitakse andmebaasis NoSQL formaadis ehk kogu info on ühe objektina salvestatud. Edukal sisselogimisel tõmmatakse vastava kasutaja info rakendusse ja täidetakse kasutaja mudel. Selle täitmisel luuakse omakorda portfelli mudelid, millest tehakse üks massiiv ning see seotakse otse kasutaja mudeliga.

Lisas 1 toodud koodinäites on näha, millise struktuuriga andmemudelid on üles ehitatud. Mudeli muutujad on määratud privaatseks ning nende lugemiseks ja ülekirjutamiseks on vastavad *get* ja *set* funktsioonid. Mudelite esialgsel loomisel käivitatakse funktsioon fillData(), mis täidab sisendobjekti põhjal selle mudeli muutujad.

Igas portfelli mudelis on massiiv varaklassidest. Nendes omakorda on massiiv tehingute ajaloost ehk tehingu mudelitest. Erinevalt portfelli mudelist ei looda tehingu mudeleid enne, kui rakendus hakkab arvutama portfelli tootlust ja teisi portfelli spetsiifilisi andmeid.

Varaklassi mudel luuakse vastavalt vajadusele ja täidetakse infoga yahoo-finance paketi abil. Varaklassi mudelit kasutatakse näiteks ühe konkreetse varaklassi vaate juures. Samas on varaklassi mudeleid vaja ka portfelli vaate juures, kus on lisaks portfelli andmetele vaja välja tuua konkreetsemalt varaklassi infot (näiteks muutus tänase päeva jooksul).

3.5 Teenused

Teenused (*services*) on osa Angulari arhitektuurist ning oma olemuselt on need kontrollerid, mis vahendavad info liikumist andmebaasi ja vaadete vahel. Teenused võivad aga pakkuda jagatud funktsionaalsust, mida rakenduse mitmes kohas kasutatakse. Rakenduses on kasutusel järgmised teenused:

- kasutaja teenus (UserService);
- portfelli teenus (PortfolioService);
- matemaatika teenus (MathService);
- varaklassi teenus (StockService).

Iga teenus hoiab endas veebirakenduses jagatud funktsioone, mis on andmemudeleid silmas pidades kokku grupeeritud. Lisas 2 on toodud koodinäide portfelli teenusest, mis sisaldab kahte funktsiooni, mille abil arvutatakse portfelli väärtuse muutus.

3.6 Vaated

Komponendid on samuti Angulari arhitektuuri osad, mis koosnevad HTML struktuurfailist ja JavaScript failist, mis konkreetse vaate initsialiseerib ja selle funktsioone endas hoiab.

Arendatavas veebirakenduses on kasutusel mitmeid komponente, mis jagunevad üldistatult kolmeks suuremaks osaks:

- otsing;
- varaklassi detailvaade;
- portfelli vaated.

3.6.1 Sisselogimine ja registreerumine

Esmasel rakenduse avamisel näidatakse kasutajale sisselogimise ja registreerimise valikuid. Kui kasutajal on konto loodud, saab ta kasutajanime ja parooliga sisse logida. Sisselogimise peale tõmmatakse andmebaasist alla kasutaja andmed, mille alusel ülejäänud kasutajast sõltuv info kuvatakse.

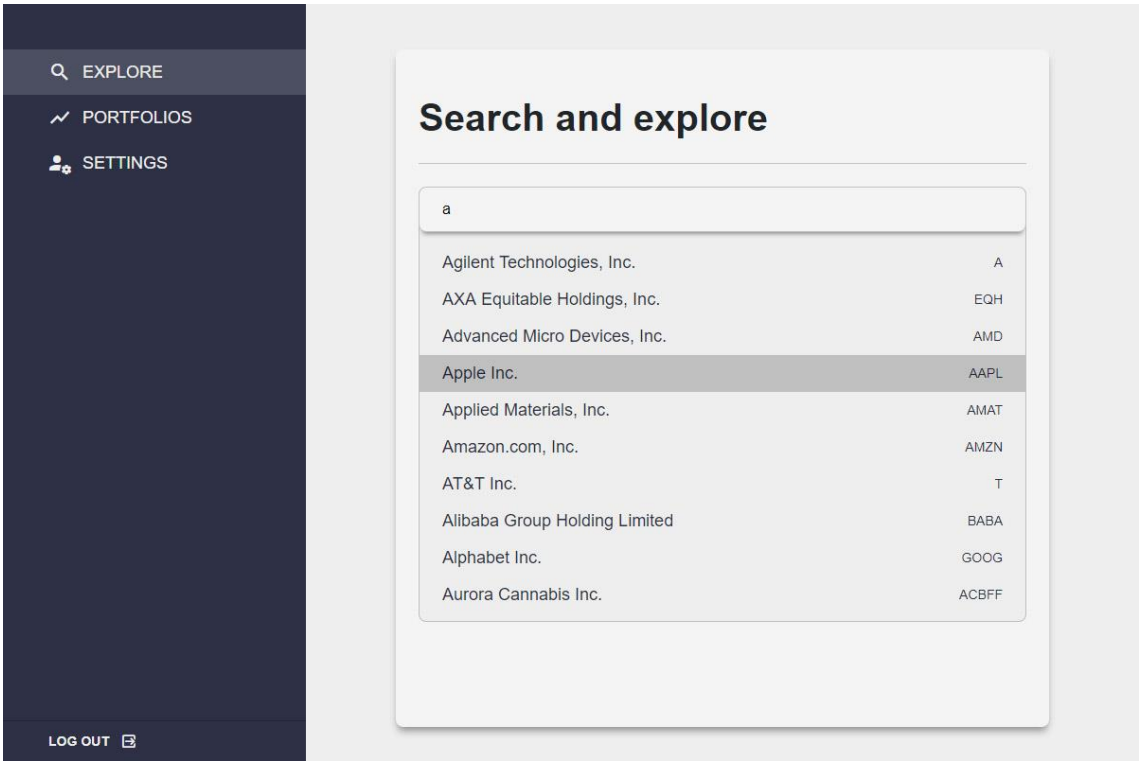
Juhul kui kasutajal ei ole kontot, saab ta selle samas vaates luua, mis kutsub kasutaja teenuses asuva funktsiooni registerUser() (Joonis 4). Peale registreerimist saab uus kasutaja vastavate andmetega sisse logida.

```
constructor(
    public _user: UserService
) {}
initRegister() {
    this._user.registerUser( this.new_username, this.new_password );
}
```

Joonis 4. Uue kasutaja registreerimine kasutaja teenuse abil.

