TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Marek Puik 134222IAPB

VEEBIRAKENDUS RAHALISTE KULUDE JÄLGIMISEKS JA ANALÜÜSIKS

bakalaureusetöö

Juhendaja: Gert Kanter

Tehnikateaduse magister

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: Marek Puik

11.05.2019

Annotatsioon

Antud lõputöö eesmärgiks oli luua veebirakendus, mis aitab kasutajal hallata oma kulusid ja tulusid. Rakendus võimaldab luua visuaalseid jooniseid kasutaja rahalise vahendite ja nende kasutamise kohta. Samuti oli arendamisel oluline, et oleks kulude projekteerimise ja optimeerimise funktsionaalsus.

Eesmärkide saavutamiseks analüüsiti erinevaid tehnoloogiaid ja erinevate kaupluste veebilehtede andmestikku. Töö tulemusena tekkis rakendus, mis suudab konto väljavõtte alusel kuvada kasutajale erinevaid jooniseid. Samuti on rakendusel funktsionaalsus ennustada kasutaja kulusid ühe kuu lõikes ning oskab välja tuua neid tooteid mille pealt on kasutajal võimalik kokku hoida.

Lõputöö on kirjutatud eesti keeles keeles ning sisaldab teksti 32 leheküljel, 6 peatükki, 20 joonist, 1 tabel.

Abstract

Web application for monitoring and analyzing financial costs

The main purpose of this thesis is to create a web application, that gives the user an overview of their income and spendings. The application uses visual graphs to display information about the user's wealth and how the user has spent their money. The graphs are able to show how the user has spent their money by category and also by the most common transactions.

To reach the goals of the thesis, the author analyses different technologies and the webpages of different grocery stores and how the data is displayed on them. As a result an application is created, that is able to create different graphs and display them to the user. The data for the graphs comes from the bank account statement. The application has the capability to somewhat predict how much money the user usually spends per month, which is based upon the history of the users spendings. The application also has the capability to notify users when a product they usually buy is on sale or cheaper than usual. Notifications can be sent via email or are displayed on the webpage.

The thesis is in Estonian and contains 32 pages of text, 6 chapters, 20 figures, 1 table.

Lühendite ja mõistete sõnastik

Ajax	*Asynchronous Javascript and XML*, kasutusel rakenduse poolel serveriga suhtlemiseks.
CSS	*Cascading Style Sheets*, kirjeldab elementide kujutust veebis.
CSV	*Comma Separated Values*, tekstifail mis sisaldab andmete kogu.
DOM	*Document Object Model*, puu HTML objektidest.
Git	Versioonihaldustarkvara.
Javascript	Programmeerimiskeel.
jQuery	Javascripti raamistik, mis lihtsustab Javascripti kirjutamist.
PHP	*Hypertext Preprocessor*, programmeerimiskeel serveri poolel.
Web scraping	Veebist andmete kättesaamise meetod.
XML	*Extensible markupl language*, andmete kujutamise formaat.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Antud töö eesmärgiks on luua rakendus mis annab kasutajale visuaalse pildi enda sissetulekutest ja väljaminekutest, aitab planeerida enda rahaliste vahendite kasutamist ning pakub alternatiive igapäevaste ostude puhul. Rakendus loob erinevaid informatiivseid graafikuid kasutaja konto väljavõtte alusel, pakub võimalust võrrelda erinevaid perioode ja teha märkeid kalendrisse. Samuti proovib rakendus ennustada kasutaja kulutusi ühe kuu piires ning võimaldab kulutusi optimeerida, tuues välja kasutajale sooduspakkumisi mis on igale kasutajale sobivad. Süsteemil on ka taustateenused mis koguvad e-poodidest toidukaupade andmeid ning saadavad kasutajatele meile nende toodete soodustuste kohta mida kasutaja on varem soetanud.

2 Ülevaade tööst

2.1 Taust ja probleem

Hetkel ei ole laialt levinud sellist rakendust, mis annaks infot kasutajatele oma rahaliste vahendite üle sellisel viisil, et rakendus suudab visualeerida kulusid ja tulusid, suudaks mingil määral ennustada kasutaja kulutusi ning suudaks välja tuua kohti kus oleks võimalik säästa.

Pankadel on küll olemas oma rahaplaneerija teenused, kuid nende informatiivsus on piiratud, ning puudub ennustav funktsionaalsus. Samamoodi on suurtel toidupoodidel olemas enda veebilehed oma toodete kuvamiseks, kuid sealt info kätte saamiseks peab kasutaja ise vaeva nägema ja otsima, eriti kui on soov vaadata mitmest poest.

Selleks, et valida sobivad toidupoed mille ühilduvust rakendusega hakatake looma, võrdles töö autor suurimate toidupoodide veebilehti ja toodete kättesaadavust (Tabel 1). Eestis asuvad enamlevinud toidupoed on: Selver, Prisma, Rimi, Coop, Maxima, Grossi ja Comarket. Võrdluses on näha, et kõiki tooteid kuvavad ainult 3 toidupoe veebilehed, ning kõik need tooted on kuvatud teksti kujul. Need lehed kus on tooted kuvatud pildina, ei ole rakendusel võimalik kätte saada andmestikku.

Tabel 1. Toidupoe veebilehtede võrdlus

Pood	**Selver**	**Prisma**	**Rimi**	**Coop**	**Maxima**	**Grossi**	**Comarket**
Toodete kättesaadavus	Kõik	Kõik	Soodus	Kõik	Soodus	Soodus	Soodus
Esitamise kuju	Tekst	Tekst	Tekst	Tekst	Tekst	Pilt	Pilt

Töö autor valis rakendusega ühildamiseks selveri ja prisma toidupoed, kuna need pakuvad talle kõige rohkem huvi ja infot.

2.2 Ülesandepüstitus

Eelnevate probleemide alusel on töö eesmärk luua selline rakendus, mis suudab visuaalsel viisil kasutajale kuvada ülevaate oma sissetulekutest ja väljaminekutest. Samuti peab rakendus oskama mingil määral ennustada kasutaja kulutusi ning pakkuma võimalust säästmiseks.

2.3 Konto väljavõte

Tehes pangakaardiga tehinguid, olgu see siis internetis või reaalselt kaardiga makstes, jääb panga kontole nimekiri tehingutest. Iga tehing on konto omanikule kätte saadav vastavas interneti pangas. Antud rakenduse arendamisel on arvesse võetud swedbank konto väljavõtte formaadist. Kõik tehingud omavad väljasi *kuupäev*, *saaja / maksja*, *selgitus*, *käive*, *saldo*. Interneti pangas on võimalik teha väljavõtet konto tehingutest ning neid on võimalik eksportida erinevates formaatides, näiteks csv, bdoc, pdf või xls. Antud rakendus kasutab xls formaadis konto väljavõtteid.

Eksporditud xls failid on sarnase formaadiga, kõik read on indekseeritud kuid failil on ka päis ja jalus. Päis sisaldab informatsiooni konto omanikust, numbrist, perioodist ja väljavõtte tegemise ajast. Jaluses on info kogu perioodi väljaminekute ja sissetulekute kohta. Rakendus ei loe ega salvesta informatsiooni mis asub päises või jaluses kuna see on ebaloluline, rakendus loeb endasse ainult tehingud.

2.4 Vahendid

Antud süsteem koosneb kolmest suuremast osast: veebirakendus, andmebaas ning tausta rakendused.

