TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Pärtel Relve 193668IADB

Eraisiku kulude haldamise rakenduse loomine Blazor raamistikus

Bakalaureusetöö

Juhendaja: Lembit Viilup
PhD

Tallinn 2022

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: Pärtel Relve

16.05.2022

Annotatsioon

Bakalaureusetöö eesmärgiks on luua eraisiku kulude haldamise rakenduse prototüüp ja luua selle käigus kokkuvõtlik ülevaade teekidest, mida saab kasutada Blazor raamistikus diagrammide kuvamiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 49 leheküljel, 5 peatükki, 28 joonist, 3 tabelit.

Abstract

Developing a Personal Finance Application Using the Blazor Framework

The aim of the thesis is to develop a prototype of a personal finance application using the Blazor framework and to produce an analytical overview of the libraries available for displaying charts within the Blazor framework.

The thesis is in Estonian and contains 49 pages of text, 5 chapters, 28 figures, 3 tables.

Lühendite ja mõistete sõnastik

.cs fail	C# keeleruumi lähtekoodi failiformaat.
.razor fail	Blazor raamistikus ja mujal .NET tehnoloogial põhinevates veebirakendustes kasutatav failiformaat, mis põimib omavahel C# ja HTML keeli veebilehtede kuvamiseks.
Css	*Cascading Style Sheets* on veebilehte kujundamiseks kasutatav märgistuskeel, millega määratakse muuhulgas näiteks kuvatavate elementide suuruseid ja värve.
Blazor	C# keeleruumi raamistik veebirakenduste valmistamiseks.
JSON Web Token	Internetistandard, mis sätestab põhimõtted veebirakenduse kasutaja autentimiseks ja autoriseerimiseks.
Komponent	Blazor raamistiku kontekstis tähendab komponent veebilehe korduvat osa – näiteks nuppu või diagrammi – mis moodustab omaette kooditerviku. Tavaliselt on komponendil parameetrid, mis täpsustavad selle sisu (nt nupu tekst või diagrammi andmepunktid).
MIT litsents	Tarkvara kasutamise litsents, mille põhimõte on, et teegi kasutamine ja muutmine on lubatud nii era- kui ärikasutuseks, kuid tuleb säilitada algne autoriõiguse teavitus [1].
NuGet pakett	C#/.NET keeleruumis kasutatav standardiseeritud pakett projektile teekide lisamiseks.
Transpiling	Protsess, kus üks programmeerimiskeel teisendatakse ümber teiseks programmeerimiskeeleks. Erinevalt näiteks *cross-compilation* protsessist, kus ühes süsteemis kirjutatud kood kompileeritakse teises süsteemis toimivaks programmiks, tõlgitakse *transpiling* protsessiga ühes keeles kirjutatud lähtekood teises keeles toimivaks lähtekoodiks.
Võlusõne	Inglise keeles *magic string*. Programmeerimises konstantse nimetuse edasi andmine vabal kujul kirjutatud sõnena. Tava, mida tuleks muude valikute olemasolul vältida, sest arenduskeskkond ei kontrolli sõnede sisu ja ainus näpuviga tähendab, et programm ei tunne konstantset nimetust ära [2].
WebAssembly	WebAssembly (lühend WASM) on binaarjuhiste formaat, mis võimaldab programmide käivitamist veebibrauseris.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Kuni 19. sajandini valitses Eestis naturaalmajandus ja rahaga rehkendamine oli vajalik oskus vaid väikesearvulise eliidi jaoks [3]. Tänaseni on maailmas mitmeid rahvaid, nagu näiteks korowaid, kus tavainimestel puudub rahaga tegelemise soov ja vajadus [4].

Moodsas maailmas, sealhulgas Eestis, toimib aga kapitalistlik majandussüsteem, kus rahaga tuleb tegeleda kõikidel inimestel, sõltumata sellest kas neil on finantsmaailma vastu huvi või mitte. Oleks ennatlik arvata, et rahaga ringi käimine taandub üksnes vastutustundele. Tuleb arvesse võtta ka inimeste erinevaid huvisid ja soodumusi. Mõne jaoks on rahaasjade üle arvestuse pidamine meeldiv tegevus. Teiste jaoks on see tüütu kohustus, mille peale ei taheta kulutada energiat ega aega. Kuigi rahaga planeeritult ringi käimine suurendab nii turvalisust kui ostujõudu, pole see teadmine siiski kõikide inimeste jaoks piisav, et rahaplaneerimist käsile võtta.

Käesoleva töö raames valmistatakse veebirakendus, mille kasutusmugavus ja praktilised soovitused aitavad hakata kulusid planeerima inimestel, kellel pole soovi või tahtejõudu oma raha haldamisega sügavuti tegeleda.

Veebirakenduse valmistamiseks kasutatakse C# keeleruumi raamistikku nimega Blazor. Blazor on selle töö kirjutamise ajal uudne raamistik, mille üks eripärasid on, et see võimaldab ühelehe veebirakenduse käitamist kasutaja veebibrauseris ilma, et tuleks programmeerimisel kasutada JavaScripti keelt. Kuna Blazor raamistik on uus, pole programmeerijatel veel välja kujunenud ühtseid tavasid või vastuseid mitmetele raamistiku kasutamisel ettetulevatele küsimustele. Üks sellistest küsimustest on, et kuidas kuvada Blazor raamistikus valmistatud rakendustes diagramme.

Lisaks rakenduse prototüübile valmib töö käigus programmeerijatele suunatud ülevaade erinevatest teekidest, millega on võimalik Blazor raamistikus diagramme kuvada. Tutvustatakse kõiki levinud diagrammiteeke. Võrreldakse nende eeliseid ja puuduseid, eelkõige antud rakenduse loomise kontekstis.

1.1 Ülesande püstitus

Lõputöö käigus valmiva rakenduse sihtgrupiks on kõik Eestis pangakontot kasutavad inimesed, kes pole leidnud piisavalt lihtsat meetodit oma kulude kontrolli all hoidmiseks ja raha kogumiseks.

Rakendus aitab kasutajal igapäevakulutusi optimeerida. Lähteinformatsioonina kasutatakse kasutaja sissetulekut ja kasutaja valikute alusel paika pandud tarbimiseelistusi ja kogumisvajadusi. Rakendus arvutab lähteandmete põhjal inimesele eelarve kulukategooriate lõikes. Kasutusharjumuse tekkimiseks peab rakenduse kasutamine olema võimalikult lihtne ja võtma võimalikult vähe aega, kuid andma seejuures selget praktilist kasu.

Rakendus hoiatab kasutajat, kui tema püsikulud on sissetuleku kohta liiga kõrged. Lisaks joonistab rakendus varasemate eelarvete põhjal diagrammi, mis aitab kasutajal näha oma finantsolukorra halvenemist või paranemist läbi aja koos püsikulude mõjuga muudele rahalistele võimalustele. Rakendus võimaldab silla loomist ühe leibkonna liikmete vahel. Ühendatud kasutajad saavad ühtlustada püsikulude koormust ja näha teineteise panust ühisesse muutuvkulude eelarvesse. Rakenduse hoiatused ja soovitused põhinevad finantsasjatundjate seisukohtadel.

Lõputöö käigus valmiva Blazor diagrammiteekide võrdluse sihtgrupiks on kõik programmeerijad, kes soovivad Blazor raamistikus programmeerida rakendust, kus kuvatakse diagramme.

Enne võrdluse alustamist tutvustatakse Blazor raamistikku ja üheleherakenduse kontseptsiooni. Seejärel antakse ülevaade diagrammide kuvamise viisidest selles raamistikus ja teekidest, mis seda võimaldavad. Omavahelisse võrdlusesse võetakse kõik Blazor raamistiku laiemalt levinud diagrammiteegid. Vaadeldakse teekide tööpõhimõtet, dokumentatsiooni, kasutajaliidest, hinda, esteetikat ja katsetuste käigus ilmnevaid tugevusi ja puudusi.

Teekide analüüsi põhjal valitakse nende seast antud rakenduse valmistamiseks kõige sobivam. Lõpetuseks kirjeldatakse rakenduse toimimise tehnilisi põhimõtteid ja struktuuri.

1.2 Metoodika ja struktuur

Töö sissejuhatusele järgnevas osas antakse statistikal põhinev ülevaade Eesti inimese kulutusharjumustest. Et rakendus saaks anda pädevaid hoiatusi ja soovitusi, tutvutakse finantsasjatundjate seisukohtadega raha jaotamise küsimustes. Uuritakse, millised tunnustatud põhimõtted kehtivad eraisikute kulude jaotusele üldiste kategooriate lõikes. Püstitatakse tingimused rakendusele, mis aitaks neid põhimõtteid järgida.

Töö järgmises osas tutvustatakse Blazor raamistikku. Et Blazor raamistikus loodud rakenduses saaks näidata eelarvete ajalugu diagrammina, viiakse läbi võrdlus raamistikule loodud diagrammiteekide vahel. Selleks kasutatakse nii teekide dokumentatsiooni kui praktilisi katseid. Võrdluse käigus vaadeldakse teekide kõiki olulisi aspekte, nii tugevuste kui nõrkuste osas.

Sellele järgnevas osas kirjeldatakse valminud rakenduse arhitektuuri ja funktsionaalsust. Tutvustatakse rakenduse tulevikku puudutavaid plaane.

Arendustöö viiakse läbi üksikarendaja jaoks kohandatud agiilsete töövõtetega.

2 Eraisiku kuluhaldus

Kõik inimesed jaotavad mingil viisil oma kulusid, ükskõik kas nad teevad seda teadlikult ja planeeritult või alateadlikult ja hetketujudest lähtuvalt. Selleks, et luua rakendus, mis aitab eraisikutel kulusid tõhusamalt hallata, on esmalt vaja ülevaadet hetkeolukorrast ja senistest lahendustest.

2.1 Eraisikute kulutamisharjumused

Kulude ette planeerimine ja jälgimine annab inimesele mitmeid praktilisi eeliseid. Kitsikuse korral annab see teadmise, kas probleem on ajutine ja tuleneb halbade juhuste kokkulangevusest või hoopis pidev ja tuleneb üle oma võimete elamisest. Ühtlasi teeb see elu turvalisemaks, tagades rahalise puhvri juhuks kui õnnetuse tagajärjel tekib ootamatu varaline kahju või sissetuleku kadu. Samuti suurendab see inimese ostujõudu, võimaldades suurteks kulutusteks adekvaatselt ette valmistuda. Järelmaksuga ostmise asemel on kulude planeerimise abil võimalik raha ette koguda ja soetada soovitud ese intressivabalt tavaostuna.

Rahandusministeeriumi andmetel rakendavad aruka finantskäitumise aluspõhimõtteid ainult 58,8% Eesti elanikest. Seejuures ei pane 31% oma palgast mitte midagi teadlikult kõrvale ning peaaegu sama palju, 28%, on neid, kes kogu oma raha enne uut palgapäeva ära kulutavad [5]. Pere eelarve üle peavad teadlikult arvestust kõigest 38% leibkondadest. Nende leibkondade hulgas, kes arvestust peavad, nimetavad 66% oma abivahendiks lihtsalt jooksva kontoseisu jälgimist pangas [6]. Pangarakendust või muud spetsiaalset rahajuhtimise vahendit kasutab kulude planeerimiseks 14% Eesti elanikest. Peaaegu kolmandik (32%) on pidevas mures oma arvete tasumise pärast [7].

Anastasia Shavrina toob Tallinna Tehnikaülikooli bakalaureusetöös välja, kuidas enam kui pooled Eesti elanikest kulutavad toodete ja meelelahutuse peale rohkem raha, kui nad tegelikult sooviksid. Peamiste põhjustena märgivad nad seejuures mitteteadlikke harjumusi ja impulssoste [8]. Teine Tallinna Tehnikaülikooli vilistlane, Gerda Karman, näitab oma bakalaureusetöös, et kuigi enamik üliõpilasi peavad toime tulema vähemaga

kui 500€ kuus (ja nendest omakorda rohkem kui pooled vähemaga kui 300€ kuus), tegelevad väljaminekute eelarvestamisega vaid 25% üliõpilastest [9].

2.2 Arukas kulude jaotus

Inimestel tuleb kulutamine välja loomulikumalt kui säästmine. Seetõttu tuleb säästmisele suunatud tegevused asetada finantsplaanis kesksele kohale. Kulude kaardistamise käigus tulevad sageli välja ebavajalikud väljaminekud, millest loobumine pole inimese jaoks keeruline [10].

Kulude jagamisel on oluline teha vahet püsikuludel ja muutuvkuludel. Püsikulud on perioodilised kulud, mille suurus on enam-vähem ette teada ja mis enamasti tulenevad lepingutest või kohustustest. Näiteks lähevad siia alla liising, üür, elektriarve, sidekulud, liikmemaksud, püsitellimused ja laenude tagasimaksed. Muutuvkulud sõltuvad seevastu otseselt inimese käitumisest ja ei ole ette määratud. Näiteks lähevad muutuvkulude alla toidukulud, majapidamiskulud, kingitused, riiete või elektroonika ostmine ja väljas käimine. Inimese igapäevastel otsustel on muutuvkulude suuruse üle vahetu mõju, sest ta saab pidevalt otsustada, kui kalli toidu ta valib, kui tihti käib väljas söömas või pidutsemas ning kas eelistab uhiuusi brändirõivaid või midagi soodsamat [11].

Isikliku finantsplaani kõige tähtsamate eesmärkide hulka peaks kuuluma hädareservi moodustamine. See on varuraha, mida saab kasutada ootamatute ja vältimatute vajaduste korral, nagu näiteks tasuline arstiabi, sissetuleku kaotus või purustused kodule või autole. Kuigi mõningal määral on selliseid riske võimalik maandada kindlustuste kaudu, ei ole kunagi võimalik kindlustada end kõikide mõeldavate õnnetuste vastu [10].

Kui suur peaks hädareserv olema? Vastus sõltub inimese sissetulekust, aga ka sissetuleku stabiilsusest ja sotsiaalsest turvavõrgust, ehk kui palju on inimesel lähedasi, kes teda häda korral rahaliselt aitaksid [10].

Hädareservi vajalik suurus nendest tingimustest lähtuvalt on välja toodud Tabelis 1.

Tabel 1. Finantsinspektsiooni soovitatud hädareservi suurus [10].

Hädareservi suurus mõõdetuna kuudes, mille jooksul tekkivad kulud peab saama reservist katta	**Sissetuleku stabiilsus ja sotsiaalne turvavõrk**
3 kuud	Põhisissetulek on kindel ja stabiilse suurusega ning lähedased on valmis ja võimelised häda korral rahaga aitama.
6 kuud	Põhisissetulek on ebakindel või perioodiliti kõikuv ning lähedastel ei pruugi olla valmisolekut või võimekust häda korral rahaga aidata.
9-12 kuud	Põhisissetulek on kadumisohus või ettearvamatult kõikuv ning lähedastel ei ole valmisolekut või võimekust häda korral rahaga aidata.