3.6.2 Otsing

Otsingu vaade (Joonis 5) on erinevate varaklasside otsimiseks ja nendeni navigeerimiseks. Kui kasutaja sisestab otsingulahtrisse varaklassi nime või ainult nimeosa (otsingus on võimalik kasutada ka varaklassi sümbolit), kuvatakse sellele vastavad tulemused otsingulahtri all. Klikkides ühel tulemusel, suunatakse kasutaja selle



Joonis 5. Kuvatõmmis otsingu vaatest.

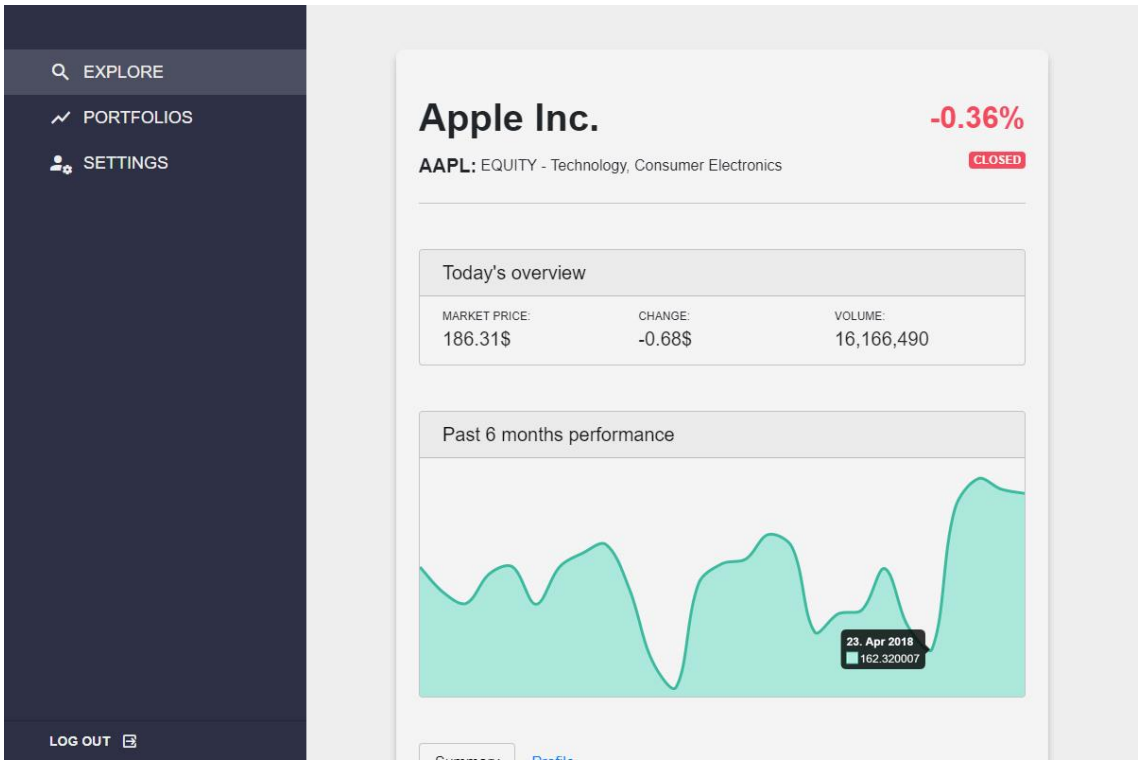
varaklassi vaatesse. Otsingu funktsionaalsus on täidetud, kasutades yahoo-finance *autocomplete* API-t.

Kuna samanimelisi varaklasse võib olla rohkem kui üks, on otsingu tulemused toodud nimekirjas varaklassi nime ja sümboliga. See aitab kasutajal leida õige varaklassi. Näiteks otsides „tallinna vesi" on tulemuste seas kaks AS Tallinna Vesi nimelist tulemust: üks Tallinna ja teine Frankfurdi börsi oma.

3.6.3 Varaklassi detailvaade

Detailvaade (Joonis 6) on mõeldud ülevaate saamiseks varaklassist erinevatel ajaperioodidel. Selle vaate saamiseks tehakse kaks päringut yahoo-finance API abil:

- üldinfo päring;
- ajalooliste hindade päring.



Joonis 6. Kuvatõmmis varaklassi vaatest.

Lehe ülaservas kuvatakse varaklassi põhiinfo: nimi, sümbol, varaklassi liik, tegevusala sektor ja protsentuaalne muutus viimase kauplemispäeva jooksul. Järgneb detailsema