Veebirakendus kasutab mitmeid erinevaid Javascripti [1] teeke. Javascript on veebilehtede skriptimiskeel miks on üks laialdasemalt levinud. Javascripti baasil on loodud väga palju teeke, mida on ka antud töös kasutusele võetud. Javascript võimaldab luua funktsioone, manipuleerida elemente veebilehel ning ka muuta veebilehe ilmet ehk muuta CSS-i.

jQuery [2] kasutus on kõige laialdasem antud projektis, millega on realiseeritud enamus back-end osa. jQuery lihtsustab klassikalise Javascripti kirjutamist ning aitab

vähendada koodi ridu, samuti pakub funktsioone mida on mugavam kirjutada ja lugeda. jQuery kasutuselevõttu põhjendab ka see, et jQueryga on võimalik teha AJAX [3] päringuid. Ajax võimaldab rakendusel teha päringuid serverile ja sealt saada andmeid. Nende andmete põhjal on rakendusel võimalik uuendada veebilehte ilma et peaks lehte värskendama. Ajaxi päringuid saab teha nii asünkroonselt kui ka sünkroonselt.

jQueryga koos on veel kasutatud SheetJS-i [4] mis pakub exceli failide lugemist ja töötlemist. Jooniste ja graafikute kujundamiseks ja funktsioneerimiseks on kasutusele võetud Chart.js [5]. Chart.js pakub erinevaid graafiku väljundeid ja pakub laialdaselt võimalusi nende muutmiseks. Joonistest piltide eksportimist võimaldab html2canvas [6] teek, mis toetub Javascriptile.

Ostutšekkidel olevate toodete nimede sobitamiseks andmebaasi toodetega on kasutatud Fuse.js [7]. Fuse.js pakub võimalust mingit fraasi võrrelda mingi listiga ja pakub sealt listist kõige ligilähedasemaid vasteid. Selle teegi kasutusele võtmine on oluline just seetõttu, et ostutšekkidel olevad toodete nimetused ei vasta alati nendega, mis nendel samadel poodidel on internetti märgitud.

Rakenduse kujunduse loomiseks on kasutusele võetud Bootstrap [8]. Bootstrap on üks populaarseimaid CSS, HTML ja JavaScript raamistikke. Bootstrap pakub erinevaid lahendusi nii veebilehe kujundamisel kui ka veebilehe funktsionaalsuse täiendamisel. Veebilehe kujundamisel on lähtutud CSS Grid Layout [9] ideedest, mis tähendab, et leht on jagatud mingiteks kindlateks osadeks ja nende osade sees paikenvad elemendid.

Andmebaasiks on võetud MySQL [10] andmebaas ning rakenduse ja baasi vaheliseks suhtluskeeleks on kasutusel PHP [11], kuna see sobib väga hästi MySQL andmebaasidega kokku. PHP on skriptimiskeel mis on enamjaolt mõeldud veebilehtede arendamiseks. PHP võimaldab teha päringuid andmebaasile, luua registreerimise ja logimise vorme ja palju muud.

Taustateenuste käivitamiseks kasutatakse Node JS-i [12]. Node JS võimaldab Javascripti koodi käivitada väljaspool veebilehitsejat. Taustateenused mis koguvad veebi lehtedelt infot kasutavad teeke cheerio [13] ja Nightmare.js [14]. Cheerio kasutab jQuery süntaksit ning võimaldab veebilehtedelt allalaadida andmeid. Nightmare.js on mõeldud veebilehtede sirvimise ülesannete automatiseerimiseks. Kasutajatele meilide saatmiseks on kasutusel nodemailer [15].

3 Arendus

3.1 Arenduskäik

Töö autor alustas arendust sobiva andmebaasi otsimisest ning Javascripti teegi otsimisest mis pakuks lahendust xls tüüpi failide töötlemiseks. Autor valis andmebaasiks MySQL andmebaasi, kuna sellega on tal varem tööalane kogemus ning MySQL sobib hästi kokku PHP päringutega. Projekti haldamiseks kasutas autor versioonihaldustarkvara Git.

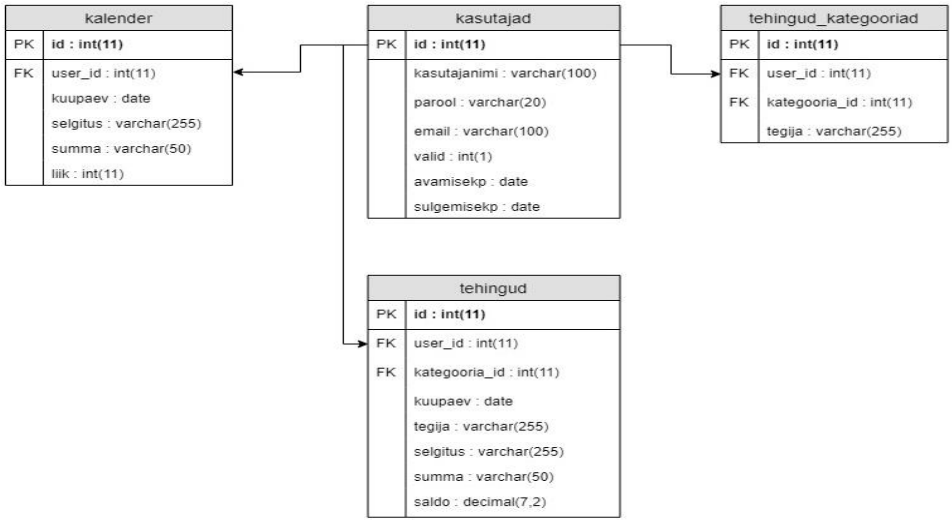
Veebirakenduse arendus algas registreerimise ja sisselogimise vormidest, peale mida realiseeriti xls tüüpi failide lugemise funktsionaalsus. Edasi mindi *dashboard*-de tutvumisega ning erinevate jooniste loomisega mis oleksid kasutajale mugav vaadata ning oleksid informatiivsed.

Kui oli saavutatud funktsionaalsus, mis pakub jooniste loomist, hakkas autor katsetama erinevaid *web scraping* [16] tehnoloogiaid, et oleks võimalik kasutajatele välja tuua neid tooteid, mille pealt on võimalik kasutajal kokku hoida. Poed mille tooteid hakkas autor veebilehtedelt kokku korjama olid selver ja prisma. Need poed said valitud seetõttu, et neil olid kõik tooted veebilehtedel olemas ja need enim külastatuimad poed.

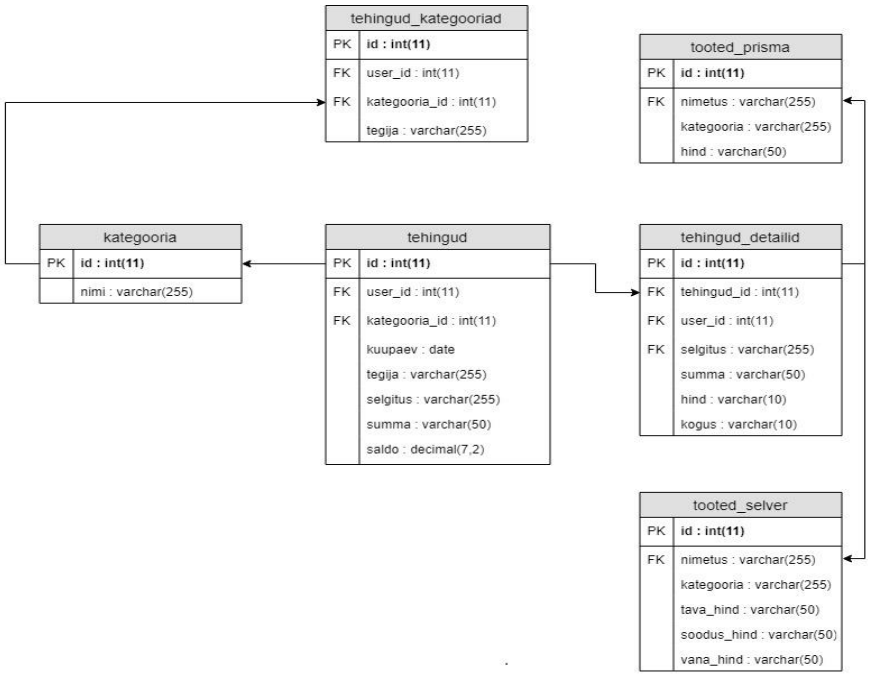
Arenduse lõpuks viimistles autor veebirakenduse välimust, lõi dashboardi kus enamus infot on kokku toodud, ning täiendas üldist kasutajamugavust ja funktsionaalsust.