Üldiselt võiks inimese vältimatud elamiskulud nagu toit, riided, hügieenitarbed, transport jm moodustada 30-60% tema kuludest. Isiklikud lisakulud, nagu näiteks meelelahutus, mugavuskaubad, enesetäiendus jm võiksid võtta 15-30% suuruse osa. Kõik ülejäänu, ehk 10-55%, võiks minna hädareservi täitmiseks ja säästudeks. Säästude mõte pole siinkohal niisama raha koguda, vaid tekitada vahendid, millega sooritada suuremaid tehinguid, näiteks investeerimiseks, laenu enneaegseks tagastuseks, remontimiseks, reisimiseks või kinnisvara sissemakseks [11].

Sarnased vahemikud kehtivad ka ülemaailmselt tunnustatud Elizabeth Warreni eraisiku kulude jaotamise meetodis, mille järgi on mõistlik suunata oma kuludest kuni 50% vältimatute vajaduste katteks, kuni 30% isiklike huvide katteks ja ülejäänu, ehk minimaalselt 20% säästmiseks ja investeeringuteks [12].

Nimetatud jaotusele lisaks peaks jälgima, et laenude tagasimaksed ei kasvaks kunagi suuremaks kui 40% sissetulekust – see on ülemine piir, mida enamik pankasid hindab laenaja maksimaalseks tagasimaksevõimeks [11].

2.3 Rakenduse roll kuluhalduses

Tuntud investor Jaak Roosaare toob „Rikkaks saamise õpikus“ välja, et eraisiku kulude haldamise kõige raskem koht on planeeritud eelarvest kinni pidamine: „Alati saabub ootamatult mõni sünnipäevakutse või on vaja osta mingi uus riideese… Lisaks nõuab

eelarvest kinni pidamine tõsist tahtejõudu ja eeldab ajakulu nii planeerimiseks kui ka selle järgimiseks. Suuremale osale meist on see liig, mis liig“ [13].

Peaaegu kolmveerand Eesti inimestest teab, kuidas ja miks peaks raha säästma, kuid suur osa elanikest ei tule sellega praktikas toime [5]. Seda hoolimata asjaolust, et kulude jälgimine on eestlase jaoks võrdlemisi mugav. Lisaks alati käepärast olevatele e-panga väljavõtetele on saadaval mitmeid kodumaiseid rakendusi, mis aitavad kulutuste kohta statistikat koostada. Sellistest rakendustest on Caroliina Laantee andnud põhjaliku ülevaate Tallinna Tehnikaülikooli bakalaureusetöös “Rahaplaneerimisrakenduste kasutajaliidesed ja kasutatavus” [14]. Samuti on tasuta saadaval mitmeid kvaliteetseid rahvusvahelisi kulude jälgimise mobiilirakendusi [15].

Neile, kes tahavad eelarvestamisega sügavuti tegeleda, leidub seega lai valik lahendusi alates spetsiifilistest kulude jälgimise rakendustest kuni üldiste arvutustabeliteni nagu Microsoft Excel ja selle analoogid. Puudub aga rakendus, mis oleks mõeldud neile, keda eelarvestamine ei huvita, kuid kes mõistavad selle vajalikkust ja oleks valmis sellega tegelema, kui see nõuaks piisavalt vähe vaeva ja annaks piisavalt selget praktilist kasu.

Võib järeldada, et rakendus, mille järele on täitmata nõudlus, peab vastama kahele tingimusele. Esiteks peab see võimaldama isikliku eelarve koostamist võimalikult vähese ajakuluga. Teiseks peab see aitama koostatud eelarvest rangelt kinni pidada.

Nende tingimuste tõttu võib välistada võimaluse, et käesolevas töös loodav rakendus hakkaks põhinema tšekkide või üksiktehingute sisestamisel ja kategoriseerimisel. Selline tegevus on ajakulukas. Pealegi on juba küllaldane valik rakendusi, mis sellist arvepidamist võimaldavad.

Tingimustele vastava lahenduse loomiseks saab ära kasutada asjaolu, et kõigis suuremates Eesti pankades on võimalik avada tasuta lisakontosid ja sõlmida tasuta igapäevaseid püsikorraldusi [16], [17], [18]. Tänu sellele oleks võimalik saada praktilist kasu rakendusest, mis arvutab kasutaja muutuvkulude jaoks mõistliku suurusega isikliku päevaeelarve. Kasutaja saab seejärel avada eraldi konto raha kulutamiseks ja sõlmida püsikorralduse kulutuskontole, mille alusel laekub sinna iga päev päevaeelarve jagu raha.

Selline meetod tagab, et kasutaja ei saa kogemata või kiiresti mööduvast impulsist lähtuvalt oma eelarvet ületada. Tema kulutuskontoga seotud pangakaardil pole kunagi

rohkem vahendeid kui eelarve ette näeb. Kui kasutaja tahab sooritada ostu, mis ületab päevaeelarvet, tuleb tal selleks oodata, kuni kulutuskontole on kogunenud piisav summa.

Eelarve ületamine on sellise süsteemiga muudetud piisavalt ebamugavaks, et kiusatust vähendada. Eelarve rikkumine on võimalik vaid sihiliku tegevusena muuks eesmärgiks mõeldud kontolt eraldi ülekandega raha kulutuskontole liigutades.

Kiire harjumuspärase teo blokeerimine ja selle asendamine teadvuse täit osalust nõudva aeglasema tegevusprotsessiga vähendab võimalust, et inimene halvale harjumusele järele annab [19].

Samal ajal vabastab selline lahendus kasutaja pideva säästmise stressist, sest kulutuskontol olev raha pole mõeldud säästmiseks ega investeerimiseks ja seda võib kulutada ilma süümepiinadeta.

2.4 Eelarve koostamise protsess

Eestis on tavaks maksta palka kalendrikuu kaupa. Valdav enamik Eesti inimesi mõtleb oma tuludest ja kuludest samuti kuude lõikes [7]. Tagamaks võimalikult lihtsat kasutajakogemust, kasutatakse ka loodavas rakenduses kuupõhist kuluarvestust.

Rakendus eeldab, et kasutaja avab vähemalt viis erinevat (tasuta) arvelduskontot – vältimatute vajaduste kulutuskonto, isiklike huvide kulutuskonto, fikseeritud maksete konto, kogumiskonto ja hädareservi konto. Kontosid võib olla veelgi rohkem kui kasutaja soovib erinevate eesmärkide jaoks kogutavat raha hoida eraldatuna. Näiteks võib ühe suure kogumiskonto asemel avada eraldi reisikonto, remondikonto ja investeerimiskonto.

Viis minimaalset kontot koos kirjeldustega on välja toodud Tabelis 2.

Tabel 2. Viis minimaalset arvelduskontot loodava rakenduse kasutamiseks.

Konto nimetus	**Konto eesmärk**
Vältimatute vajaduste kulutuskonto	Esimene kahest kulutuskontost. Sellel kontol olev raha on mõeldud jooksvate elamisvajaduste katmiseks. Nende kulude alla lähevad väljaminekud, mida pole enamasti võimalik ära jätta ega edasi lükata. Näiteks kuuluvad siia toit, hügieenitarbed, ravimid ja kütus.
Isiklike huvide kulutuskonto	Teine kahest kulutuskontost. Sellel kontol olev raha on mõeldud isiklike huvitegevuste ja naudingute jaoks. Nende kulude alla lähevad väljaminekud, mida on enamasti võimalik ära jätta või edasi lükata. Näiteks kuuluvad siia väljas käimised, koju pitsa tellimine, raamatud, tasulised kursused, uued riided või elektroonilised vidinad.
Püsimaksete konto	Konto, kuhu eraldatakse raha püsikulude maksmiseks olukorras, kus makset pole võimalik kohe teha, sest kuluga seonduvat arvet pole veel laekunud.
Kogumiskonto	Konto, kuhu kogutakse raha suuremate finantstegude jaoks, näiteks reisimiseks, remondiks või investeerimiseks. Kogumiskontosid võib olla rohkem, näiteks eraldi reisimise ja investeerimise jaoks.
Hädareservi konto	Konto, millel olevat raha kasutatakse ootamatute õnnetustega kaasnevate kulude korral.

Kasutajal peaks olema pangakaart mõlema kulutuskonto jaoks. Isiklike huvide kulutuskonto arvelt võib vajadusel katta ka vältimatute vajadustega seonduvaid kulusid, aga vastupidist ristkasutust tuleks vältida. Teiste kontodega ei tohiks ühtegi pangakaarti siduda, sest neid kontosid ei tohiks ostude tegemiseks kasutada.

Kasutaja sissetulek ei tohiks laekuda kummalegi kulutuskontole. Nii väheneb võimalus eelarveväliste kulutuste tegemiseks valearvestuse või hetkeemotsiooni tõttu. Sissetulek peaks laekuma kogumiskontole või eraldiseisvale sissetulekukontole.

Eelarve arvutamiseks tuleb kasutajal sisestada rakendusse enda sissetulekud. Lihtsuse huvides ei saa tulusid ühe arvestusperioodi keskel juurde lisada. Ootamatute lisatulude tekkimisel pannakse lisatulud kõrvale kuni tuleb aeg järgmise kuu eelarvestamiseks. Kui kasutaja soovib kasutada ühekordset suuremat sissetulekut mitme kuu väljaminekute katmiseks, tuleb tal sissetulek ise vastavalt nii mitme kuu eelarvete vahel ära jagada.

Piiratud päevaeelarve idee on mõttekas ainult muutuvkuludele kontekstis. Päevaeelarvega ei saa piirata näiteks üüri või elektriarve tasumist. Seega tuleb enne päevaeelarve arvutamist kanda rakendusse sisse kuu jooksul tasumist vajavad püsikulud.

Püsimaksete puhul, mille arveldusperiood on pikem kui üks kuu – näiteks liikluskindlustusmakseid tehakse sageli kord aastas – arvutatakse rakenduse poolt ümber igakuiseks osamakseks, mida kasutaja kogub püsimaksete kontole kuni tuleb aeg arve tasumiseks. See aitab vähendada püsikulude kõikumist. Stabiilsemad kulud võimaldavad kasutajal kergemini enda rahaasju planeerida.

Püsikulude hulka loetakse siinkohal ka muud ette teadaolevad regulaarse suuruse ja kindla intervalli järel korduvad kulud, mis ületavad märkimisväärselt tüüpilist päevaeelarvet. Näiteks kui kasutajal on harjumus käia igal teisel kuul juuksuri või kosmeetiku juures, tuleks need arvestada püsikuludeks.

Kui püsikulud kasvavad tuludega võrreldes ebamõistlikult suureks, annab rakendus selle kohta hoiatuse. Kasutaja peaks sel juhul oma püsikulusid kriitiliselt hindama ja otsima kärpimiskohti. Võib-olla on tal võimalik valida näiteks soodsam sidepakett, loobuda mõnest püsitellimusest või alustada odavama üürihinnaga elukoha otsinguid.

Pärast püsikulude maha arvestamist teeb rakendus allesjäänud sissetulekujäägi pealt arvutuse, mis näitab:

- millise summaga igapäevase püsikorralduse peaks kasutaja tegema üheks kuuks vältimatute vajaduste kulutuskontole,
- millise summaga igapäevase püsikorralduse peaks kasutaja tegema üheks kuuks isiklike huvide kulutuskontole,
- millise summaga ühekordse makse peaks kasutaja kandma hädareservi ja/või kogumiskonto(de)le.

Abielupaaridel või muul viisil ühiselt leibkonna eelarvesse panustavatel kasutajatel on otstarbekas kasutada ühte ja sama vältimatute vajaduste kulutuskontot, millele mõlemad raha lisavad ja millele mõlemad võrdset ligipääsu omavad, sest vältimatutest kuludest enamiku moodustavad tavaliselt leibkonna vajadused.

Rakendus võimaldab sellistel isikutel siduda oma profiilid. Rakendus annab seotud profiilide puhul hoiatuse, kui üks eelarvesse panustajatest on püsimaksetega koormatud palju raskemalt kui teine, kusjuures koormus arvutatakse protsendina kasutaja isiklikust sissetulekust. Sellise hoiatuse tekkimisel võiksid kasutajad otsida võimalust püsikulude tasakaalustamiseks – näiteks muuta, kes maksab elektriarveid või kes katab kui suurt osa üürist või kodulaenumaksetest.

Kirjeldatud lahendus võimaldab kasutajal hallata oma kulusid tõhusal viisil, ilma et ta peaks seejuures igapäevaselt kulusid märkima või muul viisil pidevalt raha üle teadlikku arvestust pidama. Rakenduse kasutamine nõuab kasutajalt vaid ühte umbes veerand tunni pikkust kasutuskorda kuu jooksul. Esmakordsel kasutamisel võib kasutusaeg olla pikem seoses rakenduse tundma õppimisega ja vajadusega avada internetipangas mõned lisakontod.

Sellise süsteemiga toimiv rakendus täidab eelnevalt seatud eesmärgi anda kasutajale võimalikult vähese ajapanusega selgesti järgitav eelarve, millest on lihtne kinni pidada.

2.5 Kulujaotuse diagrammid

Kasutajad saavad kuueelarveid säilitada ja vaadelda oma eelarvete ajalugu diagrammil, kus antakse ülevaade rahajaotusest püsikulude, mõlemat liiki muutuvkulude ja kogumiseks eraldatud summade lõikes. See annab kasutajale visuaalse ülevaate finantsolukorra muutumisest läbi aja, ehk võimaldab võrrelda, kas ja millal on tema rahaline seis võtnud parema või halvema suuna. Ühtlasi aitab diagramm näha ja tõsta teadlikkust püsikulude mõjust kulujaotusele (mida väiksemad on püsikulud, seda rohkem raha jääb muutuvkuludeks ja kogumiseks).

Ühte leibkonda seotud kasutajate puhul on diagrammi peal nähtav mõlema kasutaja panus vältimatute vajaduste kulutuskontole.

3 Blazor raamistik

Blazor on uudne C# keeleruumi raamistik veebirakenduste loomiseks. Selle abil saab muude võimaluste seas valmistada ühelehe veebirakendusi [20].

3.1 Üheleherakendused

Klassikaliste veebilehtede puhul kuvatakse sisu HTML lehtedena, mida päritakse kas otse serverist või vahendatakse ja koostatakse vaheprogrammi kaudu, mis võib olla kirjutatud näiteks PHP programmeerimiskeeles. Sellistel veebilehtedel navigeerides saadab server uuesti välja kogu veebilehe igakord kui lehel on vaja uusi andmeid näidata, sõltumata sellest, kui väikese muutusega on tegemist.

Ühelehe veebirakenduse korral päritakse veebileht brauserisse ainult ühe korra, lehe külastamise alguses. Edaspidi lehel toiminguid sooritades ja uusi andmeid pärides ei laeta uuesti sisse kogu lehte, vaid uuendatakse juba sisse laetud lehel olevaid andmeid ja komponente. Tänu sellele vajab ühelehe veebirakendus võrreldes traditsioonilise veebilehega vähem andmevahetust ning pakub kiiremat ja sujuvamat kasutuskogemust. [21]. Tuntud näited ühelehe veebirakendustest on näiteks Gmail, Google Maps, Netflix ja Paypal.

Üheleherakenduste jaoks kombineeritakse sageli erinevaid programmeerimiskeeli. Brauseri poolel on enamasti kasutuses mõni JavaScriptil põhinev raamistik [16]. JavaScripti kasutatakse laialdaselt andmete kuvamiseks, kuid seda sageli seetõttu, et siiani on see olnud ainuke programmeerimiskeel, mida brauserid universaalselt toetavad. JavaScriptil on mitmeid tõsiseid puuduseid [22]. Seetõttu on tavapärane, et JavaScripti kasutatakse andmete kuvamiseks, aga andmete salvestamiseks ja töötluseks võetakse juurde mõni võimekam, töökindlam ja parema mainega keel.