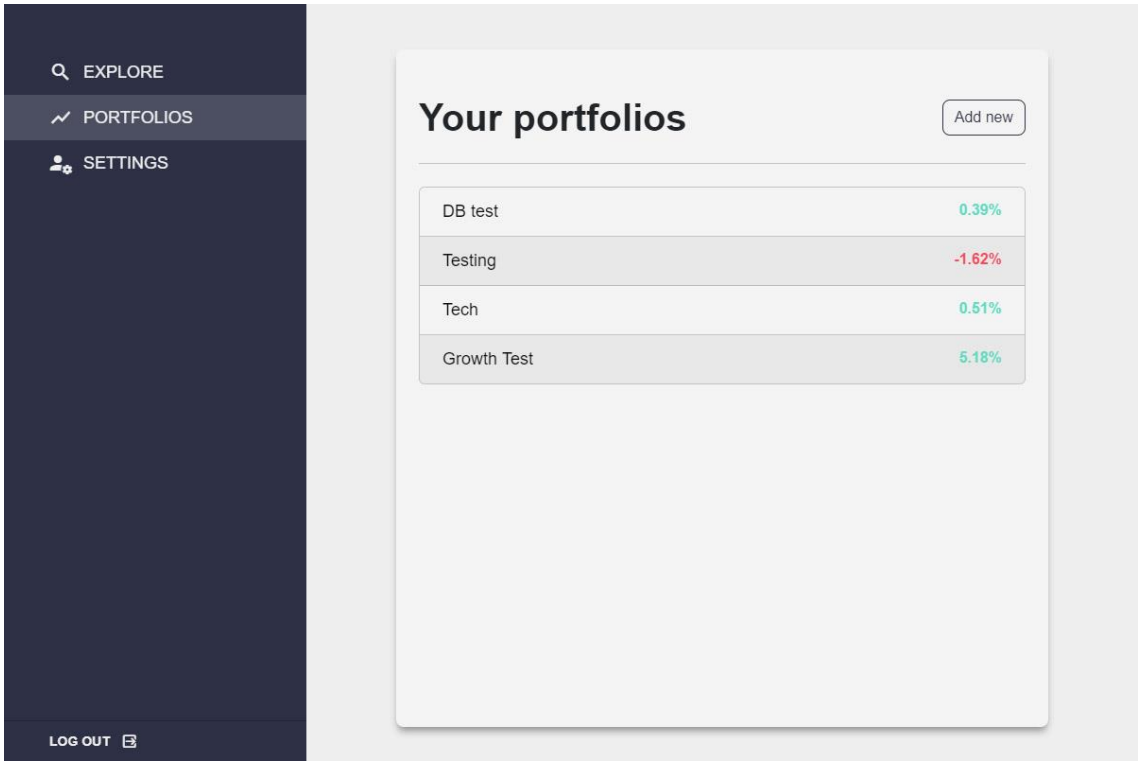
infoga sektsioon viimase kauplemispäeva tähtsamate detailide kohta: turu hind, muutus varaklassi valuutas ja kauplemise maht.

Viimase kuue kuu jõudluse graafiku sektsioon aitab kasutajal saada parema ülevaate, kuidas varaklassi hind liikunud on. Rakenduse kasutaja on tõenäoliselt sellisest näitajast huvitatud, sest pikaajalisel investeerimisel on tähtis mõista üleüldist hinna liikumise trendi.

Detailide sektsioonil on sakkidel (*tabs*) välja toodud täpsemad andmed tänase päeva kohta ja ka üldisem info varaklassi enda kohta. Vastavad detailid on jagatud eraldi kategooriateks, mis teeb infost ülevaate saamise kergemaks ja kiiremaks.

3.6.4 Portfellide nimekiri

Kasutaja portfellid on toodud nimekirjana portfellide põhivaates (Joonis 7). Iga portfelli kohta on toodud protsentuaalne muutus investeeringute kulutuste summa ja hetke turuväärtuse suhtes. Vastavad numbrid on saadud, liites kokku portfelli iga varaklassi tehingute kulu ning korrutades varaklasside kogused turu hetkeväärtusega. Selles vaates on toodud ka navigatsioon uue portfelli loomise vaatesse.

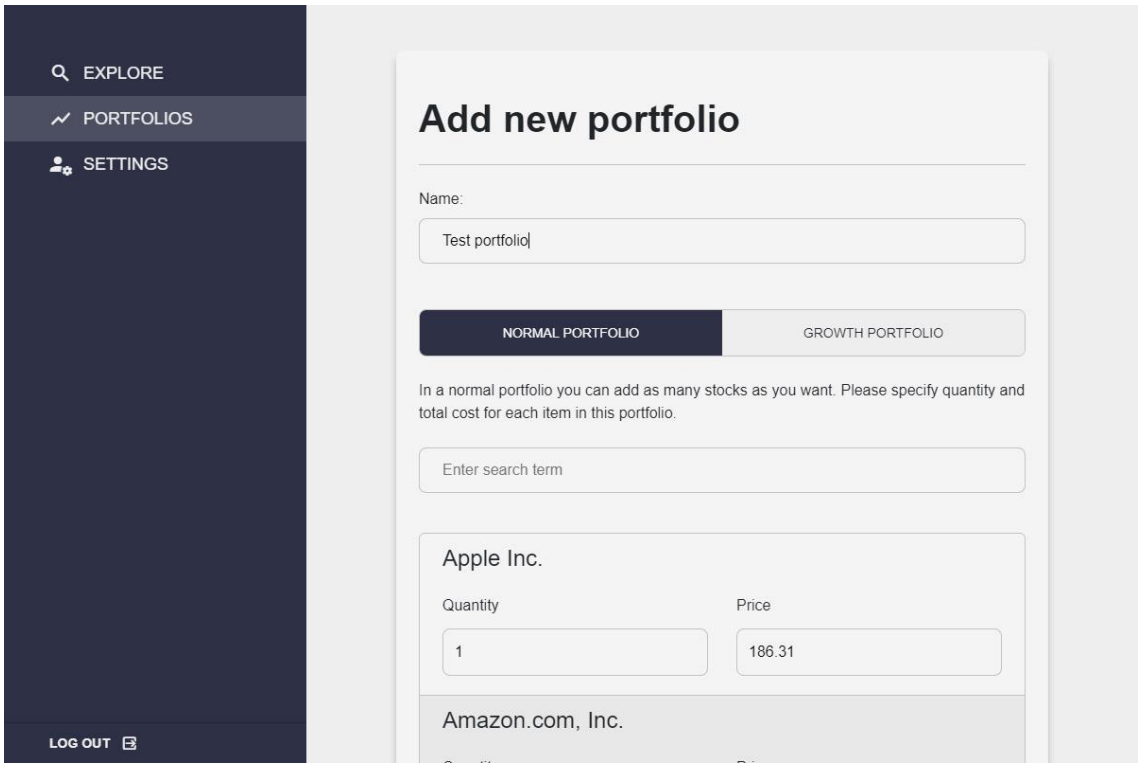


Joonis 7. Kuvatõmmis portfellide nimekirja vaatest.