3.2 Andmebaas

Peale andmebaasi leidmist seadis töö autor üles andmebaasi ja lõi esimesed tabelid. Kõik tabelite loomise päringud asuvad kaustas nimega „scriptid“. Andmebaasis on 2 põhilist tabelit, milleks on „kasutajad“ (Joonis 1) ja „tehingud“ (Joonis 2). Igal kasutajal on õigus näha ja muuta ainult enda kasutajaga seotud andmeid, samuti on igal kasutajal enda kalender ning tegijate kategooriad. Viimane on oluline just seetõttu, et igal kasutajal ei ole täpselt sama kategoriseeritus tegijate kaupa. Näiteks võib üks isik käia Viimsi SPA-s lõõgastumas ja vaba aega nautimas, teine aga sporti teha. Isikustatud kategooriad on olulised seepärast, et rakendus saaks kuvada kasutajale korrektseid graafikuid.



Joonis 1. Kasutajad olemi-suhte diagramm



Joonis 2. Tehingud olemi-suhte diagramm

Tehingute kategooriate automaatseks lisamiseks andmebaasi on loodud trigger (Joonis 3). Trigger lisab tabelisse „tehingud_kategooriad“ iga tegija kohta kategooria mis on kasutaja poolt sisestatud ning mille kategooria_id ei ole 1 ehk „PUUDU“.

```
CREATE DEFINER=`root`@`localhost` TRIGGER `t_after_update` AFTER
UPDATE ON `tehingud` FOR EACH ROW
	IF NEW.tegija NOT IN
		(SELECT tehingud_kategooriad.tegija FROM
		tehingud_kategooriad)
		AND NEW.kategooria_id IS NOT NULL
		AND NEW.kategooria_id NOT IN (1)
		AND NEW.user_id IS NOT NULL
	THEN
		INSERT INTO
		tehingud_kategooriad(tegija,kategooria_id,user_id)
		VALUES (NEW.tegija, NEW.kategooria_id,
		NEW.user_id);
	END IF
```

Joonis 3. Kategooriate lisamise trigger

Baasis olevad e-poodide tooted tulevad vastava poe veebilehelt ning nende uuendus käib perioodiliti vastavalt vajadusele, näiteks 1 kord päevas. Nende tabelite uuendamine on oluline, et kasutajad saaksid iga päev teateid nende toodete kohta, millest nad on huvitatud ja millel on tavalisest väiksem hind.

3.3 Rakenduse arendus

3.3.1 Registreerimise- ja sisselogimisvormid

Rakenduse arendus algas sisselogimise ja registreerimise vormi loomisest. Registreerimisel on lihtne vorm kus tuleb täita 4 välja: kasutajatunnus, email parool ja parooli kinnitus. Parooli krüpteerimiseks kasutatakse PHP funktsiooni *password_hash* ning dekrüpteerimiseks *password_verify*. Kasutajanime ja parooli valimisel on nõutud, et need koosneksid ainult tähtedest, numbritest ja etteantud märkidest vähemalt ühte ning oleksid pikkusega 4-20 tähemärki. Viimase nõude kontrollimiseks kasutati rakenduses PHP funktsiooni *preg_match* mille esimene parameeter on regexi muster kujul „/[\'^£$%&*()}{@#~?><>,=_+!-]/“. See muster ootab sisendina mingit sõne mis sisaldab vähemalt ühte eelnimetatud märkidest. Sql injectioni [17] ennetamiseks on kasutatud PHP funktsiooni „mysql_real_escape_string“, mis asendab spetsiaalsed märgid vastavate koodidega.

Sisselogimisel küsib rakendus kasutajatunnust ning parooli kus on samad tingimused kasutusel mis olid registreerimise puhul.

3.3.2 Exceli faili üleslaadimine ja töötlemine

Exceli faili lugemiseks kasutab rakendus Javascripti teeki SheetJS mis võimaldab mõne koodi reaga anda rakendusele ette xls tüüpi fail ning sealt võtta vajalik info. Xls faili sisu lisatakse uude 2 dimensioonilisse arraysse kust on hiljem lihtne kätte saada iga tehingu andmed. Faili sisselugemisel esimestest ridadest minnakse üle, kuna failis olev päis on rakenduse jaoks ebaoluline ning võib sisaldada informatsiooni mida kasutaja ei taha jagada. Peale faili üleslaadimist korjab rakendust tehingute küljest kõik unikaalsed tegijad ning küsib kasutajalt nende kategooriate kohta. Kõik kategooriad on iga kasutaja jaoks unikaalsed. Seejärel laetakse andmed üles andmebaasi kasutades Ajax post meetodit. Samal ajal luuakse ka rakendusse tabel, et näidata kasutajale visuaalselt mis andmeid ta hetkel lisas.

3.3.3 Joonised

Jooniste loomiseks on kasutuses Javascripti teek Chart.js mis pakub eri tüüpi jooniseid, võimaldab nende modifitseerimist ja animeerimist. Rakenduse arendamisel võeti kasutusse 3 graafiku tüüpi: tulpdiagramm, joondiagramm ja sõõrik. Igale joonisele on lisatud salvestamise funktsionaalsus (Joonis 4) ja lähteandmete allalaadimise funktsionaalsus (Joonis 5), mis võimaldab kasutajal vajaduse korral neid edasi töödelda või presenteerida.

```
$('#indexDiv').on('click', 'a[name="save-picture"]', function(){
	var HTMLcanvas =
$(this).parent().parent().parent().parent().parent().find('canvas');
	html2canvas(HTMLcanvas, {
		onrendered: function(canvas) {
			var link = document.createElement('a');
			link.href = canvas.toDataURL('image/jpeg');
			link.download = 'chart.jpeg';
			link.click();
		}
	});
});
```

Joonis 4. Pildi allalaadimise meetod

```
$('#indexDiv').on('click', 'a[name="export-chart"]', function(){
	var index = ($(this).attr('data-chart'));
	var wb = XLSX.utils.book_new();
	wb.Props = {
		Title: "eksport",
		Subject: "eksport",
		Author: "analyser",
		CreatedDate: new Date()
	};
	wb.SheetNames.push("Sheet1");
	var ws = XLSX.utils.json_to_sheet(xlsObj[index]);
	wb.Sheets["Sheet1"] = ws;
	var wbout = XLSX.write(wb, {bookType:'xlsx', type:'binary'});
	function s2ab(s){
		var buf = new ArrayBuffer(s.length);
		var view = new Uint8Array(buf);
		for(var i=0; i < s.length; i++) view[i] = s.charCodeAt(i)
& 0xFF;
		return buf;
	}
	saveAs(new Blob([s2ab(wbout)],{type:"application/octet-
stream"}), 'eksport.xlsx');
});
```

Joonis 5. Lähteandmete eksportimise meetod

3.3.4 Kulu ennustamine

Kasutaja kulude ennustamiseks kasutab rakendus maksimaalselt viimase 12 kuud andmeid. Kasutajal on võimalik perioodi muuta kas viimase 6 kuu järgi, viimase 3 kuu järgi või viimase kuu järgi. Samuti on valik kas arvutused tehakse kuu lõikes, või päeva lõikes. Kuu lõikes valib programm baasist perioodi järgi kõik sellised tehingud mis on väljaminekud, summeerib nende tehingute summad, kuu kaupa ja seejärel jagab kuude arvuga, et saada kätte kuu keskmine.