Blazori abil saab üheleherakendusi kirjutada algusest lõpuni C# programmeerimiskeeles, ilma JavaScripti kasutamata. Blazor raamistik ei kasuta selleks teisendust, nagu näiteks *transipiling* protsess, mille käigus muudetaks C# peidetult ümber JavaScriptiks. Selle

asemel kasutab Blazor WebAssembly tehnoloogiat, mis käivitab C# keeles kirjutatud rakenduse otse kasutaja brauseris [20].

3.2 Diagrammiteekide võrdlus

Blazor on üheleherakenduse programmeerimiseks avanud uued võimalused. Kuna raamistik on uus, pole paljudele küsimustele ja probleemidele veel välja kujunenud tüüplahendust. Üks sellistest küsimustest on, et kuidas kuvada Blazor raamistikus valmistatud üheleherakenduses diagramme. Diagrammide kuvamiseks tuleb raamistikule liita mõne kolmanda osapoole kirjutatud teek.

Järgnevalt on selliseid teeke võrreldud. Joonistel on välja toodud iga teegiga loodud näidisdiagramm ja programmikoodi tööpõhimõte.

Võrdlusesse võeti kõik sellised Blazor diagrammi teegid, mis on mõeldud Blazor üheleherakendusele ja millele otsingumootorid Google, DuckDuckGo ja Bing viitasid otsingusõnade „Blazor charts“ vastuste seas esimese 5 lehekülje jooksul kahel või enamal korral. Kokku leidus selliseid teeke 11.

Näidisdiagrammid on loodud vastavalt teegi vaikekujundusele, soovitustele ja näidistele, nende välimust pole täiendavalt seadistatud ega muudetud. Teekides, mis seda võimaldavad, on võrdluses kasutatud aladiagrammi. Kui teek ei võimaldanud aladiagrammi kuvamist (kas põhimõtteliselt või mõistliku pingutuse piires), kasutati joondiagrammi.

Näidisdiagrammidele andmestikku luues ja programmikoodi tööpõhimõtet demonstreerides on lähtutud rakenduse reaalsest vajadusest kuvada kasutaja kulude jaotuse ajalugu.

3.2.1 Ant Design Blazor Charts

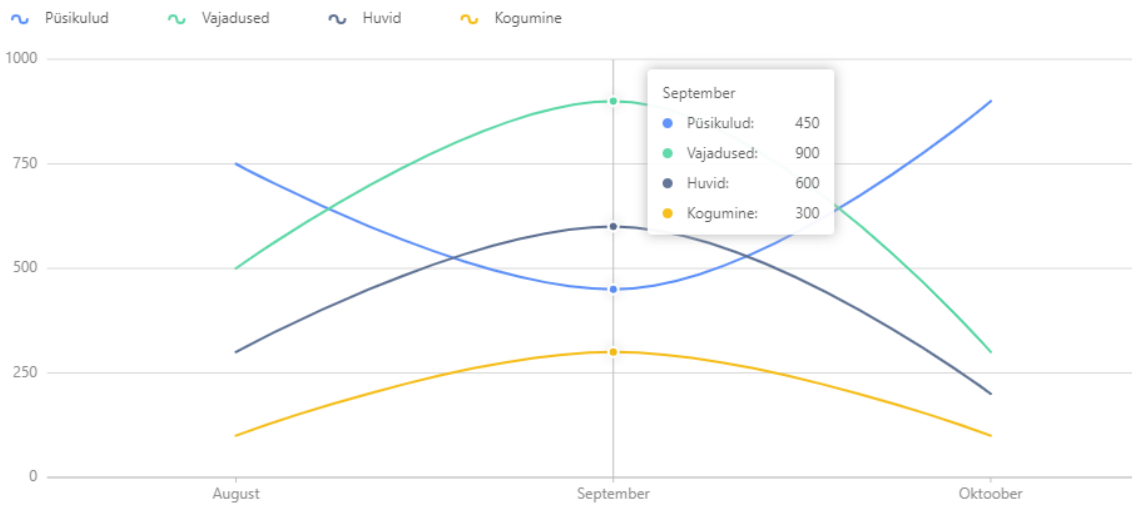
Ant Design Blazor Charts on tasuta teek, millele kehtib „Apache License 2.0“ litsents. Litsentsi tuumaks on, et teegi kasutamine ja muutmine on lubatud nii era- kui ärikasutuseks, kuid tuleb säilitada algne autoriõiguse teavitus. Blazori jaoks valmistatud teek põhineb JavaScriptile loodud G2Plot diagrammiteegil. Viimane uuendus Ant Design Blazor Charts teegile on lisatud 2021 novembris [23].

Teegi dokumentatsioon on puudulik. Kuigi teegi tutvustus on ingliskeelne, on näited ja kirjeldused kohati hiinakeelsed. Dokumentatsioon hõlmab paigaldusjuhendit, kuid mitte kasutusjuhendit [23]. Osalise kasutusjuhise leiab G2Plot dokumentatsioonist [24].

Diagrammi kuvamiseks tuleb lehele lisada diagrammitüübile vastav komponent. Erinevate diagrammide jaoks on teegis 36 erinevat komponenti. Komponent võtab parameetritena sisse andmed ja konfiguratsiooni.

Andmetena sisestatakse kollektsioon objekte, kusjuures iga objekt peab vastama ühele andmepunktile. Konfiguratsioonis täpsustatakse andmepunktidena kuvatavate väljade nimed. Lisaks saab konfiguratsiooni kaudu lisada diagrammile pealkirja, sildid ja teha mõningaid valikuid kujunduse osas. Juhiste järgi toimides diagramm paraku siiski programmis sisestatud pealkirja ega kirjeldust ei kuva.

Ant Design Blazor Charts teegis loodud diagrammi välimuse näidis on välja toodud Joonisel 1.



Joonis 1. Ant Design Blazor Charts teegiga kirjutatud diagrammi näidis.

Kursoriga üle diagrammi liikudes kuvatakse eraldi hüpikaknas kursori alla jäävate andmepunktide detailid.

Kujundusvalikud konfigureeritakse võlusõnedena, kuid dokumentatsioon ei anna ülevaadet sõnede võimalikest väärtustest.

Võlusõne, ehk inglise keeles *magic string*, on programmeerimises konstantse nimetuse edasi andmine vabal kujul kirjutatud sõnena. Seda tava tuleks muude valikute olemasolul

vältida, sest arenduskeskkond ei kontrolli sõnede sisu ja ainus näpuviga tähendab, et programm ei tunne konstantset nimetust ära [2].

Teegi dokumentatsioonis toodud näited on kohati vigased ja nende tööle saamiseks tuleb etteantud koodis teha korrektuure. Teegi C# koodis on kasutatud JavaScriptile omast stiili ja nimetustavasid.

Ant Design Blazor Charts teegis näidise loomiseks kirjutatud koodi tööpõhimõte on välja toodud Joonisel 2.

```
<Line Data="budgets" Config="config" />

@code{
   private List<Budget> budgets;

   LineConfig config = new()
    {
        Title = new Title
        {
            Visible = true,
            Text = "Eelarvete ajalugu",
        },
        Description = new Description
        {
            Visible = true,
            Text = "Kirjeldus...",
        },
        Padding = "auto",
        ForceFit = true,
        XField = "time",
        YField = "money",
        Smooth = true,
        SeriesField = "type"
    };

    private class Budget
    {
        public string Time { get; set; }
        public decimal Money { get; set; }
        public string Type { get; set; }
    }

}
```

Joonis 2. Ant Design Blazor Charts teegiga kirjutatud diagrammi tööpõhimõte.

3.2.2 Blazly

Blazly on tasuta teek, millele kehtib millele kehtib MIT litsents. Litsentsi tuumaks on, et teegi kasutamine ja muutmine on lubatud nii era- kui ärikasutuseks, kuid tuleb säilitada

algne autoriõiguse teavitus. Blazori jaoks valmistatud teek põhineb JavaScriptile loodud Plotly graafikateegil. Viimane uuendus Blazly teegile on lisatud 2019 aprillis [25].

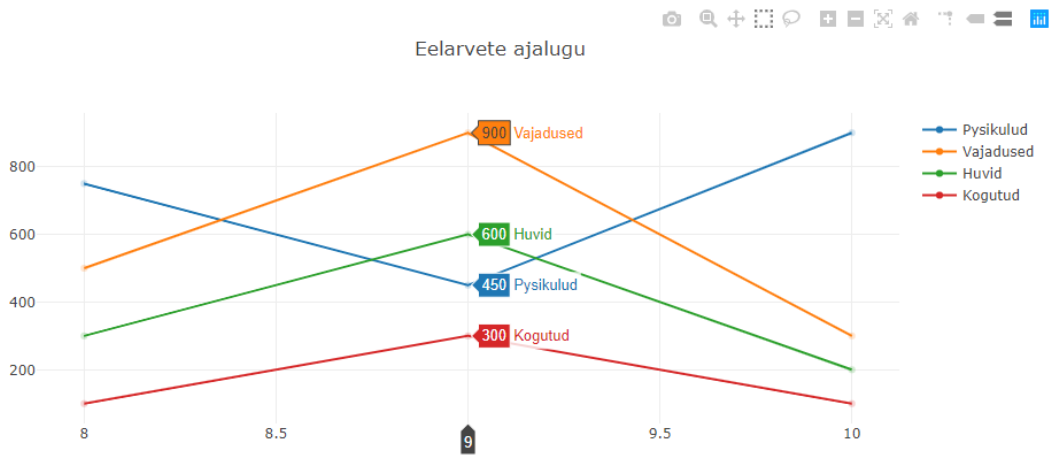
Teegi dokumentatsioon piirdub *readme* failiga, kus on paigaldusjuhend ja üks kasutusnäidis. Tegemist on minimalistliku kergkaalu teegiga, kus diagrammi kujundamise valikuid ei pakuta. Diagrammi tüüp määratakse võlusõnega, mille vasted saab leida Plotly JavaScript dokumentatsioonist [26].

Blazly on sobiv teek rakenduse jaoks, kus on vaja kuvada lihtsat diagrammi ja seada see üles võimalikult väheste ressurssidega. Teegi C# osa koosneb vaid kahest .cs failist ja ühest .razor lehest.

Diagramm lisatakse ühe kindlaksmääratud komponendina. Andmepunktid tuleb salvestada teegiga kaasa tulevatele `Trace` klassi objektidele. Diagrammi komponent võtab parameetritena sisse komponendi Id, pealkirja ja kollektsiooni `Trace` instantsidest.

Erinevalt teistest võrreldud teekidest ei ole Blazly saadaval NuGetina (NuGet on C#/.NET platvormi teekide paigaldamise standardlahendus). See tähendab, et teegi projekt ja vajalikud JavaScript failid tuleb rakendusse kopeerida teegi repositooriumist käsitsi.

Blazly teegis loodud diagrammi välimuse näidis on välja toodud Joonisel 3.



Joonis 3. Blazly teegiga kirjutatud diagrammi näidis.

Blazly diagrammidele genereeritakse vaikimisi interaktsiooninupud, millega kasutaja saab diagrammi sisse-välja suurendada, vaadelda kindlaid silte või andmepunkte ning

selekteerida diagrammi lõike. Samuti kuvatakse eraldi hüpikaknas kursori alla jäävate andmepunktide detailid.

Mõlemale teljele on võimalik väärtusena anda ainult ujuvkomaarve, mitte näiteks kuupäevasõnesid. Siltidel ei ole vaikimisi võimalik kasutada täpitähti.

Blazly teegis näidise loomiseks kirjutatud koodi tööpõhimõte on välja toodud Joonisel 4.

```
<Plot Id="budgetChart"
      Data="@traces"
      Layout="@chartLayout"
      />

@code {
    Trace[] traces;

    public class Trace
    {
        public float[] X { get; set; }
        public float[] Y { get; set; }
        public string Type { get; set; }
        public string Name { get; set; }
    }

    Layout chartLayout = new()
    {
        Title = "Eelarvete ajalugu"
    };
}
```

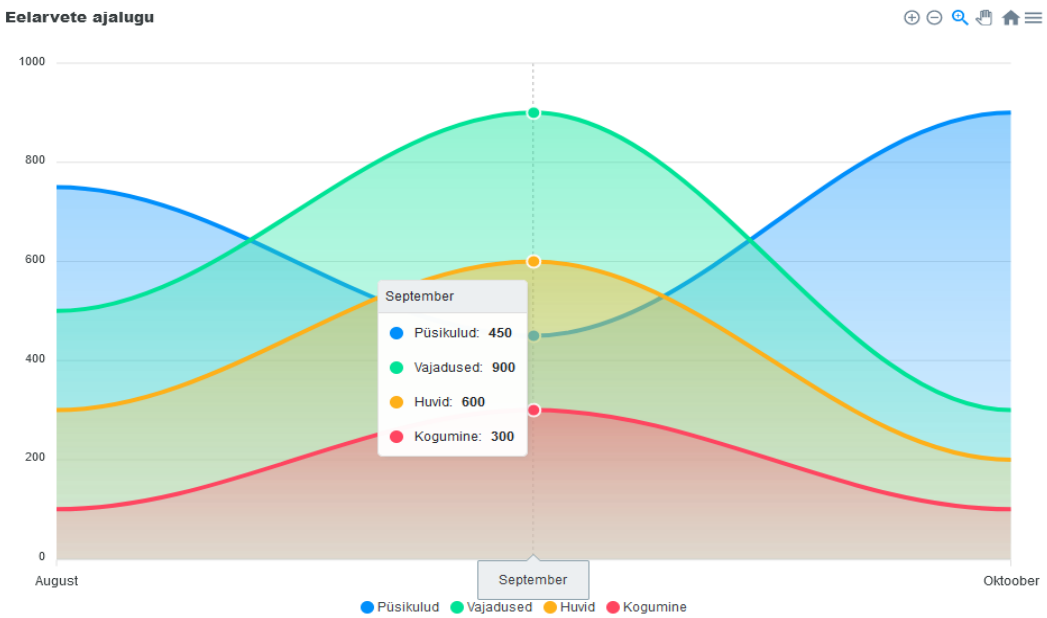
Joonis 4. Blazly teegiga kirjutatud diagrammi tööpõhimõte.

3.2.3 Blazor ApexCharts

Blazor ApexCharts on tasuta teek, millele kehtib MIT litsents. Blazori jaoks valmistatud teek põhineb JavaScriptile loodud ApexCharts diagrammiteegil. Viimane uuendus Blazor ApexChart teegile on lisatud 2022 aprillis [27].

Teek on dokumenteeritud näidislehtede abil. Igale diagrammitüübile vastab üks näidisleht, mille alaosas on välja toodud kasutatud koodi põhimõte [28]. Teegi dokumentatsioonis on selgelt arusaadav paigaldusjuhend [27], aga eraldiseisev kasutusjuhend on koostatud ainult ApexChart JavaScripti teegi kohta [29].

Blazor ApexCharts teegis loodud diagrammi välimuse näidis on välja toodud Joonisel 5.