3.6.5 Portfelli loomine

Portfelli loomise vaates (Joonis 8) saab kasutaja määrata portfellile nime ja tüübi. Tüübi valikus on kaks võimalust:

- tavaline portfell;
- osakaaludega portfell.



Joonis 8. Kuvatõmmis uue portfelli loomise vaatest.

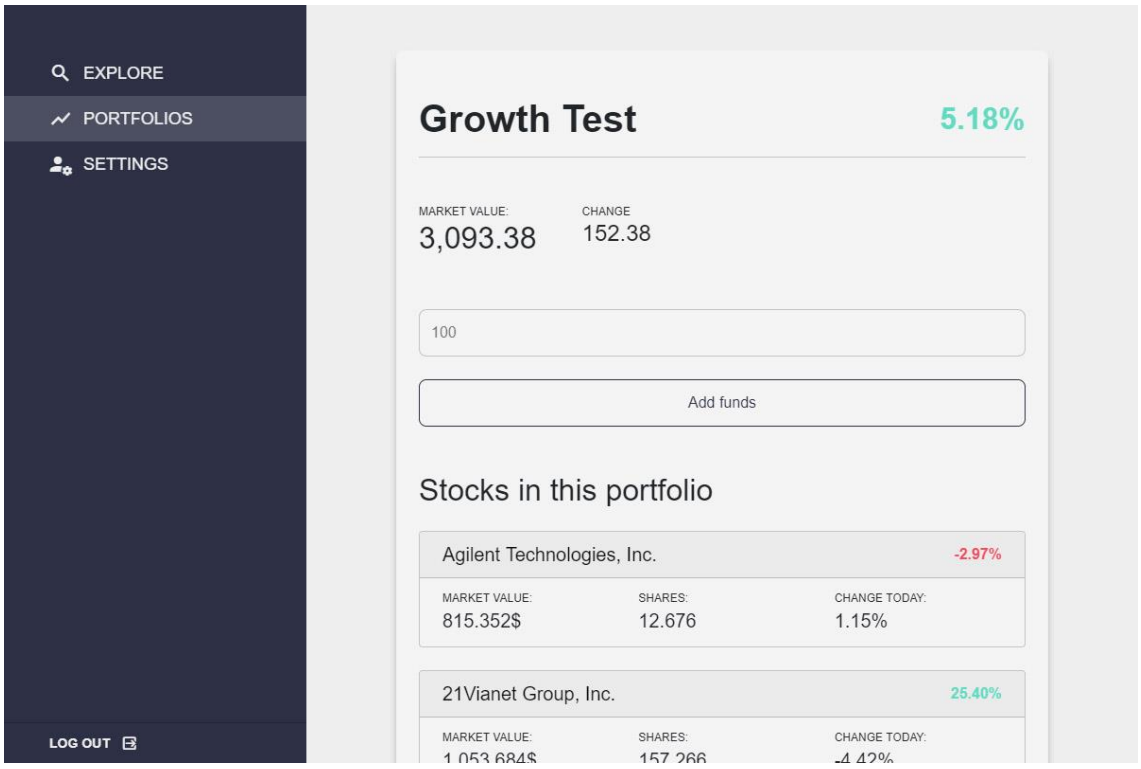
Et kasutajal oleks nende erinevus selgem, on kummagi kohta toodud ka kirjeldus, mis selgitab nende toimimise mehhanismi.

Portfelli tüübi valiku all on otsingulahter, mis toimib sarnaselt otsinguvaatega. Ainus erinevus on selles, et klikkides otsingu tulemusele, lisatakse vastav varaklass otsingu alla nimekirja, millest moodustub portfell. Vastavalt portfelli tüübile on nimekirjas erinevad lahtrid. Tavalise portfelli korral on kaks lahtrit: kogus ja hind. Mõlemal on vaikimisi määratud väärtus, koguseks “1” ja hinnaks turu hetkeväärtus. Osakaaludega portfellil on üks lahter, mille väärtusega määratakse varaklassi osakaal portfelli suhtes. Väärtuseks võib olla ükskõik milline positiivne arv, mille põhjal seatakse vastavale

varaklassile protsentuaalne osakaal. Näiteks kui portfellis on kaks varaklassi, esimesel on osakaaluks 1 ja teisel 4, siis protsentuaalsed väärtused oleksid vastavalt 20% ja 80%.

3.6.6 Portfelli detailvaade

Portfelli detailvaates (Joonis 9) on sarnaselt varaklassi detailvaatele oluline info toodud lehe ülaosas. Märgitud on portfelli turuväärtus, väärtuse muutus valuutas ja protsentides. Kui tegemist on osakaaludega portfelliga, siis on põhiinfo all lahter portfelli raha lisamiseks. Lahtrisse sisestatud arv jagatakse vastavalt osakaaludele varaklasside vahel ära ja suurendatakse nende koguseid turuhinnast sõltuvalt. Järgmine sektsioon on mõlemal tüübil sarnane - nimekiri portfellis olevatest varaklassidest. Tavalise portfelli korral on aga igal varaklassil eraldi võimalus osakuid lisada. Osakute lisamine käib sarnaselt nagu uue portfelli loomisel.



Joonis 9. Kuvatõmmis portfelli detailvaatest.