Kui kasutaja valib aga päeva lõikes, arvutab programm iga kuu päeva kohta eraldi palju tal raha kuluda võiks. Rakendus käib läbi päeva numbri järgi kõik päevad valitud perioodi kohta, summeerib nende päevade kulutused ja jagab selle päevade arvuga, mitmes kuus need päevad esinevad. Seejärel läheb edasi järgmise päeva peale, jättes meelde eelneva päeva kulutuse summa.

3.3.5 Detailandmete lisamine

Et kasutaja kulutusi oleks võimalik optimeerida, on vaja rakendusel tooteid mille hulgast otsida vajalikku infot. Selle info sisestamiseks on rakendusele loodud 2

võimalust. Kasutajal on võimalus igale prisma või selveri poe tehingule, mis on viimase 30 päeva jooksul sooritatud, lisada juurde detail andmed, ehk need samad andmed mis on ostutšekil. Teine võimalus on kasutada liidestust ostutšekkide digitaliseerimisega [18], mis teeb ostutšekkist digitaalse versiooni XML [19] formaadis. Kui tehingule lisada digitaalne ostutšekk xml formaadis, siis töötab rakendus selle läbi ning lisab vastavad tooted tehingule juurde.

Kuna ostutšekkidel olevad toodete nimed ei vasta alati nendega, mis on internetis e-poes, on rakenduses kasutusele võetud ennustav sõnaotsing, Fuse.js. Fuse liidestus võimaldab otsida ühest listist mingit kindlat fraasi ja seejärel tagastab need fraasist sellest listist, mis vastavad seadistustele (Joonis 6).

```
var options = {
  shouldSort: true,
  threshold: 0.2,
  includeScore: true,
  location: 0,
  distance: 100,
  maxPatternLength: 50,
  minMatchCharLength: 1
};
```

Joonis 6. Sõnaotsingu seadistused

Need tooted millele vastet ei leitud, lisatakse listi ja kui kõik tooted on läbi käidud, tagastatakse see list kasutajale hüpikaknas.

3.3.6 Optimeerimine

Kulude optimeerimiseks on võetud arvesse need tehingud, mis on tehtud selveri või prisma poodides, ning need tehingud peavad olema tehtud viimase 30 päeva jooksul. Selver ja prisma on valitud seetõttu, et nende tooted koos hindadega on internetis üleval ja kehtivate hindadega. Kui andmebaasis esineb mõni selline toode, mida kasutaja on viimase 30 päeva jooksul endale soetanud, ning selle hetke hind või soodushind on alla selle, millega tema selle soetas, siis kõik sellised tooted kasutajale kuvatakse vastavas tabelis. Sama loogika järgi saab kasutajale kuvada ka neid tooteid, mis on kallimaks muutunud.

4 Taustateenused

Rakendusel on ka 3 teenust mida jooksutatakse väljaspool veebirakendust ehk serveris. Nende teenuste jooksutamiseks kasutatakse Node.js-i. Node.js on avatut lähtekoodiga programm, mis võimaldab käivitada Javascripti koodi väljaspool veebirakendust.

Kasutusel on 2 *scrape* teenust mis käivad veebilehed läbi ja korjavad sealt andmeid. Web scrapingu all tuntakse tehnoloogijaid mis koguvad andmeid veebilehtedelt kas läbi HTTP või veebilehitseja. Web scraping on sisuliselt sama nagu oleks käsitsi kõikide toodete nimetuste ja hindade üles kirjutamine exceli faili, kuid kogu see protsess on automatiseeritud ning ajakulu on vähendatud väga suurel määral.

Taustal jookseb ka meiliteenus mis saadab andmebaasis olevatele kasutajate meiliaadressidele infot nende toodete kohta, mis on hetkel allahindlusega või tavalisest odavamad.

4.1 Selveri veebipood

Esimene taustateenus on nimetusega „scrapeSelver“ (Joonis 7) ja selle eesmärk on koguda selveri veebilehelt kõikide toodete hinnad, koos soodushindade ja nimetustega. Kuna selveri veebileht ei kasuta uuemaid tehnoloogiaid, nagu näiteks React [20] või Angular [21], on veebilehe DOM-is [22] kogu info juba olemas, mida teenusel on vaja, ehk ei ole vaja avada veebilehitsejat vaid piisab HTTP pöördumisest.

Kasutusele on võetud Cheerio JS mis muudab lihtsamaks ja mugavamaks DOM elementide käsitlemise. Programm, mille autor lõi, läheb selveri veebilehele, käib ükshaaval kõikide kategooriate kõik lehed läbi ja korjab iga toote andmed kokku. Andmete kirjutamine toimub sünkroonselt, st. et samal ajal kui programm külastab iga kategooria igat lehekülge, korjab ta samal ajal andmed kokku ja kirjutab need faili. Faili kirjutatakse andmed CSV formaadis, et lihtsustada andmete sisestamist andmebaasi.

```
var reqString = 'https://www.selver.ee/'
              +kategooriad[j]+'?limit=96&p=';
              var url = 'https://www.selver.ee/'+kategooriad[j]+'?limit=96';
              try {
                     const html = await preloadPage(url);
                     var $ = cheerio.load(html);
                     var pageCount = $('div.toolbar > div.pages >
                     ol.pagination > li')
                     .eq(-2).find('a').text();
                     for(var i = 1; i <= pageCount; i++){
                            const html = await preloadPage(reqString+i);
                            var $ = cheerio.load(html);
                                   $('#products-grid').find('li')
                                   .each(function(){
                                          var $el = $(this);
```

Joonis 7.Cheerio pöördumine veebilehe poole

4.2 Prisma veebipood

Prisma toodete kogumiseks on autor loonud teise teenuse nimega „scrapePrisma“ ja selle eesmärk on sama nagu selveri oma, koguda tooted nimetuste ja hindadega kokku. Kuna prisma veebilehel on kasutusel uuemad tehnoloogiad, ei ole eelnev Cheerio kasutamine võimalik. Seetõttu on autor võtnud kasutusele Nightmare.js'i. Nightmare.js on Javascripti teek mis käivitab *headless browser*'i. *Headless browser* on veebilehitseja millel puudub kasutajaliides, ehk puudub otsingu riba, järjehoidjate riba ja lisamine ja kõik muu.

Kuna prisma veebilehelt pole võimalik HTTP pöördumisega toodete nimekirja kätte saada, on töö autor kirjutanud programmi mis kasutades Nightmare.js'i avab *electron* [23] lehitseja ja hakkab läbi käima prisma lehte. Kuna prisma veebileht on teistsuguse ülesehitusega, on ka andmete kogumise loogika natuke teistsugune. Esmalt peab lehitseja avama kategooria, seejärel alamkategooria ja seejärel kerima lehe lõppu (Joonis 8), et saada kõik tooted kätte. Nii käib programm läbi kõikide kategooriate alamkategooriad, et saada kätte kõik tooted koos hindade ja nimetustega. Sarnaselt eelneva scraperiga, kirjutab ka prisma scraper sünkroonselt kõik andmed csv formaadis faili.

```
var previousHeight, currentHeight=0;
while(previousHeight !== currentHeight) {
	previousHeight = currentHeight;
	var currentHeight = yield nightmare.evaluate(function() {
		return document.body.scrollHeight;
	});
	yield nightmare.scrollTo(currentHeight, 0)
	.wait(500);
}
```

Joonis 8.Nightmare.js lehitseja kerimine

Kui selveri toodete kätte saamiseks läheb keskmiselt 300-400 sekundi vahel, siis prisma toodete kättesaamiseks kulub alates 600st sekundist kuni 1000 sekundini. Peale andmete kätte saamist tehakse andmebaasis toodete tabel tühjaks ning sisestatakse uued tooted automaatselt.

4.3 Meiliteenus

Meilide saatmiseks on kasutusel Node.js'i moodul Nodemailer mis võimaldab rakendustel saata emaile. Meilide saatmiseks on vaja anda ette aadress kust saadetakse, kuhu saadetakse, pealkiri ning sisu (Joonis 9). Rakendus saadab igale kasutajale personaliseeritud meili. Meilis on informatsioon selle kohta, mis tooted on hetkel odavamad kui tavaliselt.