Joonis 5. Blazor ApexCharts teegiga kirjutatud diagrammi näidis.

Blazor ApexCharts diagrammidele genereeritakse vaikimisi interaktsiooninupud, millega kasutaja saab diagrammi sisse-välja suurendada või edasi-tagasi lohistada. Samuti kuvatakse eraldi hüpikaknas kursori alla jäävate andmepunktide detailid.

Diagrammi kuvamiseks tuleb lehele lisada universaalne diagrammikomponent ja selle sisse eraldi alamkomponendid iga andmerea kohta.

Nii diagrammikomponent kui andmerea komponent võtavad parameetrina sisse andmeid esindava klassi. Diagrammikomponendile antakse parameetrina sisse ka diagrammi pealkiri. Andmerea komponendi juures täpsustatakse, millistelt klassi väljadelt tuleb lugeda diagrammil kuvatavaid andmeid ja silte. Lisaks antakse andmerea komponendile atribuutideks diagrammi tüüpi määrav konstant, andmerea nimetus ja soovi korral sortimisjärjestus. Kokku on teegis saadaval 12 erinevat diagrammi tüüpi.

Komponentide atribuudid on C# keeleruumi integreeritud ja kasutavad valdavalt kompilaatori poolt kontrollitavaid klassinimesid ja konstantide loendeid (*enum*).

Blazor ApexCharts teegis näidise loomiseks kirjutatud koodi tööpõhimõte on välja toodud Joonisel 6.

```
<ApexChart TItem="Budget"
           Title="Eelarvete ajalugu">

    <ApexPointSeries TItem="Budget"
                     Items="budgets"
                     SeriesType="@SeriesType.Area"
                     Name="Püsikulud"
                     XValue="@(x => x.Time)"
                     YValue="@(y => y.Expenses)"/>

    <ApexPointSeries TItem="Budget"
                     Items="budgets"
                     SeriesType="@SeriesType.Area"
                     Name="Vajadused"
                     XValue="@(x => x.Time)"
                     YValue="@(y => y.Needs)"/>
</ApexChart>

@code {
    private List<Budget> budgets;

    private class Budget
    {
        public decimal Expenses { get; set; }
        public decimal Needs { get; set; }
        public decimal Wants { get; set; }
        public decimal Savings { get; set; }
        public string Time { get; set; }
    }

}
```

Joonis 6. Blazor ApexCharts teegiga kirjutatud diagrammi näidis.

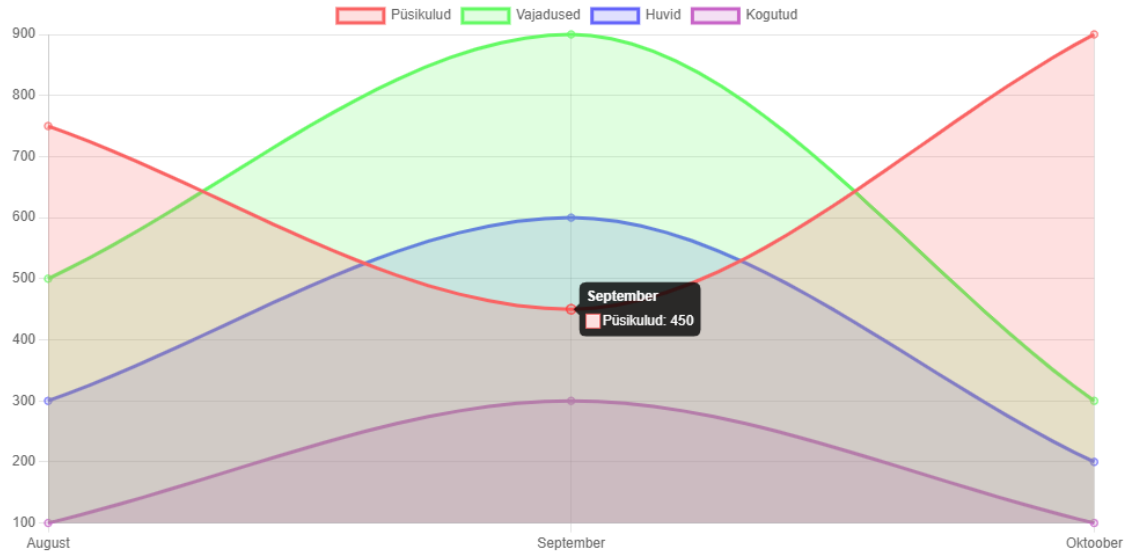
3.2.4 Blazorise

Blazorise on *freemium* hinnastuspõhimõttega teek, mis tähendab, et selle alusosad on tasuta, kuid võimekamad ja spetsiifilisemad osad on saadaval vaid tasu eest. Tasuliste funktsioonide kasutamiseks tuleb liituda Blazorise Professional programmiga, mis maksab 499 USA dollarit aastas. Diagrammide kuvamise tööriistad on saadaval teegi tasuta versioonis [30].

Blazorise teegile kehtib erilitsents, mille tuumaks on, et Blazorise tasuta versiooni tohivad kasutada eraisikud ja sellised ettevõtted, mille aastakäive ei ületa ühte miljonit USA dollarit ja kuhu ei kuulu rohkem kui viis töötajat. Litsents keelab teegi edasi müümist või jagamist. See seab spetsiifilised piirangud lähtekoodi muutmisele. Litsents keelab teegi ostmise ja selle abil valmistatud rakenduste müümise teatud riikides [31].

Viimane uuendus Blazorise teegile on lisatud 2022 aprillis [32].

Blazorise teegis loodud diagrammi välimuse näidis on välja toodud Joonisel 7.



Joonis 7. Blazorise teegiga kirjutatud diagrammi näidis.

Diagrammi kohale genereeritakse vaikimisi iga andmerea kohta vastava värviga kastike, millele vajutades saab andmerida diagrammil sisse-välja lülitada, kusjuures diagrammi suurendusaste kohandub automaatselt hetkel kuvatavate andmeridade järgi. Samuti kuvatakse eraldi hüpikaknas kursori alla jääva üksiku andmepunkti detailid.

Blazorise teek on põhjalikult dokumenteeritud ja hõlmab nii paigaldus- kui kasutusjuhiseid. Lisaks on lähtekood dokumenteeritud ka kommentaaridega. Blazorise on loodud eksklusiivselt Blazor raamistiku jaoks. Tänu sellele on teegis ära kasutatud Blazor raamistiku spetsiifikat.

Blazorise teegi paigaldamiseks tuleb projektile liita mitu erinevat NuGet paketti ja modifitseerida rakenduse käivitusprogrammi seadistust.

Blazorise on mahukas ja võimas teek, kus diagrammide kuvamine on ainult üks paljudest funktsionaalsustest. Diagrammi lisamiseks tuleb lehele lisada diagrammi tüübile vastav komponent. Teegis on valida 7 erineva diagrammitüübi komponendi vahel.

Diagrammi komponendil on ainult üks sisendparameeter, milleks on diagrammil kuvatav andmetüüp. Diagrammi komponent tuleb salvestada muutujasse. Andmete lisamiseks tuleb muutujasse salvestatud komponendi pealt välja kutsuda meetod, mis võtab sisendiks

sildid ja andmestiku. Sildid sisestatakse sõnede massiivina. Andmestik sisestatakse ettemääratud klassi kujul. Näiteks joondiagrammi puhul vastab ühele andmereale üks `LineChartDataset<andmetüüp>` instants. Sama klassi peal täpsustatakse ka andmerea kujundus.

Blazorise teegis näidise loomiseks kirjutatud koodi tööpõhimõte on välja toodud Joonisel 8.

```
<LineChart @ref="lineChart" TItem="decimal" />

@code{
    LineChart<decimal> lineChart;

    protected override async Task OnAfterRenderAsync(bool firstRender)
    {
        if (firstRender)
        {
            await lineChart.AddLabelsDatasetsAndUpdate(
                Labels,
                Expenses,
                Needs,
                Wants,
                Savings);
        }
    }

    LineChartDataset<decimal> Expenses = new()
    {
        Label = "Püsikulud",
        Data = new List<decimal> {750, 450, 900},
        BackgroundColor = new List<string>
        {
            ChartColor.FromRgba( 250, 100, 100, 0.2f )
        },
        BorderColor = new List<string>
        {
            ChartColor.FromRgba( 250, 100, 100, 1f )
        },
        Fill = true,
        PointRadius = 3,
        CubicInterpolationMode = "monotone",
    };

    string[] Labels = {"August", "September", "Oktoober"};

}
```

Joonis 8. Blazorise teegiga kirjutatud diagrammi tööpõhimõte.

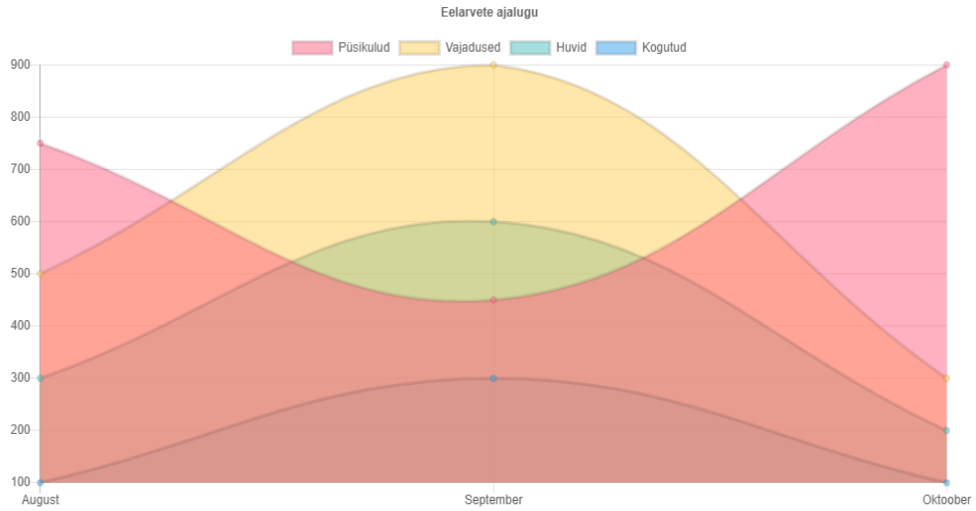
3.2.5 ChartJs.Blazor

ChartJs.Blazor on tasuta teek, millele kehtib MIT litsents. Blazori jaoks valmistatud teek põhineb JavaScriptile loodud ChartJs diagrammiteegil. Viimane uuendus ChartJs.Blazor teegile on lisatud 2021 jaanuaris [33].

Teek on dokumenteeritud *readme* faili [33] ja näidislehtedega, [34]. Lähtekoodi on dokumenteeritud ka kommentaaridega. Teegi lähtekoodis esineb aga kompileerimisvigu. Samuti esineb vastuolusid teegi avalike näidislehtede ja lähtekoodi vahel, sest näidislehed on valmistatud teegi vanema versiooniga [35].

Teegi esialgse looja loobumise järel võttis arenduse ja haldamise ainuisikuliselt üle 18-aastane nooruk, kelle käe all jõudis teek kasutusse Microsofti ametlikel näidistel ja esimeste kohtade hulka Google jt otsingumootorite vastetes päringule „blazor chart“. Noor autor ehmus teegi plahvatuslikust edust ja on tänaseks teegi hoolduse ja arenduse lõpetanud [35].

ChartJs.Blazor teegis loodud diagrammi välimuse näidis on välja toodud Joonisel 9.



Joonis 9. ChartJs.Blazor teegiga kirjutatud diagrammi näidis.

Diagrammi kohale genereeritakse vaikimisi iga andmerea kohta vastava värviga kastike, millele vajutades saab andmerida diagrammil sisse-välja lülitada, kusjuures diagrammi suurendusaste kohandub automaatselt hetkel kuvatavate andmeridade järgi.

Teek on C# keeleruumi hästi integreeritud. Võlusõnede asemel on valdavalt kasutatud konstante, klasse ja meetodeid.

Diagrammi kuvamiseks tuleb lehele lisada diagrammitüübile vastav komponent. Erinevate diagrammide jaoks on teegis 13 erinevat komponenti. Komponent võtab üheainsa parameetri, milleks on diagrammitüübile spetsiifilise konfiguratsiooniklassi instants. Konfigurasiooniklassi kaudu antakse diagrammi komponendile nii andmestik kui kujundusvalikud. Andmestik sisestatakse ettemääratud klassi kujul, millel on väljad sildi, taustavärvi ja andmepunktide jaoks. Andmepunktid salvestatakse klassile numbriliste väärtuste nimekirjana.

ChartJs.Blazor teegis näidise loomiseks kirjutatud koodi tööpõhimõte on välja toodud Joonisel 10.

```
<Chart Config="_config" />

@code {

    private LineConfig _config = new()
    {
        Options = new LineOptions()
        {
            Responsive = true,
            Title = new OptionsTitle
            {
                Display = true,
                Text = "Eelarvete ajalugu"
            }
        },
        Data =
        {
            Labels = {"August", "September", "Oktoober"},
            Datasets =
            {
                new LineDataset<decimal>(new List<decimal>()
                {
                    750, 450, 900
                })
                {
                    Label = "Püsikulud",
                    BackgroundColor = ColorUtil.ColorString(
                        255, 99, 132, 0.5)
                },
                new LineDataset<decimal>(new List<decimal>()
                {
                    100, 300, 100
                })
                {
                    Label = "Kogutud",
                    BackgroundColor = ColorUtil.ColorString(
                        54, 162, 235, 0.5)
                }
            }
        }
    };

}
```

Joonis 10. ChartJs.Blazor teegiga kirjutatud diagrammi tööpõhimõte.

3.2.6 ComponentOne

ComponentOne on tasuline teek, mille kaudu saab luua kasutajaliideseid erinevate C#/.NET tehnoloogiate peal ja muuhulgas ka Blazor raamistikus. Kitsalt Blazori jaoks mõeldud teegiosa kasutamise hind on 449 USA dollarit ühe kasutaja kohta aastas. Teegiga tutvumiseks on lubatud selle tasuta kasutamine 30-päevase prooviperioodi jooksul [36].

ComponentOne teegile kehtib erilitsents, mis annab õiguse teegi kasutamiseks ainult aktiivse tasulise tellimusega arendajatele. Teeki saab paigaldada ainult koos litsentsi

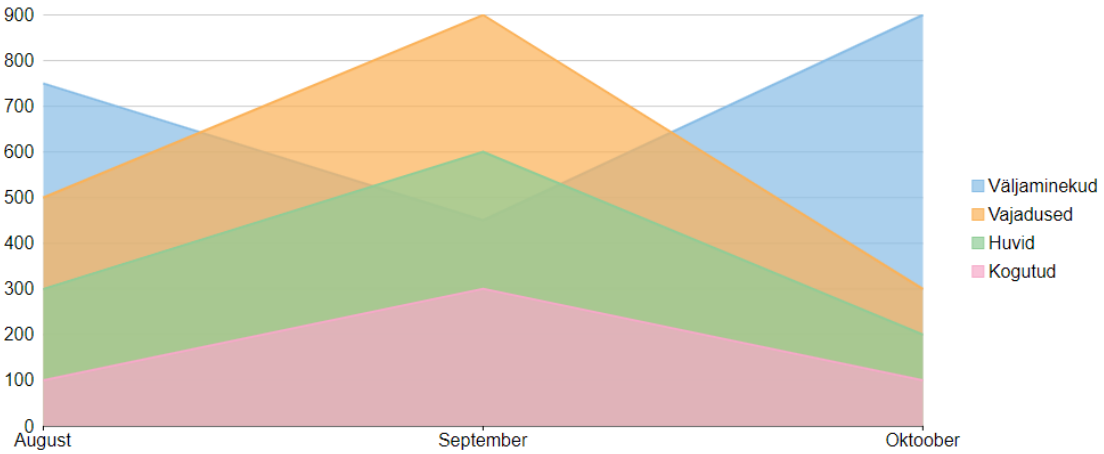
kontrolliva tarkvaraga. Litsents keelab teegi edasi müümist või jagamist ja seab spetsiifilised piirangud lähtekoodi muutmisele. Litsents keelab teegi ostmise ja selle abil valmistatud rakenduste müümise teatud riikides [37].