3.6.7 Kasutajate andmete vaade

Selles vaates on välja toodud kõik kasutaja infot puudutavad detailid – ees-, pere- ja kasutajanimi, e-maili aadress ning konto parool. Kõik andmed on lihtsasti muudetavad Angulari *two-way-binding* abil. Vaates toodud sisendilahtrid on seotud otseselt kasutaja

andmetega ehk muudatuste tegemisel uuendatakse automaatselt kasutaja mudelit. „*Update*” nuppu vajutades salvestatakse selle kasutaja uuendatud mudel andmebaasi (Joonis 10).

```
updateUser( _user: UserModel = this.user ) {
        const headers = new Headers( { 'Content-Type': 'application/json' } );
        const options = new RequestOptions( { headers: headers } );
        this.data_state.next( 'loading' );

        this._http.post(   'http://localhost:3000/api/user',   JSON.stringify( _user ), options )
            .map( res => res.json() )
            .subscribe( res => {
                if ( res.hasOwnProperty( '_id' ) ) {
                    this.data_state.next( 'good' );
                    return res;
                } else {
                    return -1;
                }
            });
}
```

Joonis 10. Kasutaja andmete uuendamise funktsioon.

3.7 Optimeerimine

Tagamaks valminud rakenduse tehniline võimekus, tuleb seda testida objektiivselt. Antud ülesande jaoks on Lighthouse ideaalne lahendus. Lisaks kiirusele saab testida ka rakenduse struktuuri kvaliteeti. Testimisel ei keskenduta PWA tulemusele, kuna see ei ole hetkel oluline, aga selle võib tulevikus implementeerida.

Lighthouse’i kasutatakse testimiseks käsurealt, see võimaldab testide üle rohkem kontrolli saavutada ja testide parameetreid muuta. Testide jaoks muudetakse protsessori aeglustamise parameetrit – tavaline neljakordne aeglustamine lülitati välja. Kuna aeglustamine toimub suhteliselt teste sooritava arvuti protsessoriga, siis antud aeglustamise strateegia pole mõistlik ja ei annaks pärismaailmaga võrdseid tulemusi.

3.7.1 Jõudlus

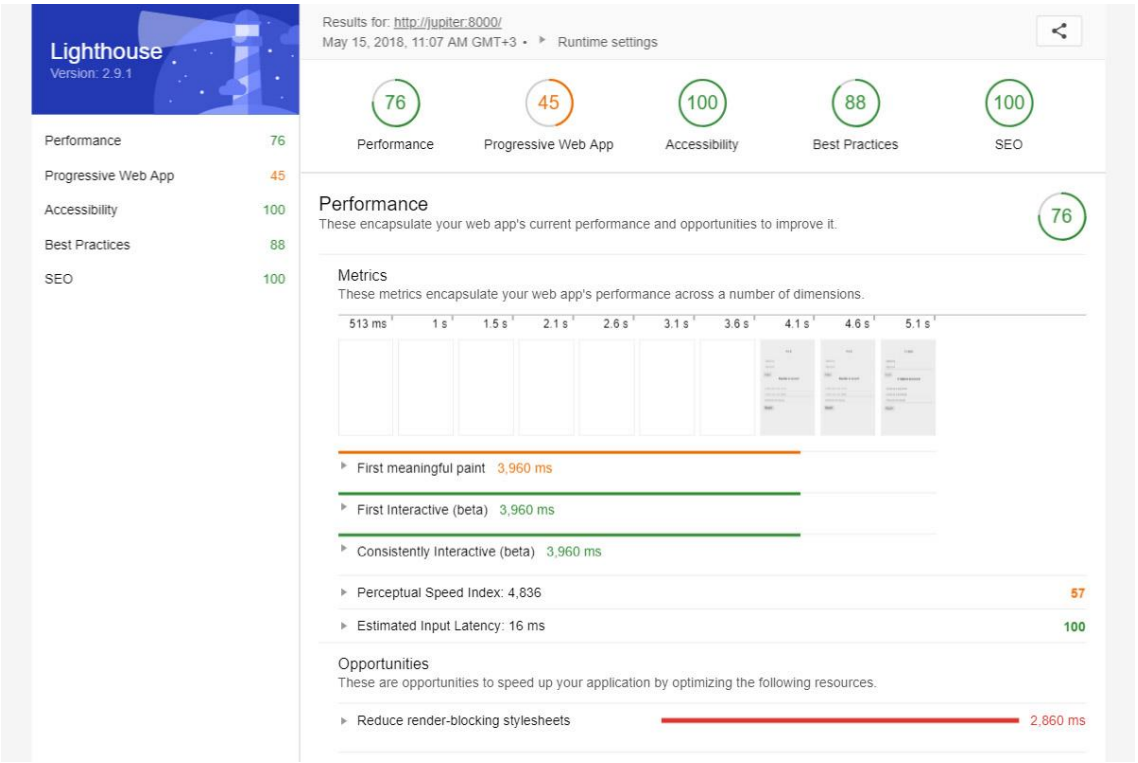
Esimesel testi sooritamisel on jõudluse skoor 31 – kaugel soovitud 75-st. Järgmiseks prooviti rakenduse kompileerimisel (*ng build --prod*) kasutada lisaparameetrit *„--build-optimizer*" [23], aga tulemus ei muutunud oluliselt – hinnanguks oli 32.

Lighthouse'i soovitusel kasutati gzip tehnoloogiat. Gzip on tavapärane failide kokkupakkimine, et vähendada failide suurust. Enamus brausereid oskavad vastavad failid lahti pakkida ja kokkupakkimine toimub samuti automaatselt, kui server on vastavalt seadistatud. Peale gzipi kasutuselevõtmist pakkus Lighthouse hinnanguks 61. Järgmine soovitus oli eemaldada kasutamata CSS reeglid, milleks olid peamiselt Bootstrapi raamistiku reeglid. Tänu modulaarsele ehitusele saab eemaldada komponentide kaupa kasutamata reeglid, tänu millele jõuti hinnanguni 76-ni (Joonis 11).