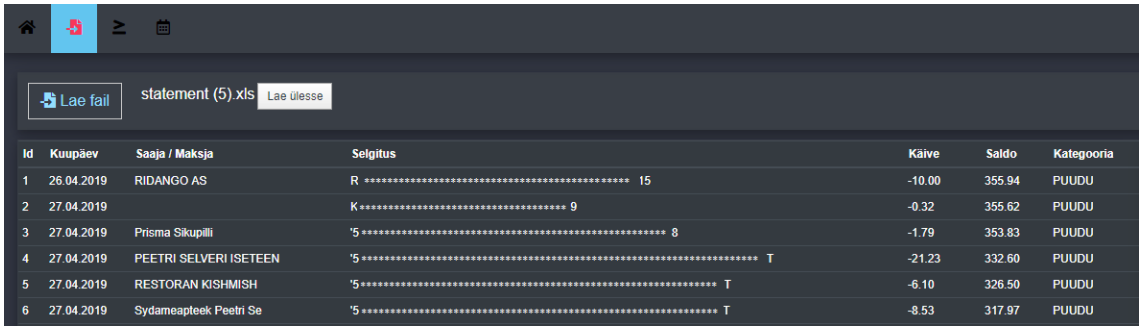
```
var mailOptions = {
	from: 'analyserservice@gmail.com',
	to: curEmail,
	subject: 'Päevane väljavõte',
	text: sendText
}
transporter.sendMail(mailOptions, function(error, info){
	if (error){
		console.log(error);
	} else {
		console.log('Email sent: ' + info.response);
	}
});
```

Joonis 9. Koodinäide meili saatmiseks

5 Kasutuslood kasutaja vaatest

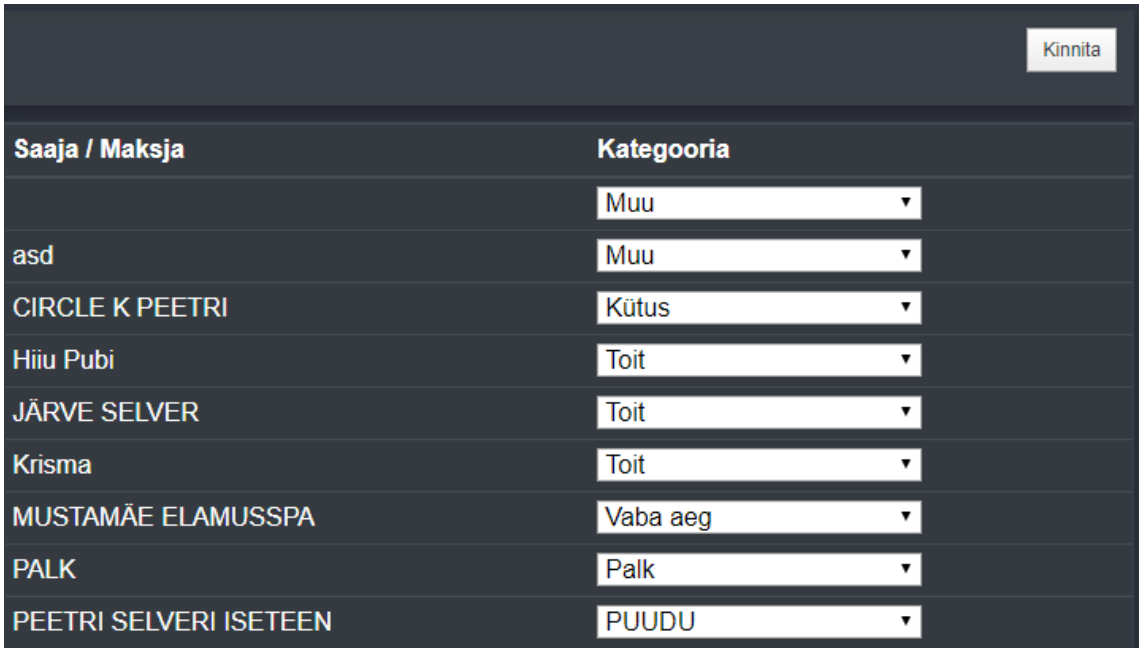
5.1 Andmete sisestamine

Peale registreerimist ning esmast sisselogimist on oluline, et kasutaja lisaks enda kontole esmased andmed. Andmete sisestus käib üleslaadmise saki alt (Joonis 10). Kui exceli üleslaadimine on õnnestunud, küsib rakendus kasutajalt iga unikaalse tegija kohta kategooriat (Joonis 11). Need kategooriad salvestatakse andmebaasi ning tulevikus uute andmete juurde lisamisel olemasolevate tegijate kohta kategooriaid ei pea lisama, need tulevad automaatselt. Peale kategooriate kinnitamist salvestatakse tegijate kategooriad andmebaasi kasutajapõhiselt.



Id	Kuupäev	Saaja / Maksja	Selgitus	Käive	Saldo	Kategooria
1	26.04.2019	RIDANGO AS	R ********** 15	-10.00	355.94	PUUDU
2	27.04.2019		K********** 9	-0.32	355.62	PUUDU
3	27.04.2019	Prisma Sikupilli	'5********** 8	-1.79	353.83	PUUDU
4	27.04.2019	PEETRI SELVERI ISETEEN	'5********** T	-21.23	332.60	PUUDU
5	27.04.2019	RESTORAN KISHMISH	'5********** T	-6.10	326.50	PUUDU
6	27.04.2019	Sydameapteek Peetri Se	'5********** T	-8.53	317.97	PUUDU

Joonis 10. Andmesisestus sakk



Joonis 11. Kategooriate lisamine

5.2 Perioodide võrldus

Peale andmete üleslaadimist on kasutajal võimalik hakata analüüsima enda sissetulekuid ja väljaminekuid. Esmalt on võimalik kasutajalt panna kõrvuti kaks perioodi, et võrrelda kuidas nendel perioodidel on kasutajale laekunud raha ning kontol toimunud väljaminekud. Näiteks saab kasutaja vaadata kahte kuud korraga ning näha, kui palju on tal neil kuul sissetulekuid ning väljaminekuid olnud.

Perioodide võrdlus aitab ka planeerida tuleviku kulutusi. Kui näiteks kasutaja mõtleb laenu või liisingu võtmise peale, saab ta võrrelda näiteks kahe poolaasta kaupa, et näha kuidas suvisel ja talvisel ajal tema kogu kulutused erinevad. Üldjuhul on talvel kulutused kõrgemad, kuna siis on mitmed pühad, kütte arved kõrgemad, käiakse reisil ning soetatakse talve riideid või varustust.

Jäädes planeerimise teemale, siis on kasutajal võimalik kasutada ka rakenduse kalendrit, mis võimaldab tal märkida sissetulekuid või väljaminekuid. Kõik kuupäevad kuhu kasutaja on märkinud mingi info, värvitakse kalendris ära, et oleks kasutajal parem ülevaade ja kergem leida neid päevi kuhu ta on midagi planeerinud.