Teek on dokumenteeritud kodulehe eraldi sektsioonis. Dokumentatsioonis on kasutatud illustratsioone ja koodinäiteid. Dokumentatsioon hõlmab nii paigaldusjuhist kui kasutusjuhiseid. Lisaks on olemas eraldi otsinguvõimalus dokumentatsioonist spetsiifiliste võtmesõnade leidmiseks. [38]

Teegi paigaldamiseks tuleb kasutada spetsiaalset tarkvara nimega ComponentOne ControlPanel, mille saab alla laadida ComponentOne kodulehelt. Selle kaudu laetakse arvutisse soovitud teeki sisaldav NuGet pakett, misjärel muutub pakett nähtavaks arenduskeskkonnas, kust saab selle projektile liita.

Teegi funktsioonide hulka kuulub lisaks diagrammide näitamisele veel mitmeid teisi kasutajaliidese kujundamise võimalusi. Kokku on ComponentOne teegis saadaval 25 erinevat diagrammi tüüpi.

ComponentOne teegis loodud diagrammi välimuse näidis on välja toodud Joonisel 11.



Joonis 11. ComponentOne teegiga kirjutatud diagrammi näidis.

Diagrammi kuvamiseks tuleb lehele lisada üks võimalikest diagrammi ülemtüüpidest, kuhu peab sisestama vähemalt kolm parameetrit: esiteks diagrammi alamtüüpi täpsustava konstandi, teiseks andmestiku, milleks sobib kollektsioon mis tahes klassi objekte ja kolmandaks andmestikuks valitud klassi selle välja nime, mida kasutatakse X-telje väärtuste jaoks.

Diagrammi komponendi sisse antakse omakorda alamkomponendid, kus üks komponent tähistab ühte andmerida. Andmerea komponent võtab parameetritena sisse andmerea nimetuse ja andmestikuks valitud klassi selle välja nime, mida kasutatakse Y-telje väärtuste jaoks.

Lisaks kirjeldatud parameetritele saab mõlemale komponendile lisada hulgaliselt diagrammi käitumist ja kujundust määravaid parameetreid. Sisse ehitatud võimaluste asemel saab komponendile parameetrina sisestada ka toorest Css keeles kirjutatud kujunduskoodi, mis on integreeritud selliselt, et programmeerimiskeskkond selle ära tunneb, ehk oskab koodiosasid vastavalt värvida ja märgata vigu Css süntaksis.

ComponentOne teegis näidise loomiseks kirjutatud koodi tööpõhimõte on välja toodud Joonisel 12.

```
<FlexChart ChartType="ChartType.Area" ItemsSource="budgets" BindingX="Time">
     <SeriesCollection>
         <Series Name="Väljaminekud" Binding="Expenses" />
         <Series Name="Vajadused" Binding="Needs" />
         <Series Name="Huvid" Binding="Wants" />
         <Series Name="Kogutud" Binding="Savings" />
     </SeriesCollection>
</FlexChart>


@code {
    private List<Budget> budgets;

    public class Budget
    {
        public decimal Expenses { get; set; }
        public decimal Needs { get; set; }
        public decimal Wants { get; set; }
        public decimal Savings { get; set; }
        public string Time { get; set; }
    }
}
```

Joonis 12. ComponentOne teegiga kirjutatud diagrammi tööpõhimõte.

3.2.7 DevExpress

DevExpress pakub erinevaid tasulisi teeke, kuhu hulka kuulub ka Blazor raamistikule valmistatud komponentide teek. Blazorile tehtud teeki ei saa eraldi osta. Selle kasutamiseks tuleb tellida ASP.NET teekide kogum, kuhu kuulub lisaks Blazorile veel mitmete teiste ASP.NET raamistike jaoks loodud teeke. Teekide kogumi hinnaks ühe

arendaja kohta on 999 USA dollarit aastas. Teegiga tutvumiseks on lubatud selle tasuta kasutamine 30-päevase prooviperioodi jooksul [39].

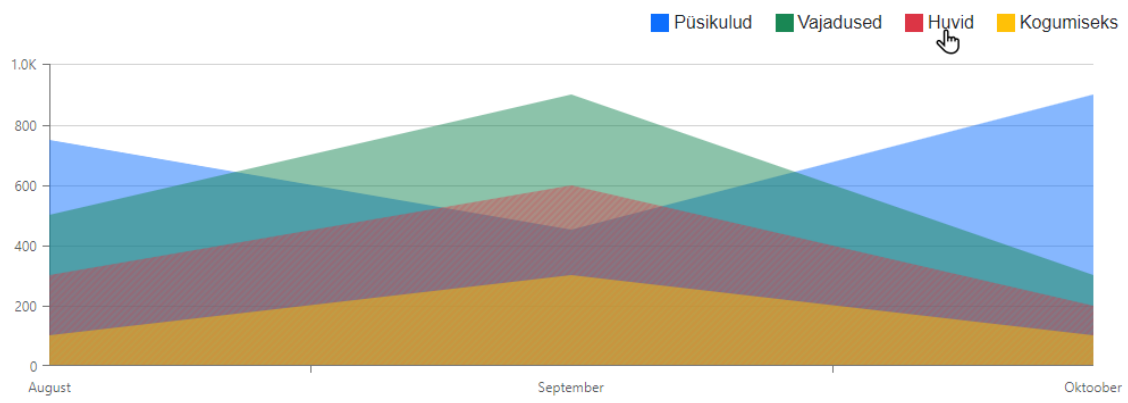
DevExpress teegile kehtib erilitsents, mis annab õiguse teeki paigaldada, kopeerida ja kasutada ainult aktiivse tasulise tellimusega arendajatele. Teeki saab paigaldada ainult koos litsentsi kontrolliva tarkvaraga. Litsents keelab teegi edasi müümist või jagamist ja seab spetsiifilised piirangud lähtekoodi muutmisele. Litsents keelab teegi ostmise ja selle abil valmistatud rakenduste müümise teatud riikides [40].

Teek on dokumenteeritud kodulehe eraldi sektsioonis. Dokumentatsioonis on kasutatud illustratsioone ja koodinäiteid. Dokumentatsioon hõlmab nii paigaldus- kui kasutusjuhendeid [41].

Teegi paigaldamine on võimalik kahel viisil – kas läbi spetsiaalse paigaldustarkvara, mille saab alla laadida DevExpressi kodulehelt või kasutades NuGet paketti. NuGet paketina paigaldamiseks tuleb DevExpressi kodulehel luua konto, hankida isiklikud tuvastuskoodid ning luua ühendus DevExpessi NuGet otspunktiga.

DevExpress Blazor teegi funktsioonide hulka kuulub lisaks diagrammide näitamisele veel mitmeid teisi kasutajaliidese kujundamise võimalusi. DevExpress teegis on 7 erinevat diagrammi põhitüüpi, kuid sõltuvalt diagrammi kujundusest võivad ühe ja sama põhitüübiga diagrammid olla oluliselt erineva välimusega.

DevExpress teegis loodud diagrammi välimuse näidis on välja toodud Joonisel 13.



Joonis 13. DevExpress teegiga kirjutatud diagrammi näidis.

Diagrammile lisatakse vaikimisi funktsionaalsus, kus kursoriga andmerea sildi kohale liikumine värvib diagrammil vastava ala triibuliseks.

Diagrammi kuvamiseks tuleb lehele lisada üks üldine diagrammi komponent, mis võtab parameetritena sisse andmestiku ja diagrammi laiuse. Andmestikuks sobib kollektsioon mis tahes klassi objekte.

Diagrammi komponendi sisse antakse omakorda alamkomponendid. Alamkomponente on üks iga andmerea kohta ja lisakomponendid siltide kuvamiseks ja kujundamiseks. Andmerea komponent võtab parameetrina sisse nime, andmerea klassi, sildi andmetüübi, sildi välja, väärtuse andmetüübi ja väärtuse välja. Parameetrid sisestatakse kontrollitud C# väärtustena.

Lisaks kirjeldatud parameetritele saab kõikidele komponendile lisada mõningaid diagrammi käitumist ja kujundust määravaid parameetreid.

DevExpress teegis näidise loomiseks kirjutatud koodi tööpõhimõte on välja toodud Joonisel 14.

```
<DxChart Data="@budgets"
         Width="100%">
    <DxChartAreaSeries Name="Püsikulud"
                       T="Budget"
                       TArgument="string"
                       TValue="decimal"
                       ArgumentField="b => b.Time"
                       ValueField="b => b.Expenses"/>
    <DxChartAreaSeries Name="Vajadused"
                       T="Budget"
                       TArgument="string"
                       TValue="decimal"
                       ArgumentField="b => b.Time"
                       ValueField="b => b.Needs"/>
    <DxChartArgumentAxis SideMarginsEnabled="false"/>
    <DxChartLegend Position="RelativePosition.Outside"
                   HorizontalAlignment="HorizontalAlignment.Right"/>
</DxChart>

@code {
    private List<Budget> budgets;

    public class Budget
    {
        public decimal Expenses { get; set; }
        public decimal Needs { get; set; }
        public decimal Wants { get; set; }
        public decimal Savings { get; set; }
        public string Time { get; set; }
    }
}
```

Joonis 14. DevExpress teegiga kirjutatud diagrammi tööpõhimõte.

3.2.8 Ignite UI

Ingite UI on tasuline teekide kogum, mis sisaldab teeke erinevate JavaScripti ja C#/.NET raamistike jaoks, sealhulgas ka Blazorile. Blazorile tehtud teeki ei saa eraldi osta. Selle kasutamiseks tuleb tellida terve Ignite UI teekide kogum, mille hinnaks ühe arendaja kohta on 1295 USA dollarit aastas. Teegiga tutvumiseks on lubatud selle tasuta kasutamine 30-päevase prooviperioodi jooksul [42].

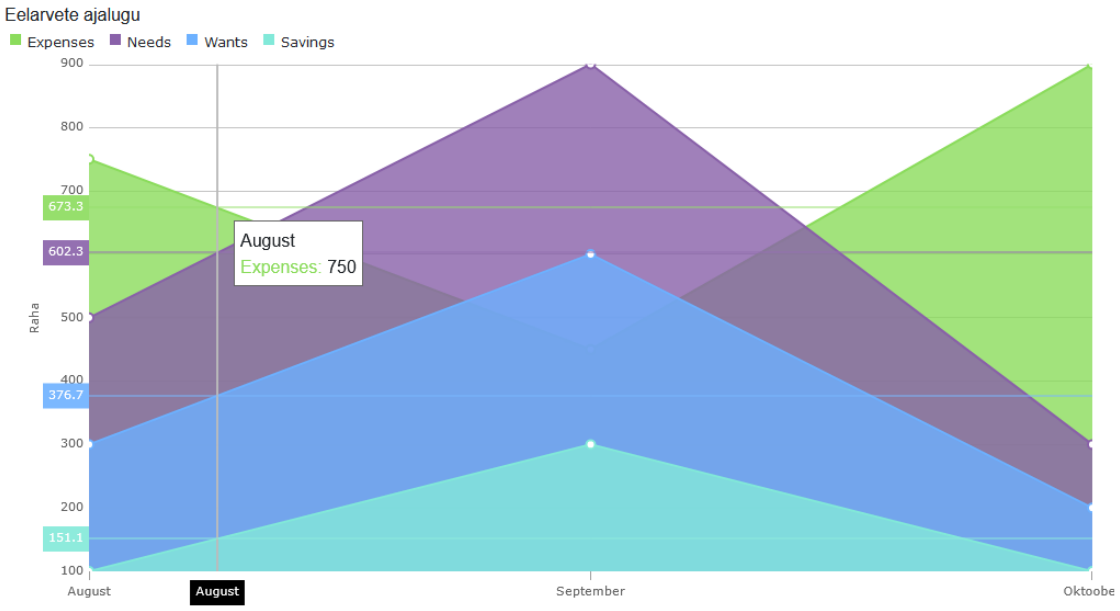
Ignite UI teegile kehtib erilitsents, kus hoiatatakse, et teegiga valmistatud rakendused kuuluvad küll valmistajale, aga kõikvõimalikud teegi edasiarendused kuuluvad sõltumata valmistajast teegi omanikele. Litsents keelab teegi edasi müümist või jagamist ja seab spetsiifilised piirangud lähtekoodi muutmisele. Litsents keelab teegi ostmise ja selle abil valmistatud rakenduste müümise teatud riikides [43]. Isiklikel mitteärilistel eesmärkidel on lubatud teeki kasutada ilma litsentsita [44].

Teek on dokumenteeritud kodulehe eraldi sektsioonis. Dokumentatsioon hõlmab nii paigaldusjuhist kui kasutusjuhiseid. Dokumentatsioonis on kasutatud illustratsioone ja koodinäiteid [45].

Kuigi Ignite UI teegi äriliseks kasutuseks on vaja tasulist litsentsi, käib selle paigaldamine läbi üldkättesaadava NuGet paketi. Teegi toimimiseks peab arvutisse olema paigaldatud Node.js (avatud lähtekoodiga spetsiifiline JavaScripti käitussüsteem). Teegi üles seadmiseks on vaja kohandada rakenduse käivitusprogrammi ja kasutada teegispetsiifilisi sõltuvussisestusi.

Ingite UI Blazor teek sisaldab 19 erinevat diagrammi tüüpi. Kokkuvõttes ei ole Ignite UI siiski kitsalt diagrammide kuvamise tööriist ja sisaldab mitmeid muid võimalusi kasutajaliidese kujundamiseks ja elavdamiseks.

Ignite UI Blazor teegis loodud diagrammi välimuse näidis on välja toodud Joonisel 15.



Joonis 15. Ignite UI teegiga kirjutatud diagrammi näidis.

Vaikesätetega loodud diagrammi puhul kuvatakse kursoriga üle diagrammi liikudes diagrammi vasakul küljel andmeridade tuletatud vaheväärtused erinevate andmepunktide vahel.

Diagrammi kuvamiseks tuleb lehele lisada üks üldine diagrammi komponent, mis võtab parameetritena sisse andmestiku, kujundusvalikud ja diagrammi tüübi. Mõnede

diagrammi tüüpide puhul on legendi kuvamiseks vaja kasutada teist eraldiseisvat komponenti.