3.7.2 Juurdepääsetavus, parimad tavad ja SEO

Esimesel testil on juurdepääsetavuse, parimate tavade ja SEO hinnangud vastavalt 60, 63 ja 89. Uurides Lighthouse soovitusi, siis selgus, et rakenduses puuduvad *meta tag*id ja seadistamata oli ka lehe keel, mida vajavad ekraanilugejad. Lisaks soovitati eemaldada konsooli logimised. Peale nende nõuannete implementeerimist olid vastavad hinnangud 72, 88 ja 100.

Ekraanilugejate jaoks on vajalikud sisendlahtritega seostatud lipikud (*label*). Kasutades Bootstrapi klassi „sr-only", saab lisada rakendusele spetsiaalselt ekraanilugejatele mõeldud elemente, mis ei muuda üldist disaini ülejäänud kasutajate jaoks. Tulemuseks oli juurdepääsetavuse hinnang 100.



Joonis 11. Kuvatõmmis Lighthouse raportist peale optimeerimist.

4 Arenduse analüüs

Valminud veebirakendus Indic täidab püstitatud eesmärgi ja nõutud funktsionaalsuse kõik punktid. Saamaks hea ülevaate tehtud tööst, sooritati taaskord analüüs, võrreldes valminud veebirakendust alternatiivsete programmidega.

4.1 Võrdlus alternatiivsete rakendustega

Antud bakalaureusetöö ei ole funktsionaalsuse poolest võrreldav ennist analüüsitud rakendustega, aga Lighthouse'i kasutades saab võrrelda nende jõudlust ja teisi parameetreid.

Tabelist (Tabel 2) on näha, et jõudluse poolest on Indic võrdne Google Finance rakendusega, ülejäänud kaks jäävad juba kaugele maha. Sarnane trend jätkub ka teistes testitud kategooriates - kõige lähemale testi hinnangute poolest jõuab Google Finance ning Yahoo Finance ja Bloomberg on mitmekümne punktiga tagapool. Erinevus tekib alles SEO hinnangus, mis on kõigil hea. Huvitav on see, et Google rakendus sai madalaima tulemuse, ent ilmub otsingumootorites siiski esimese kolme tulemuse seas.

Tabel 2. Tehniliste testide võrdlus.

	Jõudlus	**Juurdepääsetavus**	**Parimad tavad**	**SEO**
Google Finance	76	90	88	78
Yahoo Finance	45	59	69	90
Bloomberg	23	61	63	90
Indic	76	100	88	100

Suuremas pildis pole Indic veebirakendus siiski võrdne analüüsis osalenud programmidega, kuna teistes on oluliselt rohkem funktsionaalsust. Google Finance'i uuenenud veebirakendus samas ei paku enam üldse võimalust portfelle luua, millega võrreldes on arendatud rakendusel oluline eelis.

4.2 Testijate tagasiside

Rakenduse testimine kõrvaliste inimestega tuleb kasuks, et märgata probleeme, mida arenduses osalevad inimesed tähele ei pane või triviaalseteks peavad. Lisaks saab objektiivset hinnangut rakenduse disaini kohta, mida ei saa tehniliste testidega mõõta.

Testijad tõid välja veebirakenduse lihtsuse nii positiivse kui ka negatiivse argumendina. Osa testijaid mainis, et portfellihalduse funktsionaalsus on põhiline ja see on selgelt välja toodud. Sellele vastuargumendiks aga öeldi, et veebirakendus on liiga lihtne ega loo piisavalt lisaväärtust, et seda regulaarselt kasutama hakataks. Testis osales ka inimene, kes isiklikult LHV Panga Kasvukontot kasutab ja ta leidis, et selline abivahend investeeringute haldamiseks oleks kindlasti teretulnud.

Veebirakenduse laadimiskiirust keegi otseselt ei maininud, küll aga soovitas üks testijatest rakendust animatsioonidega sujuvamaks teha. Mõnes kohas vahetub testija sõnul info nii kiiresti, et ta ei märganud koheselt, kas tegevus oli edukas või ei juhtunud üldse midagi.

Välimusega oli enamik testijaid rahul, kuigi paar testijat ei olnud värvilahendusega eriti rahul ja soovitati katsetada teistsuguste värvide mõju veebirakenduses.

Lisaks tehti ka mõned ettepanekud veebirakenduse funktsionaalsuse parandamiseks:

- varaklassi detailvaate graafik võiks olla interaktiivsem, näiteks võimaldada graafiku ajaperioodi muuta;
- võimaldada automatiseerida osakaaludega portfellides osakute ostmist kindlaks määratud aegadel.

4.3 Järeldused

Esmase tagasiside põhjal võib väita, et veebirakenduse arendamine oli edukas. Täidetud said kõik püstitatud eesmärgid ja nõutud funktsionaalsus on loodud veebirakenduses implementeeritud. Tehnilised testid näitasid, et Indic on kiirem kui võrdluses olnud rakendused ning üldiselt jäid ka testkasutajad rakendusega rahule.

Tehnoloogiate valik sobis hästi selle projekti jaoks – Angular on suurepärane ja väga võimekas raamistik, sama võib Node.js kohta öelda. Mõlemad veebirakenduse loomisel

kasutatud tehnoloogiad tagavad, et tulevikus saab veebirakendust lihtsalt edasi arendada ning see suuremale kasutajaskonnale kättesaadavaks teha. Ära peab ka märkima npm pakettide registri süsteemi olulisuse projekti valmimisel - mitmetel kordadel osutus läbi npm-i uute raamistikkude lisamine kõige lihtsamaks ja kiiremaks viisiks, kuidas lõpptulemus saavutada.

Arenduse jagamine etappideks ning iga etapi planeerimine alamülesannete läbi tagas rakenduse kavandamise ning realiseerimisega ajagraafikus püsimise. Kuigi tuli ette olukordi, kus taheti liialt põhjalikult probleemi süveneda ja seda lahendada, siis tänu planeerimisel loodud ülesannete nimekirjale, püsisid fookuses prioriteetsemad probleemid.