5.3 Dashboard

5.3.1 Valikud

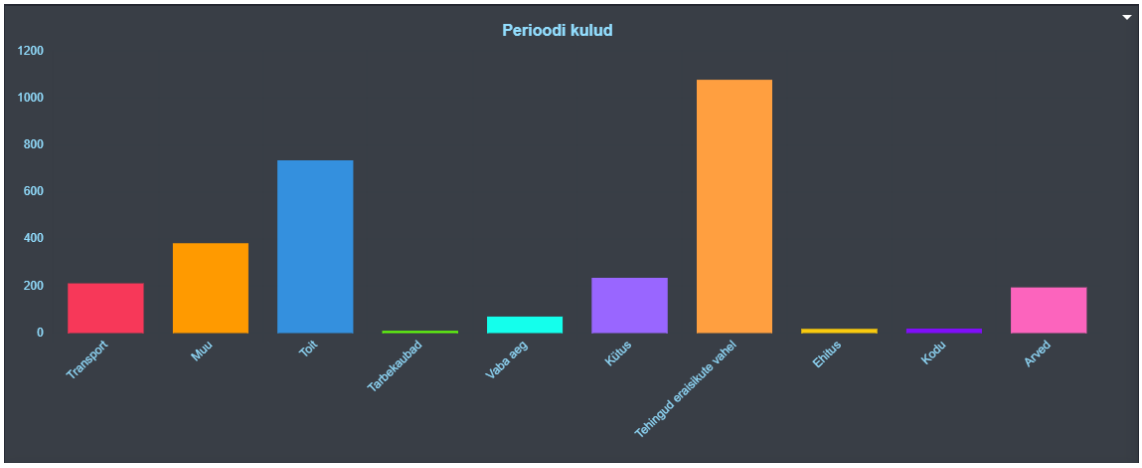
Kui kasutaja on läbi rakenduse lisanud tehingud ning tegijate kohta kategooriad, tekib veebirakenduse esilehele dashboard ning joonistatakse graafikud. Vaikimis võtab rakendus esmaseks perioodi alguskuupäevaks eelmise kuu esimese päeva ning lõpp kuupäevaks tänase päeva. Vajadusel saab kasutaja valikud saki peale vajutades vahetada perioodi ning samuti näha kõiki tehinguid mis on tehtud antud ajavahemikus (Joonis). Valikute sakil on ka näha perioodi tulusi ja kulutusi ning kas kokku on periood negatiivse või positiivse jäägiga.

Joonis 12. Perioodi valikud

5.3.2 Joonised

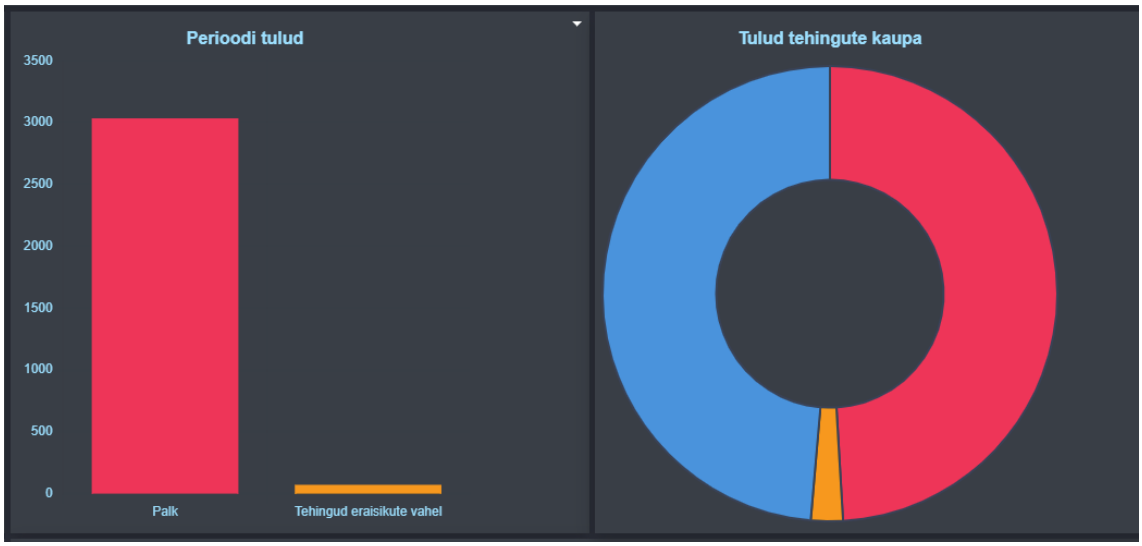
Dashboardil kujutatud joonised annavad infot kasutaja tulude ja kulude kohta. Kokku on dashboardil 7 joonist. Esimesed 5 graafikut on joonistatud valitud perioodi andmete alusel.

Esimene joonis (Joonis 13) kujutab valitud perioodil tehtud kulutusi kategooriate kaupa. Rakendus valib perioodi järgi kõik tehingud ning seejärel summeerib iga kategooria kaupa tehingute summad ning lisab need canvas elemendile. Kategooriate välja toomiseks on kasutatud tulpdiagrammi, kus vertikaalselt kuvatakse summa suurust ningi horisontaalselt erinevaid kategooriaid.



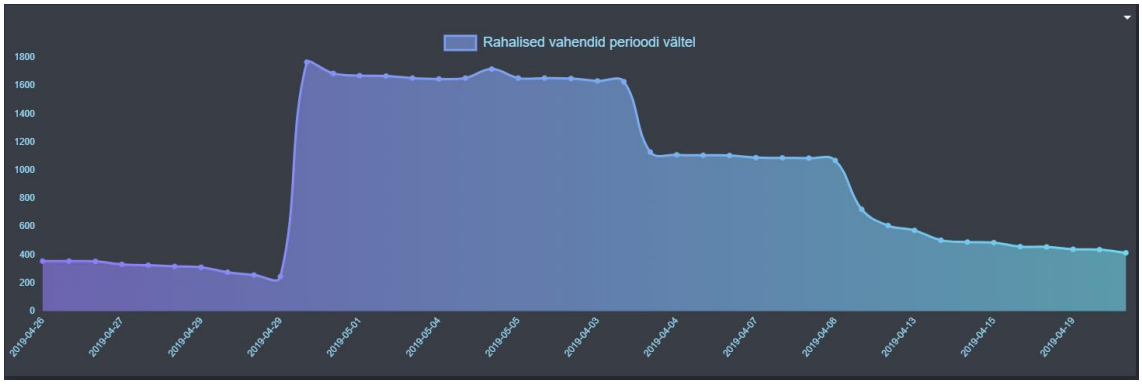
Joonis 13. Perioodi kulud

Järgmised kaks joonist kuvavad kasutajale sama perioodi kohta tema tulusi. Esimene nendest toob välja perioodi kõik tulud kategooriate kaupa ja teine toob välja kõik tehingud mis on märgitud sissetulekuna (Joonis 14). Jooniste iseloomustamiseks on kasutatud vastavalt tulpdiagrammi ja sõõrikut.



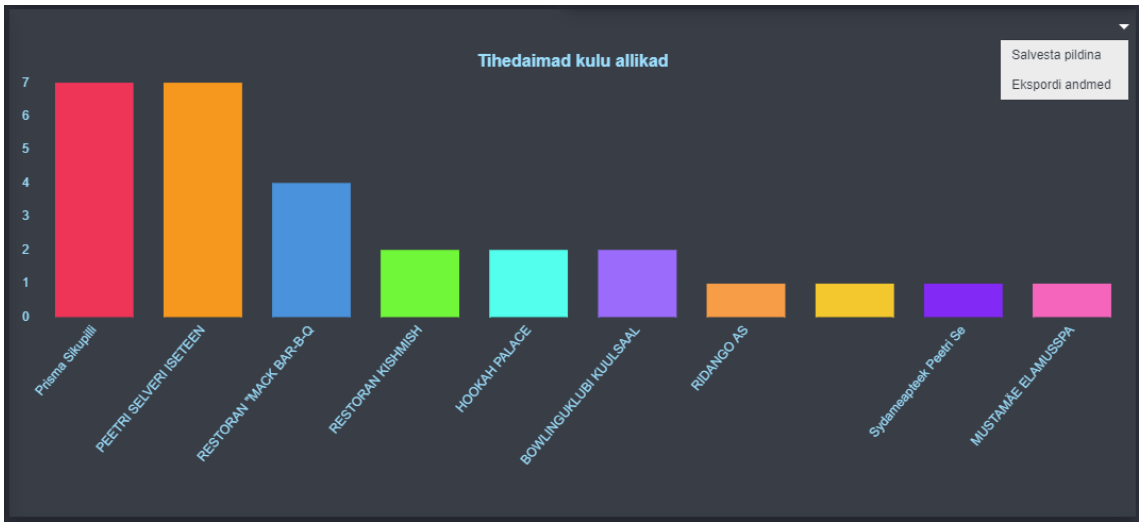
Joonis 14. Tulud

Rakenduses on välja toodud ka kasutaja rahalised vahendid perioodi vältel. Joonis on loodud joondiagrammina ning kuvab tõusude ja langustega perioodi algusest kuni perioodi lõpuni päeva kaupa kontojääki (Joonis 15).