Andmestikuks sobib kollektsioon mis tahes klassi objekte. Seejuures ei ole vaikimisi vaja täpsustada, milliseid klassi välju diagrammile kantakse. Kui klassi üks väljadest on sõne, kasutatakse seda X-telje sildina ja kõiki numbrilisi muutujaid Y-telje väärtustena, kusjuures iga väli moodustab diagrammil automaatselt eraldi andmerea. Andmerea nimena kasutatakse vaikimisi klassi vastava välja nime.

Ignite UI teegis näidise loomiseks kirjutatud koodi tööpõhimõte on välja toodud Joonisel 16.

```
@inject IIgniteUIBlazor IgniteUiBlazor

<div class="container vertical">
    <div class="options vertical">
        <span class="legend-title">Eelarvete ajalugu</span>
        <div class="legend">
            <IgbLegend @ref="Legend"
                       Orientation="LegendOrientation.Horizontal"/>
        </div>
    </div>
    <div class="container vertical">
        <IgbCategoryChart Height="500px" Width="1000px"
                          Legend="Legend"
                          DataSource="Data"
                          ChartType="CategoryChartType.Area"
                          YAxisTitle="Raha"
                          YAxisTitleLeftMargin="10"
                          YAxisTitleRightMargin="5"
                          YAxisLabelLeftMargin="0"
                          IsHorizontalZoomEnabled="false"
                          IsVerticalZoomEnabled="false"/>
    </div>
</div>

@code {

    private List<Budget> Data;
    private IgbLegend _Legend;

    private IgbLegend Legend
    {
        get { return _Legend; }
        set
        {
            _Legend = value;
            StateHasChanged();
        }
    }

    protected override void OnInitialized()
    {
        IgbCategoryChartModule.Register(_igniteUiBlazor);
        IgbLegendModule.Register(_igniteUiBlazor);
    }

    public class Budget
    {
        public decimal Expenses { get; set; }
        public decimal Needs { get; set; }
        public decimal Wants { get; set; }
        public decimal Savings { get; set; }
        public string Time { get; set; }
    }

}
```

Joonis 16. Ignite UI teegiga kirjutatud diagrammi tööpõhimõte.

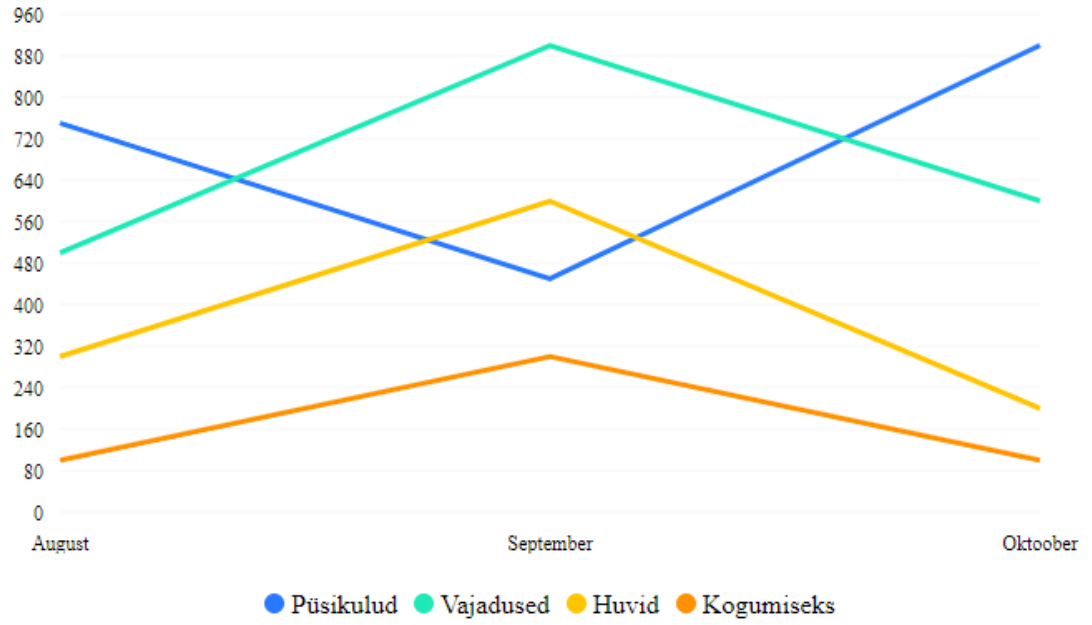
3.2.9 MudBlazor

MudBlazor on tasuta teek, millele kehtib MIT litsents. Teek on valmistatud spetsiaalselt Blazor raamistikule, ehk tegemist pole lihtsalt mõne JavaScripti teegi teisendusega. Viimane uuendus MudBlazor teegile on lisatud 2022 aprillis [46].

Teek on põhjalikult dokumenteeritud ja hõlmab nii koodinäidiseid kui illustratsioone. Dokumentatsioon sisaldab ka paigaldusjuhist. MudBlazor teek ei ole mõeldud kitsalt diagrammide kuvamiseks, vaid pakub laiemat hulka kujunduskomponente. Teek sisaldab 4 erinevat diagrammi põhitüüpi. [47]

MudBlazor kasutab rakenduse kujundamisel spetsiaalset Css koodi ega sõltu Bootstrapist (Bootstrap on laialt levinud kujunduskoodide kogumik, mis on Blazor raamistiku rakenduste kujundamise vaikelahendus). MudBlazor teegi kasutamisel on koodikonfliktide vältimiseks vaja rakendusest Bootstrap eemaldada.

MudBlazor teegis loodud diagrammi välimuse näidis on välja toodud Joonisel 17.



Joonis 17. MudBlazor teegiga kirjutatud diagrammi näidis.

Diagrammi kuvamiseks tuleb lehele lisada üks universaalne diagrammi komponent, mis võtab parameetritena sisse diagrammitüübi, andmestiku, sildid ja mõned kujundusvalikud. Andmeridadena kasutatakse spetsiaalset teegiga kaasatulevat klassi, kuhu salvestatakse kaks väärtust: andmerea nimi ja ujukomaarvude massiiv. Massiivis olevad numbrid kantakse diagrammi Y-teljele andmepunktidena.

MudBlazor teegis näidise loomiseks kirjutatud koodi tööpõhimõte on välja toodud Joonisel 18.

```
<div>
    <MudChart ChartType="ChartType.Line"
              ChartSeries="@Series"
              XAxisLabels="@XAxisLabels"
              Width="100%" Height="350px"/>
</div>

@code {
    private List<ChartSeries> Series = new()
    {
        new ChartSeries { Name = "Püsikulud",
            Data = new double[] { 750, 450, 900 } },
        new ChartSeries { Name = "Vajadused",
            Data = new double[] { 500, 900, 600 } },
        new ChartSeries { Name = "Huvid",
            Data = new double[] { 300, 600, 200 } },
        new ChartSeries { Name = "Kogumiseks",
            Data = new double[] { 100, 300, 100 } },
    };

    private string[] XAxisLabels =
    {"August", "September", "Oktoober" };
}
```

Joonis 18. MudBlazor teegiga kirjutatud diagrammi tööpõhimõte.

3.2.10 Syncfusion

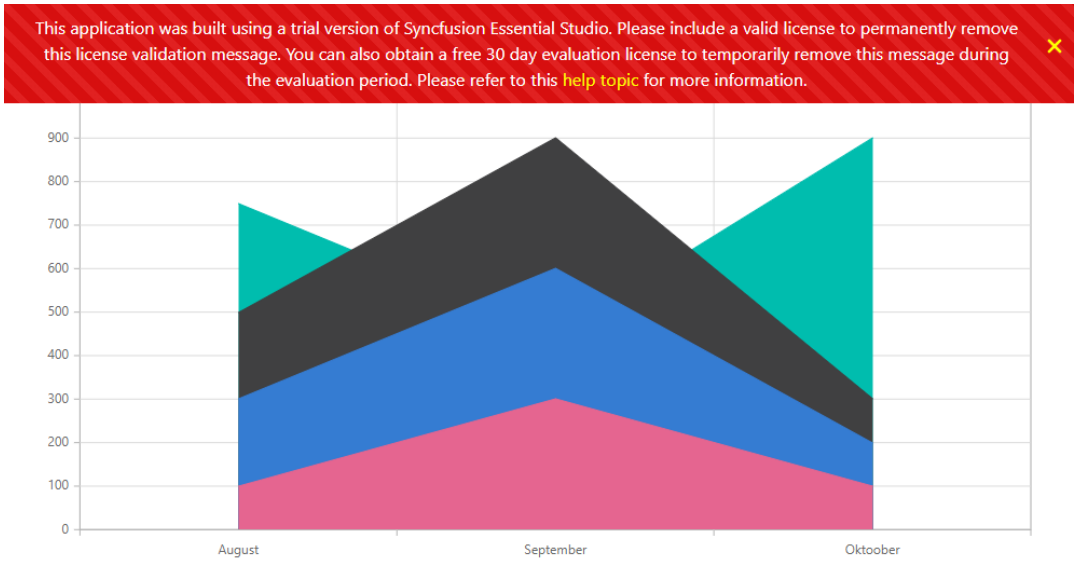
Syncfusion on tasuline teek, mille litsentsi on võimalik osta kahes versioonis – kas kitsalt Blazor raamistikus kasutamiseks või laiemalt erinevate C#/.NET ja JavaScript raamistike jaoks. Blazor raamistiku jaoks on teegi hind 995 USA dollarit ühe arendaja kohta aastas [48].

Teegile kehtib erilitsents, mis annab õiguse teegi kasutamiseks ainult aktiivse tasulise tellimusega arendajatele. Litsents keelab teegi edasi müümist või jagamist ja seab spetsiifilised piirangud lähtekoodi muutmisele. Litsents keelab teegi ostmise ja selle abil valmistatud rakenduste müümise teatud riikides [49]. Teegiga tutvumiseks on lubatud selle tasuta kasutamine 30-päevase prooviperioodi jooksul [50].

Teek on dokumenteeritud kodulehe eraldi sektsioonis. Dokumentatsioon hõlmab nii paigaldusjuhist kui kasutusjuhiseid. Dokumentatsioonis on kasutatud illustratsioone ja koodinäiteid. Teek ei ole mõeldud kitsalt diagrammide kuvamiseks, vaid pakub laiemat hulka kujunduskomponente. Syncfusion teegis on kokku 31 erinevat diagrammi põhitüüpi [51].

Syncfusion teeki saab paigaldada üldkättesaadava NuGet paketi abil, aga ilma registreeritud litsentsita paigaldamise korral kuvab teegi abil valmistatud rakendus punast teavitust, et rakenduse valmistamisel on kasutatud litsentseerimata teeki ja see on lubatud vaid proovimise eesmärgil.

Syncfusion teegis loodud diagrammi välimuse näidis on välja toodud Joonisel 19.



Joonis 19. Syncfusion teegiga kirjutatud diagrammi näidis.

Diagrammi kuvamiseks tuleb lehele lisada üks universaalne ilma parameetriteta diagrammi komponent. Diagrammi detailid lisatakse eraldi alamkomponentidena – üks komponent, mis määrab X-telje siltide kuvamise viisi ja teine komponent, kuhu alla läheb iga andmerea kohta üks täiendav alamkomponent.

Andmestikuks sobib kollektsioon mis tahes klassi objekte. Andmerea komponendis tuleb parameetriga täpsustada, millisest kollektsioonist andmeid lugeda ja milliste nimedega väljade pealt võtta X-telje ja Y-telje väärtused. Välja nimede määramisel kasutatakse võlusõnesid (kuigi nende kasutamisest on võimalik mööda pääseda `nameof()` funktsiooniga).

Syncfusion teegis näidise loomiseks kirjutatud koodi tööpõhimõte on välja toodud Joonisel 20.

```
<SfChart>
    <ChartPrimaryXAxis
ValueType="Syncfusion.Blazor.Charts.ValueType.Category"/>
    <ChartSeriesCollection>
        <ChartSeries DataSource="@Budgets"
                     XName="Time"
                     YName="Expenses"
                     Type="ChartSeriesType.Area"/>
        <ChartSeries DataSource="@Budgets"
                     XName="Time"
                     YName="Needs"
                     Type="ChartSeriesType.Area"/>
        <ChartSeries DataSource="@Budgets"
                     XName="Time"
                     YName="Wants"
                     Type="ChartSeriesType.Area"/>
        <ChartSeries DataSource="@Budgets"
                     XName="Time"
                     YName="Savings"
                     Type="ChartSeriesType.Area"/>
    </ChartSeriesCollection>
</SfChart>

@code{
    private List<Budget> Budgets;

    public class Budget
    {
        public decimal Expenses { get; set; }
        public decimal Needs { get; set; }
        public decimal Wants { get; set; }
        public decimal Savings { get; set; }
        public string Time { get; set; }
    }

}
```

Joonis 20. Syncfusion teegiga kirjutatud diagrammi tööpõhimõte.

3.2.11 Telerik

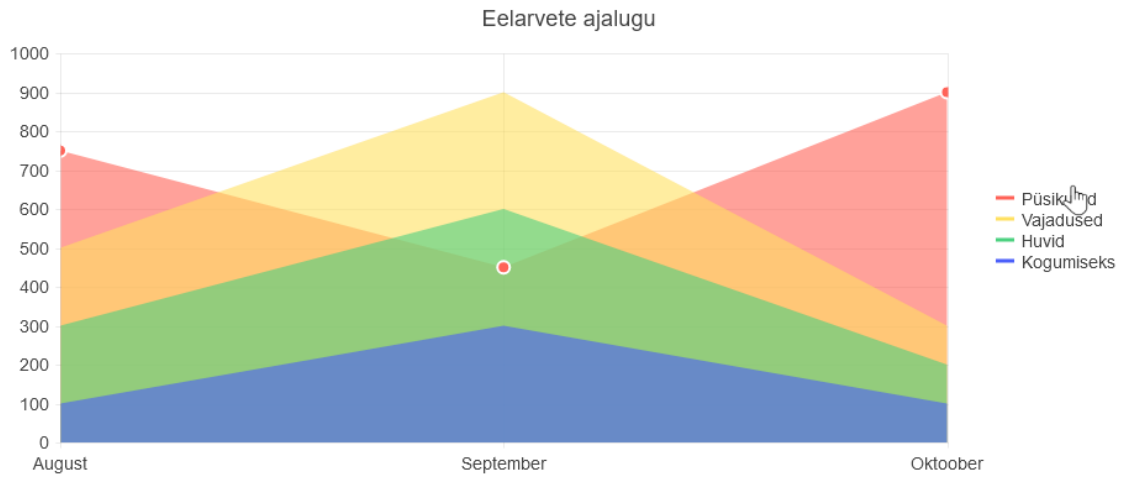
Telerik on tasuline teekide kogum kasutajaliidese kujundamiseks erinevates programmeerimiskeeltes ja raamistikes. Teise valikute hulgas on saadaval ka Blazor raamistikule valmistatud teek, mille litsentsi hinnaks on 899 USA dollarit ühe arendaja kohta aastas [52].

Teegile kehtib erilitsents, mis annab õiguse teegi kasutamiseks ainult aktiivse tasulise tellimusega arendajatele. Litsents keelab teegi edasi müümist või jagamist ja seab spetsiifilised piirangud lähtekoodi muutmisele. Litsents keelab teegi ostmise ja selle abil valmistatud rakenduste müümise teatud riikides. Teegiga tutvumiseks on lubatud selle tasuta kasutamine 30-päevase prooviperioodi jooksul [53].