Veebirakenduse ainsa arendajana oli töö autoril väga huvitav vaadata testijaid tema loodud rakendust reaalselt kasutamas – kuidas nad lehel navigeerisid, mida varaklassi detailvaates uurisid jne. Testijatelt saadud tagasiside oli konstruktiivne ning juhtis tähelepanu asjadele, mida tulevikus parandada või millist funktsionaalsust juurde lisada. Samuti panid kommentaarid värvilahenduse suunas mõtlema, milline peaks Indicu kui toote bränd olema ja kuidas see veebirakendusega kokku sobiks.

4.4 Edasised arengud

Praegusel kujul ei ole rakendus sobilik suuremal skaalal rohkete kasutajatega kasutamiseks. Veebirakendust tuleks arendada edasi ja leida lahendused järgnevatele teemadele:

- andmebaasi optimeerimine ja struktureerimine;
- serveripoolse loogika täiendused;
- tehingute kuvamine kasutaja määratud valuutas;
- rakenduse turundus ja brändi loomine;
- kasutajate hoidmine.

Andmebaasi tasandil on hetkel kasutusel NoSQL tehnoloogia, mis võimaldas programmi kiiresti arendada ja mitte keskenduda nii palju andmete struktureerimisele,

aga selline lahendus ei ole eriti jätkusuutlik. Tuleks implementeerida relatsiooniline andmebaas ja täiustada serveripoolset loogikat andmebaasiga suhtlemisel.

Serveripoolne loogika vajab täiendusi ka aktsiate andmete pärimisel. Kasutusel olev yahoo-finance API võimaldab vähendada päringute arvu, tehes näiteks mitme erineva varaklassi ajalooliste hindade päring korraga. Mida vähem päringuid, seda mugavam ja kiirem on programmi kasutada. Antud optimeerisvõtet on küllaltki lihtne Expressis implementeerida, aga vajaks olulisel määral ümber ehitamist rakenduse kasutajaliidese poole pealt.

Rakendusele tuleb lisada valuutade konverteerimise moodul, mis arvutaks tehingute ja aktsiate andmed vabalt määratud valuutasse. Lisaks tuleb implementeerida võimalus märkida teenustasude suurust, mida näiteks LHV Pank ostu- ja valuutavahetustehingutelt võtab.

Projekti edukaks toimimiseks on vaja rakenduse kasutajaid, kes seda kindla pidevusega kasutaksid. Rakendusele tuleb hakata arendama brändi ja selle põhjal turundusplaan koostada. Liiga agressiivselt ei tasu turundust teha, et oleks võimalus testida rakendusvõimalusi ja avastada vigu väiksema kasutajabaasi peal, rikkumata brändi mainet suurema kasutajaskonna ees.

Saavutamaks igapäevast külastust registreerunud kasutajate seas, on vaja neile luua ka igapäevast uut sisu. Hea võimalus seda teha oleks lisada uudiste lugemise võimalus kasutajate portfellis olevate varaklasside põhjal. Teine plaan oleks ehitada juurde kommentaarium, kus kasutajad saavad oma mõtteid ja teadmisi teiste kasutajatega jagada.

5 Kokkuvõte

Investeerimise populaarsuse kasvades tekib investoritel vajadus oma portfelle kuidagi hallata. Portfellide halduseks on loodud mitmeid rakendusi. Neid analüüsides selgus, et enamik nendest ei paku kasutaja poolt määratavate osakaaludega portfelli loomise võimalust – funktsionaalsus, mida LHV Panga Kasvukontot kasutaval investoril oleks vaja.

Bakalaureusetöö raames valmis veebirakendus, kus saab kahte erinevat tüüpi portfelle luua ja hallata. Nende info päritakse reaalajas kasutades yahoo-finance API-t ning kõik kasutaja profiiliga seotud info salvestatakse pilveserveris asuvasse andmebaasi.

Püstitatud eesmärgid said täidetud ning tehnilise testide käigus saadi kinnitust, et loodud rakendus on isegi võimekam kui alternatiivsed programmid.

Kasutatud kirjandus

[1] AS LHV Pank, „Kasvukonto,“ [Võrgumaterjal]. Available: https://www.lhv.ee/et/kasvukonto. [Kasutatud 19 Mai 2018].

[2] Mozilla Foundation, „About JavaScript,“ MDN Web Docs, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript. [Kasutatud 19 Mai 2018].

[3] Wikipedia, „JavaScript,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/JavaScript. [Kasutatud 19 Mai 2018].

[4] Microsoft, „Basic Types,“ [Võrgumaterjal]. Available: https://www.typescriptlang.org/docs/handbook/basic-types.html. [Kasutatud 19 Mai 2018].

[5] Node.js Foundation, „About Node.js,“ [Võrgumaterjal]. Available: https://nodejs.org/en/about. [Kasutatud 19 Mai 2018].

[6] npm, Inc, „What is npm?,“ [Võrgumaterjal]. Available: https://docs.npmjs.com/getting-started/what-is-npm. [Kasutatud 19 Mai 2018].

[7] Node.js Foundation, „Express 4.x - API Reference,“ 2017. [Võrgumaterjal]. Available: https://expressjs.com/en/4x/api.html. [Kasutatud 19 Mai 2018].

[8] MongoDB, Inc, „What is NoSQL?,“ 2018. [Võrgumaterjal]. Available: https://www.mongodb.com/nosql-explained. [Kasutatud 19 Mai 2018].

[9] ObjectLabs Corporation, 2018. [Võrgumaterjal]. Available: https://docs.mlab.com/plans/. [Kasutatud 19 Mai 2018].

[10] Google Inc, „FEATURES & BENEFITS,“ 2018. [Võrgumaterjal]. Available: https://angular.io/features. [Kasutatud 19 Mai 2018].