Joonis 15. Rahalised vahendid perioodi vältel

Viimase joonisena perioodi kohta kuvab rakendus kasutajale kõige tihedasemaid kulu allikaid. Siin on kasutusel sarnaselt eelnevatega tulpdiagramm, kus tehingud jooksevad horisontaalselt ja tehingute arv on kujutatud vertikaal teljel. Samuti on võimalik kõiki rakenduse jooniseid salvestada alla pildi formaadis või eksportida lähteandmed exceli failina, et saaks neid vajadusel edasi töödelda (Joonis 16). Jooniste salvestamiseks pildi kujul või lähteandmete ekspordiks on iga joonise paremale ülesse nurka lisatund nooleke, mille peale vajutades avaneb valiku kastike kus saab valida kas allalaadida või eksportida.



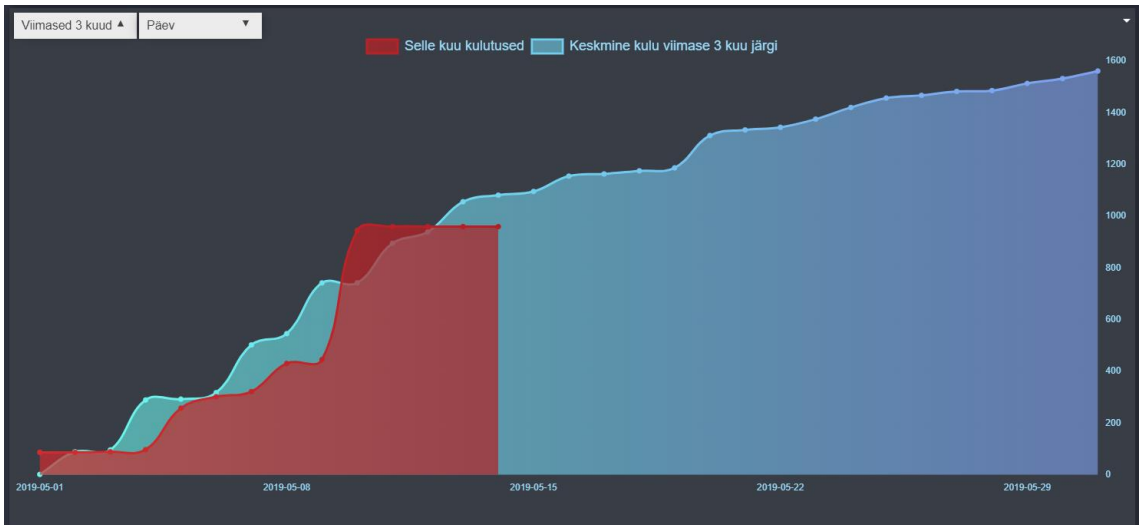
Joonis 16. Tihedaimad kulu allikad koos valikuga

5.4 Projekteerimine ja optimeerimine

5.4.1 Kulude projekteerimine

Kulude projekteerimist väljendab rakenduses joondiagramm. Joondiagrammil on kaks valikut, periood ja alus. Kulude ennustamiseks ei ole kasutatud kindlat perioodi vaid on tehtud perioodide valik, et kasutaja kulud oleks ennustatud tema hetke seisu järgi. Näiteks kui viimase poole aasta jooksul on tõusnud palk, ei ole mõtet vaadata projektsiooni viimase aasta järgi, kuna pool sellest ajast on rahalisi vahendeid vähem olnud ning see mõjutab tegelikku pilti. Perioodid on vahemikus 1 kuu kuni 12 kuud.

Aluse valikus on päev ja kuu. Kuu lõikes arvutab rakendus iga eelneva kuu kohta keskmise ja kuvab diagrammil ühte lineaarset joont. Päeva lõikes arvutab rakendus iga valitud eelneva kuu kaupa päeva lõikes keskmise kulu ja kannab selle diagrammile. Samuti on joonisel välja toodud hetke kuu kulud, et tuua välja kas kasutaja kulud on plaanipärased või mitte (Joonis 17).



Joonis 17. Kulude projektsioon

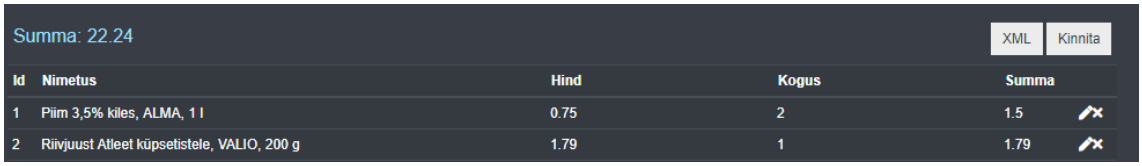
5.4.2 Kulude optimeerimine

Kulude optimeerimine arvestab viimase 30 päeva jooksul tehtud tehinguid selveri või prisma toidukauplustes. Optimeerimiseks peab kasutaja lisama tehingutele detailandmed. Selleks, et kasutaja saaks lisada ostutšeki andmeid tehingutele on rakenduses 2 võimalust. Esiteks saab iga kasutaja valida tehingute muutmise saki alt vastava tehingu ning sellele tehingule juurde lisada andmeid. Rakendus tuvastab ise ära kumma toidupoega on tegu, ning pakub vastavad kategooriad ja alakategooriad selle poe järgi.

Teine võimalus on kasutada rakenduse liidestust digitaliseeritud ostutšekkidega, mis võimaldab endasse sisse lugeda digitaliseeritud ostutšeki XML-i (Joonis 18). Selleks on rakendusse lisatud tehingute muutmise saki alla XML lisamise nupp (Joonis 19). Kui ostutšekil olevad toodete nimetused ei vasta sellega mis on toidupoe veebilehel siis proovib rakendus leida kõige lähema vaste teatud vea protsendiga, kui see ei õnnestu, teavitab rakendus kasutajat nende toodete listiga, mida rakendus ei suutnud tuvastada. Seejärel saab kasutaja need käsitsi lisada.

```
<InvoiceParties>
	<SellerParty>
		<Name>Selver AS</Name>
		<RegNumber>10379733</RegNumber>
	</SellerParty>
</InvoiceParties>
<InvoiceItem>
	<InvoiceItemGroup>
		<ItemEntry>
			<Description>Piim 3,5% kiles</Description>
			<ItemSum>0.75</ItemSum>
			<ItemQuantity>2</ItemQuantity>
			<ItemTotal>1.5</ItemTotal>
		</ItemEntry>
```

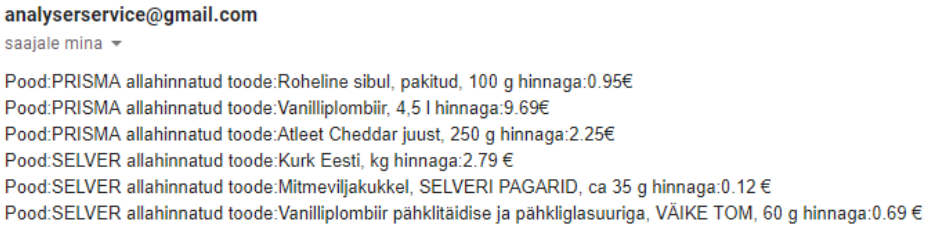
Joonis 18. Jupp digitaliseeritud ostutšeki xml-st



Joonis 19. Tehingu detailandmete lisamine

Kui andmed on lisatud kuvab rakendus neid tooteid mida kasutaja on viimase 30 päeva jooksul ostnud ning mis on tavalisest odavama hinnaga. Rakendus kuvab poodi, toote nimetust, seda hinda millega kasutaja selle soetas endale, seda hinda mis on odavam kui see millega kasutaja selle soetas ning olemasolu korral ka toote soodus hinda.