Teek on dokumenteeritud kodulehe eraldi sektsioonis. Dokumentatsioon hõlmab nii paigaldusjuhist kui kasutusjuhiseid koos illustratsioonide ja koodinäitega. Telerik teegiga on võimalik kasutada 15 erinevat tüüpi diagrammi. Lisaks hõlmab teek mitmeid teisi kasutajaliidese kujundamise võimalusi [54].

Teegi liitmiseks Blazor projektile tuleb Telerik kodulehel luua konto. Seejärel on üks võimalus teegi liitmiseks laadida alla spetsiaalne tarkvara, mis loob arvutisse arenduskeskkonnale nähtava kohaliku NuGet paketi. Teine võimalus on konfigureerida arenduskeskkond selliselt, et see ühenduks Telerik kasutajakonto tunnustega Telerik serverisse ja hangiks teegi üle selle ühenduse [55].

Telerik teegis loodud diagrammi välimuse näidis on välja toodud Joonisel 21.



Joonis 21. Telerik teegiga kirjutatud diagrammi näidis.

Vaikesätetega loodud diagrammi puhul kuvatakse kursoriga andmerea sildi kohale liikudes vastava andmerea andmepunktide peale eristavad mummud.

Diagrammi kuvamiseks tuleb lehele lisada üks universaalne ilma parameetriteta diagrammi komponent, mille sisse lisatakse mitmeid erinevaid alamkomponente. Eraldi alamkomponendid on pealkirja jaoks, andmesiltide jaoks, siltide kujundamise jaoks ja andmestiku jaoks. Andmestiku komponendile lisatakse omakorda iga andmerea kohta üks alamkomponent, millele antakse parameetritena kaasa diagrammitüüp, andmerea nimetus ja lähteandmed.

Lähteandmetena kasutatakse näidistes anonüümsete numbriliste objektide nimekirja. Tegelikult pole aga andmestikule ette antud mingit kindlat vormingut ja puudub kontroll,

et tegemist oleks numbriliste väärtustega. Kui kasutada lähteandmetes mittenumbrilisi objekte, ei tõlgendata seda veana ja vastav andmerida käitub diagrammil ettearvamatult, näiteks sõnede puhul täidab kogu tühja ala, mõnede teiste objektide puhul jäetakse aga lihtsalt kuvamata.

Telerik teegis näidise loomiseks kirjutatud koodi tööpõhimõte on välja toodud Joonisel 22.

```
<TelerikChart>
    <ChartSeriesItems>
        <ChartSeries Type="ChartSeriesType.Area"
                     Name="Püsikulud"
                     Data="@Expenses"/>

        <ChartSeries Type="ChartSeriesType.Area"
                     Name="Vajadused"
                     Data="@Needs"/>

    </ChartSeriesItems>

    <ChartCategoryAxes>
        <ChartCategoryAxis Categories="@xAxisItems"/>
    </ChartCategoryAxes>

    <ChartTitle Text="Eelarvete ajalugu"/>

    <ChartLegend Position="ChartLegendPosition.Right"/>

</TelerikChart>

@code {
    public List<object> Expenses = new() {750, 450, 900};
    public List<object> Needs = new() {500, 900, 300};

    public string[] xAxisItems =
    {"August", "September", "Oktoober"};
}
```

Joonis 22. Telerik teegiga kirjutatud diagrammi tööpõhimõte.

3.3 Diagrammiteegi valimine

Välja töötatavas rakenduses kasutatakse diagramme, et kuvada eelarvete ajaloo põhjal peamiste kulutüüpide jaotust kuude lõikes. Mitmed võrreldud teegid annavad võimaluse diagrammide kuvamiseks, kuid pole mõeldud üksnes selle jaoks, vaid laiemalt kasutajaliidese kujundamiseks. Liigne kood suurendab rakenduse mahtu ja vigade tekkimise võimalusi. Võib oletada, et diagrammidele spetsialiseerunud teek on diagrammide kuvamisel tõhusam, kui teek, kus diagrammide kuvamine on vaid üks

paljudest, vahest isegi kõrvalistest võimalustest. Seetõttu eelistatakse valiku tegemisel teeki, mis on mõeldud spetsiifiliselt just diagrammide kuvamiseks.

Rakendus kuvab kasutajale diagrammi eelarvete ajaloost selleks, et anda visuaalset ülevaadet finantsolukorra muutumise suunast ja suunamuutustest, samuti kulutüüpide osakaaludest ning püsikulude mõjust teistele kulutüüpidele. Nende eesmärkide täitmiseks sobib suurepäraselt aladiagramm. Seega eelistatakse valiku tegemisel teeki, kus on lihtne programmeerida aladiagrammi.

Diagramm peab olema esteetilise välimusega ja kuvatud andmestik selgelt jälgitav. Teek peab andma võimaluse saavutada need eesmärgid võimalikult vähese vaevaga. Väline ilu on subjektiivne, aga näidisdiagrammide välistele omadustele saab anda objektiivse hinnangu. Valiku tegemisel eelistatakse teeki, mis vaikeväärtuste korral:

- kuvab korrektsed andmeridade sildid,
- aladiagrammi korral lisab läbipaistvust, et üks andmerida teist ei varjaks,
- „suhtleb“ kasutajaga, näiteks kursori liikumisele reageerides,
- asetab Y-telje põrandaks alati nullväärtuse.

Rakendust valmistatakse soovist aidata inimestel rahaasjadega paremini toime tulla. Seda ei looda ärilistel eesmärkidel ja selle valmimisega ei kaasne ettenähtavat rahalist tulu. Seetõttu eelistatakse valiku tegemisel teeki, mille kasutamine on tasuta.

Diagrammi võimaluste maksimaalselt ära kasutamiseks on arendajal vaja ülevaadet teegis olevatest võimalustest. Samuti on arendajal vaja ülevaadet vigadest, mis võivad teegi kasutamisel ette tulla ja leida kiiresti viise nende lahendamiseks. Seetõttu eelistatakse valiku tegemisel teeke, mis on suurepäraselt dokumenteeritud.

Blazor raamistik on loodud eesmärgiga anda arendajatele võimalus luua veebirakendusi ilma C# keeleruumist lahkumata. Seda eesmärki arvesse võttes eelistatakse valiku tegemisel teeki, mis on enamat kui ümbritsev liides mõnele olemasolevale JavaScript teegile. Blazor teekidega, mis põhinevad JavaScript teegil, kaasneb suurem võimalus, et mingites spetsiifilistes olukordades tuleb ikkagi kasutada JavaScripti. See läheks Blazor raamistiku kasutamise eesmärgiga vastuollu.

Rakendus kuvab diagramme talletatud eelarvete põhjal. Eelarvete andmed on programmikoodis eraldiseisva klassi kujul. Võimalikult sujuva ja veatu arendustöö huvides eelistatakse valiku tegemisel teeki, kuhu on andmestikuna võimalik sisestada enda loodud klasside kogumikku.

Arendajal on lihtsam kasutada teeki, mille kood vastab headele tavadele. Stiilipuhas keel loob usaldust ja tugevdab teegi töökindlust. Seetõttu eelistatakse valiku tegemisel teeke, mille näidistes pole kasutatud võlusõnesid.

Nendest tingimustest lähtuv kokkuvõtte töö käigus võrreldud diagrammiteekidest on esitatud Tabelis 3. Tulpadesse on koondatud vastavused järgmistele tingimustele:

A. Teek on loodud spetsiaalselt diagrammide kuvamiseks

B. Teegis on lihtne programmeerida aladiagramme

C. Näidisdiagramm vastab varem kirjeldatud esteetika tingimustele

D. Teegi kasutamine mis tahes mahus ja eesmärgil on tasuta

E. Teegil on põhjalik dokumentatsioon ja selged näidised

F. Teek on valmistatud algusest peale Blazor raamistikku silmas pidades, mitte liidesena olemasolevale JavaScripti teegile

G. Diagrammi andmestikuks sobivad mis tahes C# klassid, millel on numbrilisi välju

H. Näidised on programmeeritud ilma võlusõnedeta

Tabel 3. Diagrammiteekide vastavus püstitatud tingimustele

Teegi nimi	**Vastab tingimusele**							
	A	B	C	D	E	F	G	H
Ant Design Blazor Charts	+	-	+	+	-	-	+	-
Blazly	+	-	+	+	-	-	-	+
Blazor ApexCharts	+	+	+	+	-	-	+	+
Blazorise	-	+	-	-	+	+	-	+
ChartJs.Blazor	+	+	-	+	-	-	-	+

ComponentOne	-	+	-	-	+	+	+	-
DevExpress	-	+	+	-	+	+	+	+
Ignite UI	-	+	-	-	+	+	+	+
MudBlazor	-	-	-	+	+	+	-	+
Syncfusion	-	+	-	-	+	+	+	-
Telerik	-	+	+	-	+	+	-	+

Võrdlusest ilmneb, et ükski võrreldud teek ei vasta kõikidele püstitatud nõuetele. Mõned teegid vastavad siiski enamatele tingimustele kui teised.

Kõige täpsemini vastavad tingimustele Blazor ApexCharts ja DevExpress teegid, mis mõlemad täidavad kaheksast püstitatud tingimusest kuus.

Blazor ApexCharts teegi eelisteks DevExpressi ees on, et selle kasutamine mis tahes mahus ja eesmärgil on tasuta ning et see on valmistatud spetsiaalselt diagrammide kuvamiseks. DevExpress teegi eelisteks Blazor ApexCharts ees on, et see on originaalne Blazor teek, mitte JavaScripti teeki ümbritsev liides ning et see on paremini dokumenteeritud.

Autor asetas nende nelja eelise seast kõige suurema kaalu asjaolule, et üks teekidest on tasuta ja teine ei ole. Rakenduse arendust jätkati teegiga Blazor ApexCharts.

4 Rakenduse tehniline kirjeldus

Rakendus on valmistatud C# keeleruumi kuuluvas Blazor raamistikus. Raamistikule on diagrammide kuvamiseks liidetud Blazor ApexCharts teek.

4.1 Autentimine

Rakenduse kasutamiseks tuleb luua kasutajakonto.

Rahaasju peetakse sageli isiklikku ruumi kuuluvaks eluvaldkonnaks, mida ei soovita võõrastega jagada. Tagamaks, et rakendust ei saa kasutada inimeste finantsolukorra järel nuhkimiseks ei nõuta konto loomisel kasutaja reaalseid andmeid. Kasutajalt ei pärita ka kontaktandmeid, sest need võivad reeta tema isiku.

Konto loomiseks küsitakse kasutajalt üksnes vabalt välja mõeldud kasutajatunnust, parooli ja soovi korral salavihjet, mis aitab parooli meelde tuletada. Kasutajanimi peab olema rakenduse kasutajate seas ainulaadne. Tagatud anonüümsuse paratamatuks puuduseks on, et kui kasutaja kaotab oma kasutajanime ja/või parooli, kaotab ta jäädavalt ligipääsu oma kontole.

Autentimise süsteem põhineb JSON Web Token internetistandardil. See tähendab, et server saadab kasutajale õigete sisselogimistunnuste vastuseks kodeeritud kasutajatuvastustõendi, mis on allkirjastatud avaliku ja salajase võtmega. Tõend salvestatakse veebibrauseri mällu ning edaspidi kui kasutaja päringuid teeb, lisatakse tõend automaatselt päringu päisesse. Tõendi alusel otsustatakse serveri poolel, millist teavet kasutajale väljastada võib. Tõend kustutatakse kui kasutaja välja logib või kui tema viimasest sisselogimisest on möödunud kindlaksmääratud hulk päevi.

4.2 Andmekaitse põhimõtted

GDPR, ehk *General Data Protection Regulation*, ehk Euroopa Liidu isikuandmete kaitse üldmäärus on seadusega jõustatud isikuandmete kaitsmise raamistik, mis seab Euroopa

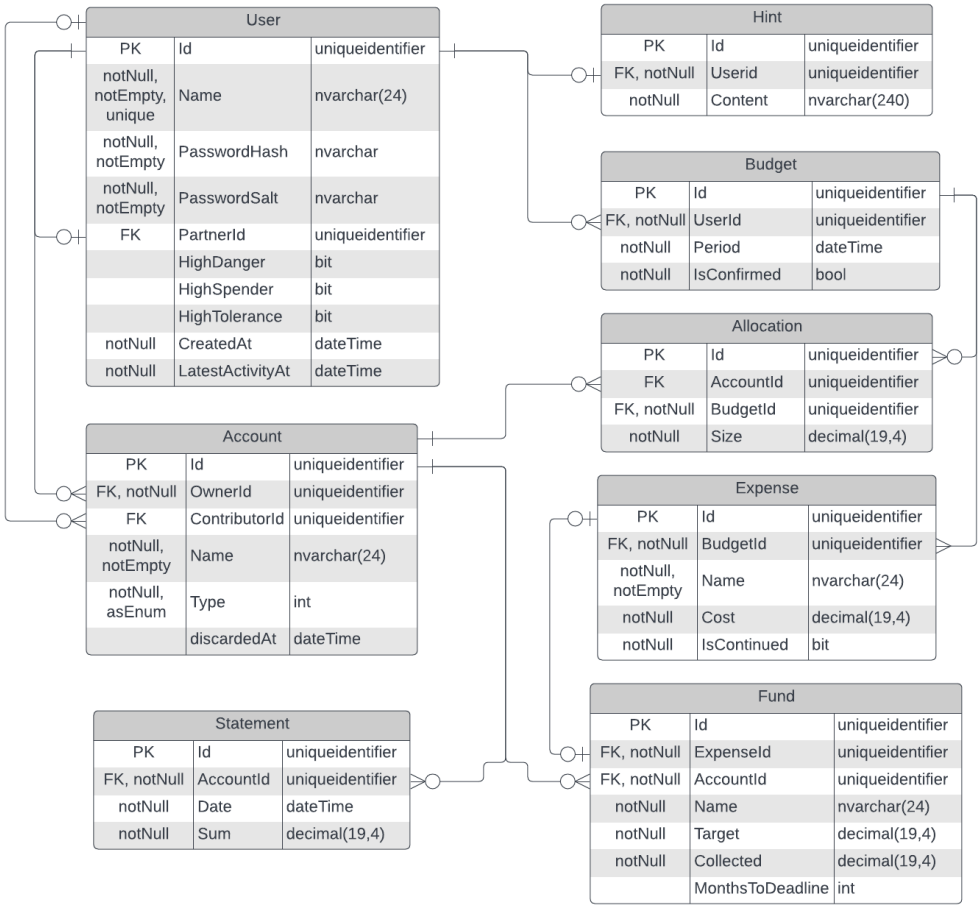
Liidus toimuvatele isikuandmete töötlemisele ja talletamisele ranged ja spetsiifilised nõuded [56].

Töö käigus loodav rakendus ei kogu kasutajate isiklikke andmeid, mistõttu ei rakendu sellele GDPR määruses kehtestatud nõuded. Sellegi poolest austab rakendus kasutaja „õigust olla unustatud“ ja võimaldab konto täielikku kustutamist.

Inaktiivsed kasutajad, kes pole oma kontole sisse loginud rohkem ühe aasta jooksul, kustutatakse.