[11] Google Inc, „Architecture overview,“ 2018. [Võrgumaterjal]. Available: https://angular.io/guide/architecture. [Kasutatud 19 Mai 2018].

[12] Google Inc, „Angular CLI,“ 2016. [Võrgumaterjal]. Available: https://cli.angular.io/. [Kasutatud 19 Mai 2018].

[13] Twitter Inc, „Bootstrap · The most popular HTML, CSS, and JS library in the world.,“ 2018. [Võrgumaterjal]. Available: https://getbootstrap.com/. [Kasutatud 19 Mai 2018].

[14] ng-bootstrap team, „Angular powered Bootstrap,“ [Võrgumaterjal]. Available: https://ng-bootstrap.github.io/#/home. [Kasutatud 19 Mai 2018].

[15] JetBrains s.r.o., „IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains,“ 2018. [Võrgumaterjal]. Available: https://www.jetbrains.com/idea/specials/idea/idea.html. [Kasutatud 19 Mai 2018].

[16] JetBrains s.r.o., „For Students: Free Professional Developer Tools by JetBrains,“ 2018. [Võrgumaterjal]. Available: https://www.jetbrains.com/student/. [Kasutatud 19 Mai 2018].

[17] Atlassian, „What is version control | Atlassian Git Tutorial,“ [Võrgumaterjal]. Available: https://www.atlassian.com/git/tutorials/what-is-version-control. [Kasutatud 19 Mai 2018].

[18] UpGuard, „Github vs Bitbucket,“ 19 December 2017. [Võrgumaterjal]. Available: https://www.upguard.com/articles/github-vs-bitbucket. [Kasutatud 19 Mai 2018].

[19] Google Inc, „Lighthouse | Tools for Web Developers,“ 9 April 2018. [Võrgumaterjal]. Available: https://developers.google.com/web/tools/lighthouse/. [Kasutatud 19 Mai 2018].

[20] Statistica, „Mobile internet usage worldwide - Statistics & Facts,“ [Võrgumaterjal]. Available: https://www.statista.com/topics/779/mobile-internet/. [Kasutatud 19 Mai 2018].

[21] P. Huh, „yahoo-finance - npm,“ 2017. [Võrgumaterjal]. Available: https://www.npmjs.com/package/yahoo-finance. [Kasutatud 19 Mai 2018].

[22] Mozilla Foundation, „Property accessors - JavaScript | MDN,“ 17 February 2018. [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Property_accessors. [Kasutatud 19 Mai 2018].

[23] Google Inc, „Angular - Deployment,“ 2018. [Võrgumaterjal]. Available: https://angular.io/guide/deployment. [Kasutatud 19 Mai 2018].

Lisa 1 – Kasutaja mudel

```
import { PortfolioModel } from './portfolio.model';
export class UserModel {
    private _id: string;
    private _email: string;
    private _password: string;
    private _username: string;
    private _firstname: string;
    private _lastname: string;
    private _portfolios: Array<PortfolioModel>;

    constructor( _object: any ) {
        if ( _object ) { this.fillData( _object ); }
    }

    get id(): string { this._id; }
    set id( value: string ) { this._id = value; }
    get email(): string { return this._email; }
    set email( value: string ) { this._email = value; }
    get password(): string { return this._password; }
    set password( value: string ) { this._password = value; }
    get username(): string { return this._username; }
    set username( value: string ) { this._username = value; }
    get firstname(): string { return this._firstname; }
    set firstname( value: string ) { this._firstname = value; }
    get lastname(): string { return this._lastname; }
    set lastname( value: string ) { this._lastname = value; }
    get portfolios(): Array<PortfolioModel> { return this._portfolios; }
    set portfolios( value: Array<PortfolioModel> ) {
        this._portfolios = value;
    }
```

```
    fillData( _object ) {
        if ( _object._id ) { this.id = _object._id; }
        if ( _object.email ) { this.email = _object.email; }
        if ( _object.username ) { this.username = _object.username; }
        if ( _object.password ) { this.password = _object.password; }
        if ( _object.firstName ) { this.firstname = _object.firstName; }
        if ( _object.lastName ) { this.lastname = _object.lastName; }
        if ( _object.portfolios ) {
            const _portfolios: Array<PortfolioModel> = [];
            _object.portfolios.forEach( function ( _portfolio ) {
                _portfolios.push( new PortfolioModel( _portfolio ) );
            } );
            this.portfolios = _portfolios;
        }
    }
}
```

Lisa 2 – Portfelli teenus

```
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs/Observable';
import { PortfolioModel } from '../models/portfolio.model';
import { MathService } from './math.service';
import { StockService } from './stock.service';

@Injectable()
export class PortfolioService {

    constructor( private _math: MathService,
                 private _stockService: StockService ) {
    }

    getPortfolioChange( _portfolio: PortfolioModel ) {
        this.calculatePortfolioChange( _portfolio ).subscribe(
            value => {
                return this._math.changeP( value[ 0 ], value[ 1 ] );
            },
            error => console.error( 'Error: ', error )
        );
    }

    calculatePortfolioChange( _portfolio: PortfolioModel ) {
        const stocks_length = _portfolio.stocks.length;
        let portfolio_cost = 0,
            portfolio_value = 0,
            i = 0;

        return new Observable( observer => {

            if ( stocks_length > 0 ) {
```

```
                for ( const stock of _portfolio.stocks ) {

                    this._stockService.calculateStockChange( stock )
.subscribe(
                        _stock => {
                            i++;
                            portfolio_cost += _stock[ 0 ];
                            portfolio_value += _stock[ 1 ];
                        },
                        err => observer.error( err ),
                        () => {
                            if ( i === stocks_length ) {
                                observer.next( [ portfolio_cost,
portfolio_value ] );
                                observer.complete();
                            }
                        }
                    );
                }

            } else {
                observer.error( 'No stocks in portfolio' );
            }
        } );
    }
}
```