Kasutajatele on võimalik ka meilitsi teavitused saata nende toodete kohta mis on temale suunatud. Meilis on kirjas toidupoe nimetus, toote nimetus, hind ja olemasolu korral ka soodus hind (Joonis 20).



Joonis 20. Meiliteenuse poolt saadetud email

6 Tulemus

6.1 Tekkinud probleemid

Arendustöö käigus tulid välja mõned sellised probleemid, mille lahendades oleks rakenduse kasutegur suurem. Esimene probleem mis tekkis, oli see, et toodete nimetused üle poodide ei olnud omavahel samad. See probleem kaotas võimaluse võrrelda toodete hindu poodide vahel mis annaks kasutajatele väga väärtuslikku infot. Selle lahendamiseks oleks vajalik lisada juurde toodetele kas tootekood või triipkood, et saaks poodide vahel võrrelda samade toodete hindu.

Järgmine probleem mis tekkis, oli see, et ostutšekil olevad nimetused ei vastanud sellega mis on internetis poodidel. See tekitas probleemi, kuna ei saanud kohe vastavusse panna ostutšekki ja poe toodet. Probleemi leevendamiseks võeti kasutuse Fuse js teek, mis leidis lähima vaste toote nimetusele kui puudus täpne vaste. Probleemi täielikuks lahendamiseks oleks vajalik ostutšekil oleva toote nimetuse vastavusse viia sellega mis on veebilehel, või lisada mingid unikaalsed võtmed, näiteks tootekood või triipkood ostutšekile ja veebilehele.

Samuti tuli välja see, et mõningatel poodidel puuduvad tooted internetis täielikult, mõnel esinevad ainult sooduspakkumisega tooted ning mõnel on ainult sooduspakkumiste pildid, mis tähendab, et rakendusel ei ole võimalik seda infot sealt kätte saada. Seetõttu võeti rakenduse skoobi alla hetkel ainult 2 toidupoodi.

6.2 Võimalikud arengusuunad

Antud rakendus sobiks aluseks sarnaste rakenduste loomiseks. Süsteem suudab soovitud määral ennustada kasutaja kulutusi ning suudab välja tuua need tooted mille pealt on võimalik säästa.

Rakendusele saaks kulude optimeerimiseks tuua erineva kategooriaga tehinguid juurde. Näiteks sobiks selleks hästi kütus, kuna kütus on see mida tarbitakse regulaarselt. Raskem on tuua välja selliseid tooteid mida soetatakse harva näiteks elektroonika.

Samuti oleks võimalik rakendust täiendada nii joonistega, nende genereerimise võimalusega ja kasutajamugavuse täiendamisega. Hetkel on kõik joonised eeldefineeritud, kuid kui luua võimalus kasutajal ise luua joonis, siis kindlasti annaks see rakendusele juurde.

7 Kokkuvõte

Antud bakalaureusetöö eesmärkideks oli luua selline rakendus, mis suudab konto väljavõtte alusel genereerida kasutajale informatiivseid graafikuid abistamaks tal hallata oma rahalisi vahendeid. Samuti oli eesmärkideks mingil määral ennustada kasutaja igakuist kulu ning tuua välja võimalusi raha säästmiseks.

Eesmärkide täitmiseks analüüsis töö autor erinevaid tehnoloogiaid kuidas antud rakendust luua ning valis välja tema jaoks parimad tehnoloogiad. Realiseeriti rakendus koos taustateenuste ja andmebaasiga. Veebirakenduses on olemas funktsionaalsus jooniste genereerimiseks, kulude projekteerimiseks ning kulude optimeerimiseks.

Seega võib eesmärkide püstituse lugeda täidetuks. Kindlasti on rakendusel arenemise ruumi, eriti just selle koha pealt, et oleks võimalik erinevate poodide tooteid võrrelda täpselt.

Kasutatud kirjandus

[1] "What is JavaScript?" [Võrgumaterjal] Available: https://www.w3schools.com/whatis/whatis_js.asp. [Kasutatud 20.04.2019]

[2] "What is jQuery?" [Võrgumaterjal] https://www.w3schools.com/whatis/whatis_jquery.asp. [Kasutatud 20.04.2019]

[3] "jQuery - AJAX Introduction" [Võrgumaterjal] https://www.w3schools.com/jquery/jquery_ajax_intro.asp. [Kasutatud 21.04.2019]

[4] "SheetJS" [Võrgumaterjal] https://sheetjs.com/. [Kasutatud 20.04.2019]

[5] "Chart.js" [Võrgumaterjal] https://www.chartjs.org/. [Kasutatud 27.04.2019]

[6] "html2canvas" [Võrgumaterjal] https://html2canvas.hertzen.com/. [Kasutatud 08.05.2019]

[7] "Fuse.js. Lightweight fuzzy-search library" [Võrgumaterjal] https://fusejs.io/. [Kasutatud 11.05.2019]

[8] "What is Bootstrap?" [Võrgumaterjal] https://www.w3schools.com/whatis/whatis_bootstrap.asp. [Kasutatud 20.04.2019]

[9] "CSS Grid Layout Module" [Võrgumaterjal] https://www.w3schools.com/css/css_grid.asp. [Kasutatud 22.04.2019]

[10] "MySQL by Examples for Beginners" [Võrgumaterjal] https://www3.ntu.edu.sg/home/ehchua/programming/sql/MySQL_Beginner.html. [Kasutatud 19.04.2019]

[11] "What is PHP?" [Võrgumaterjal] https://www.php.net/manual/en/intro-whatis.php. [Kasutatud 19.04.2019]

[12] "Node.js Introduction" [Võrgumaterjal] https://www.w3schools.com/nodejs/nodejs_intro.asp. [Kasutatud 02.05.2019]

[13] "Cheerio" [Võrgumaterjal] https://cheerio.js.org/. [Kasutatud 02.05.2019]

[14] "Nightmare. A high-level browser automation library." [Võrgumaterjal] http://www.nightmarejs.org/. [Kasutatud 02.05.2019]

[15] "The Nodemailer Module" [Võrgumaterjal] https://www.w3schools.com/nodejs/nodejs_email.asp. [Kasutatud 12.05.2019]

[16] "Web Scraping with Node.js" [Võrgumaterjal] https://stackabuse.com/web-scraping-with-node-js/. [Kasutatud 02.05.2019]

[17] "SQL Injection" [Võrgumaterjal] https://www.w3schools.com/sql/sql_injection.asp. [Kasutatud 13.05.2019]

[18] Keir Volas, 2017. "Ostutšekkide digitaliseerimine". TalTech, Magistritöö. [Kasutatud 11.05.2019]

[19] "Introduction to XML" [Võrgumaterjal] https://www.w3schools.com/xml/xml_whatis.asp. [Kasutatud 11.05.2019]

[20] "What is React?" [Võrgumaterjal]

https://www.w3schools.com/whatis/whatis_react.asp. [Kasutatud 13.05.2019]
[21] "What is AngularJS?" [Võrgumaterjal]
https://www.w3schools.com/whatis/whatis_angularjs.asp. [Kasutatud 13.05.2019]
[22] "JavaScrpt HTML DOM" [Võrgumaterjal]
https://www.w3schools.com/js/js_htmldom.asp. [Kasutatud 07.05.2019]
[23] "Electron" [Võrgumaterjal] https://electronjs.org/. [Kasutatud 02.05.2019]