4.3 Olemi-suhte diagramm

Rakendus kasutab andmete salvestamiseks relatsioonilist andmebaasi, mida hallatakse Microsoft SQL Serveri vahendusel. Andmebaasi ülesehitust iseloomustav olemi-suhte diagramm on kujutatud Joonisel 23.



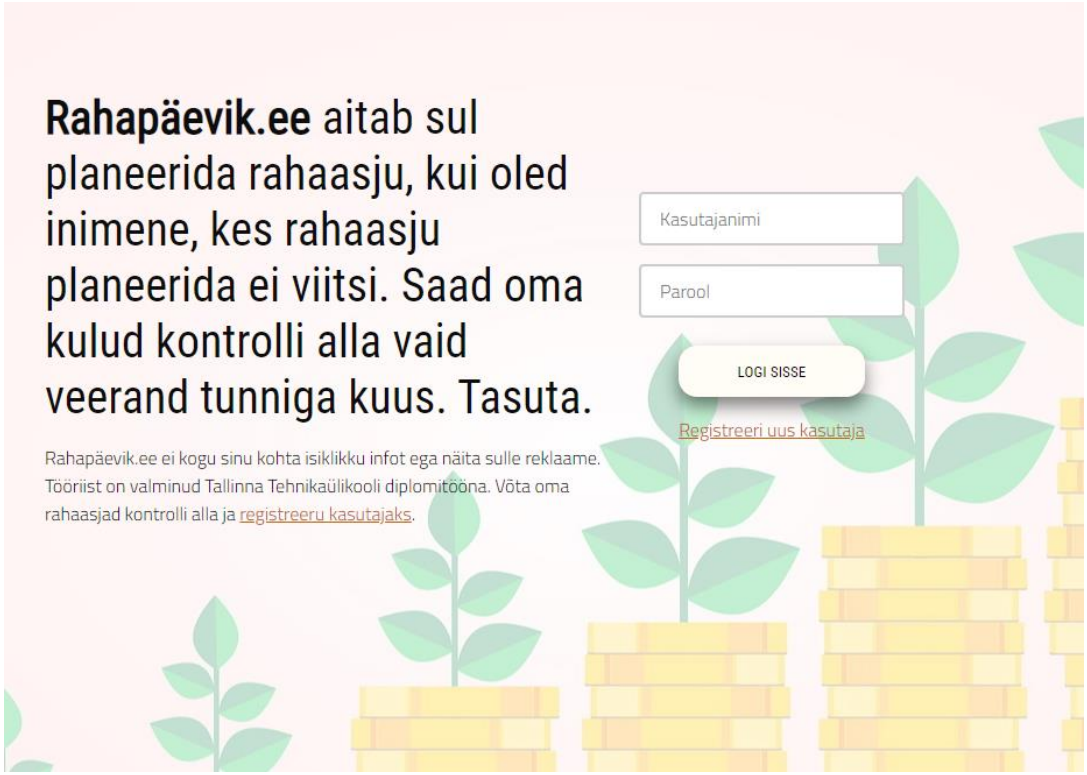
Joonis 23. Rakenduse olemi-suhte diagramm.

4.4 Kasutuskirjeldus

Veebirakenduse avamisel kuvatakse kasutajale tutvustav leht, kus võetakse paari lausega kokku, millist kasu rakendus kasutajale pakub. Kasutajal on avalehelt võimalik kas kasutajaks registreeruda või olemasoleva kontoga sisse logida.

Kõikide muude järgnevalt kirjeldatud funktsioonide kasutamiseks peab kasutaja olema sisse logitud. Rakendusse uue konto loomisel ja esimest korda sisse logides kuvatakse kasutajale automaatselt täiendavaid näpunäiteid rakenduse kasutama õppimiseks.

Kuvatõmmis avaekraanist on välja toodud Joonisel 24.



Joonis 24. Rakenduse avaekraani kuvatõmmis.

4.4.1 Kogumiskontode muutmine

Vaikimisi arvestatakse, et kasutajal on üks üldine kogumiskonto. Kasutajal on vastavale nupule vajutades võimalik kogumiskontole määrata isiklik nimi.

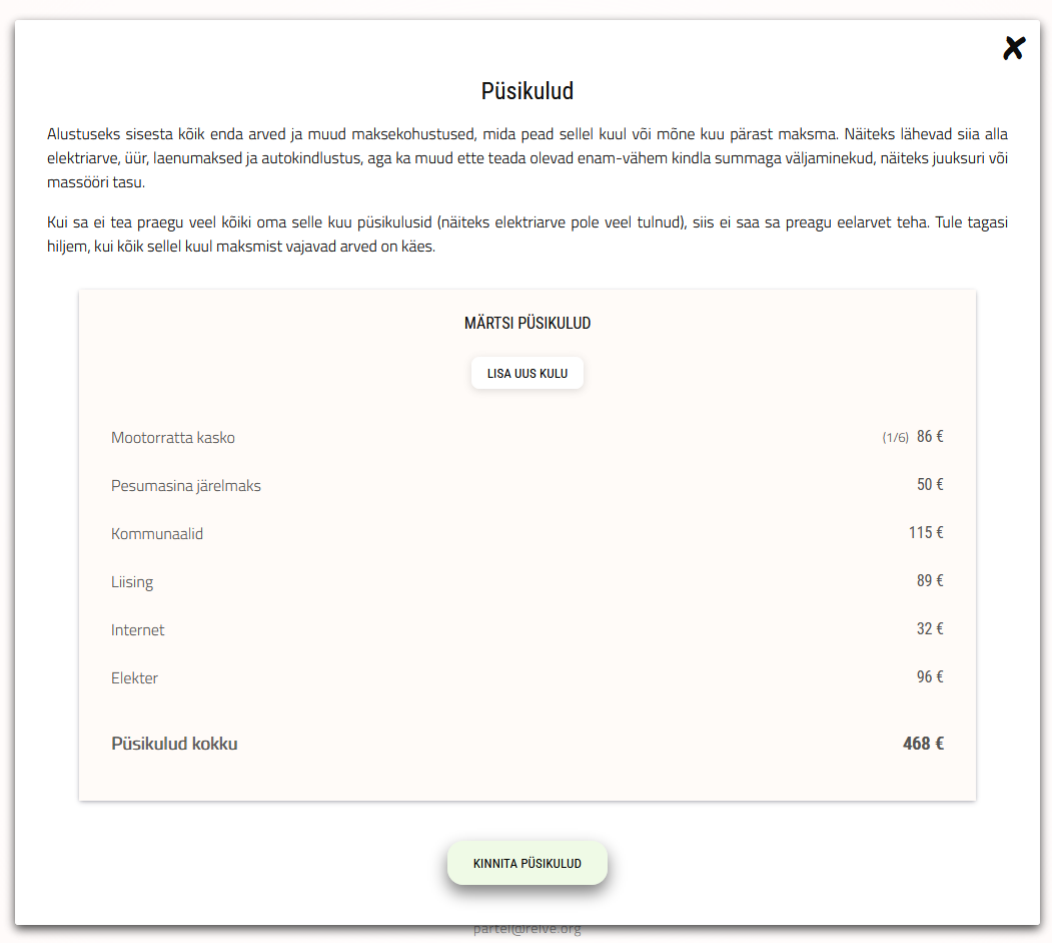
Kasutaja saab soovi korral lisada juurde uusi kogumiskontosid, näiteks reisimiseks või remondiks. Samuti saab loodud lisakontosid kustutada. Kogumiskontodele võib, aga ei pea seadma kindla summaga eesmärki.

Kasutajal on võimalus märkida üks kogumiskontodest esmaseks. Esmasesse kogumiskontosse suunab rakendus rohkem raha kui teistesse. Kasutajale automaatselt loodud esimene kogumiskonto on vaikimisi ka tema esmane kogumiskonto. Kasutaja ei pea määrama esmast kogumiskontot, mis juhul jagatakse raha kõikide kogumiskontode vahel võrdselt.

4.4.2 Eelarve koostamine

Kasutajal on võimalik vastavale nupule klõpsates alustada uue eelarve koostamist. Eelarvet saab koostada käesoleva kalendrikuu kohta, eeldusel, et eelarvet pole veel valitud kuu kohta tehtud.

Esimesena küsitakse kasutajalt kõiki tema püsikulusid, mis kantakse kokku ühte tabelisse. Tabeli lõpus tuuakse välja püsikulude summa. Püsikulude nimekirja näidis on välja toodud Joonisel 25.

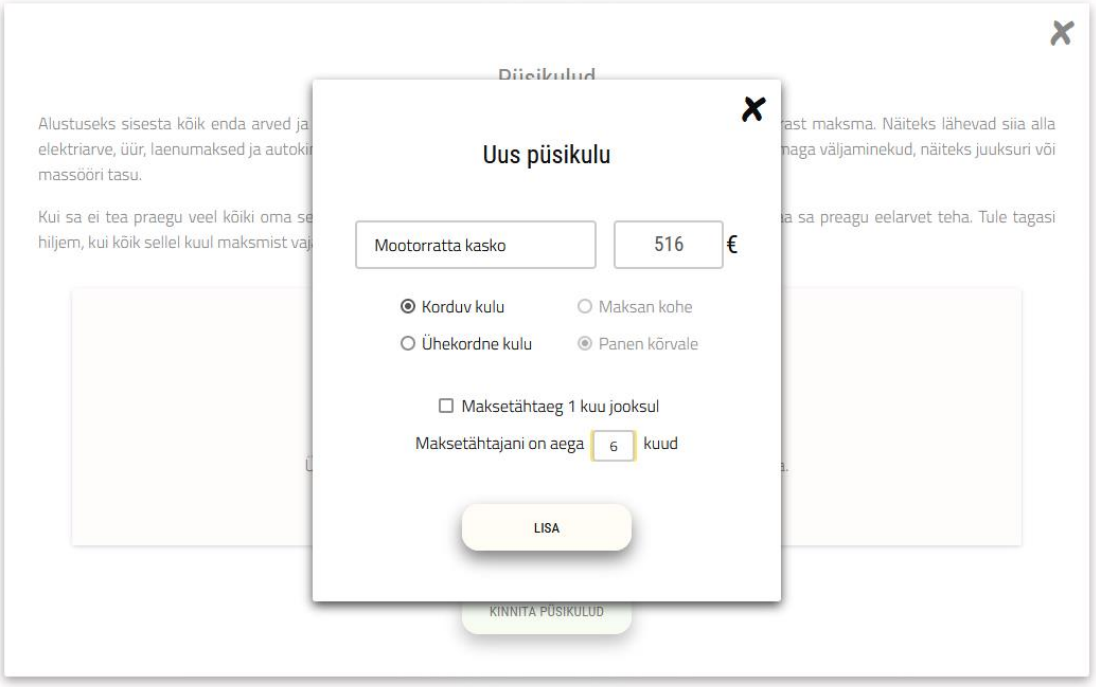


Joonis 25. Kuvatõmmis eelarve püsikulude nimekirjast.

Kasutaja märgib kulusid sisestades, kas tegemist on ühekordse kuluga või korduva kuluga. Korduvad kulud kantakse kasutaja edaspidiste eelarvete peale automaatselt, seni kuni ta püsikulu tühistab.

Samuti märgib kasutaja, kas püsikulu tuleb tasuda algava perioodi jooksul või kaugemas tulevikus. Pikema perioodi valimisel arvutatakse kulu ümber igakuisteks väiksemateks makseteks. Selliste maksete juures on püsikulude tabelis makseosa märge. Näiteks kui kasutajal tuleb teha kindlustusmakse poole aasta pärast, arvutab rakendus selle ümber kuue kuu osamakseteks ja kannab vastavale reale märke „1/6“.

Näide uue kulurea lisamisest on toodud Joonisel 26.



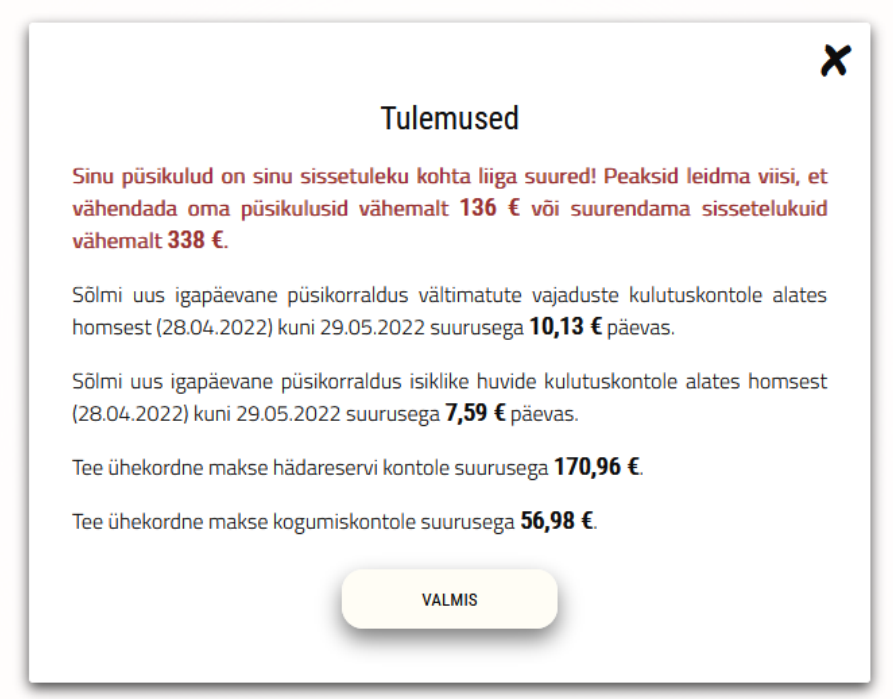
Joonis 26. Kuvatõmmis uue püsikulu lisamisest.

Kasutajal on eraldi nupust võimalik avada eelarve seaded. Seadete alt saab kasutaja teha valiku kolme näitaja osas, kusjuures iga näitaja jaoks on kolm võimalikku valikut. Näitajatel määratud valikud mõjutavad rakenduse arvutuskäiku. Kasutaja saab nendega määrata oma finantsturvalisuse taseme (mis mõjutab soovitusliku hädareservi suurust), oma elustiili eelistuse (mis mõjutab muutuvkulude jaotust) ja oma vabadusvajaduse (mis mõjutab soovituslikku püsikulude ülempiiri).

Kui kasutaja on väljad täitnud, saab ta andmed kinnitada, misjärel küsitakse kasutajalt eelarveperioodi sissetulekud. Iga sissetulek kantakse eraldi reale.

Viimaseks vaatab kasutaja üle summad, mis peaksid olema tema kontodele varasemate eelarvete põhjal kogunenud ja viib vajadusel sisse parandused. Sellega lõppeb eelarve andmete sisestamine.

Sisestatud andmete põhjal arvutab rakendus kasutajale kuueelarve. Eelarvel on märgitud, kui suure summa peaks kasutaja kandma üle oma hädareservi ja/või kogumiskontodele ning millise summaga igapäevase püsikorralduse peaks kasutaja määrama oma kulutuskontodele. Ühtlasi on eelarvel märgitud, kas püsikulud on sissetulekutega võrreldes mõistlikus suurusjärgus. Juhul kui kasutaja konto on kaaslase kontoga liidetud, kuvatakse ka märge selle kohta, kas nende püsikulud on tasakaalus. Näidis arvutatud tulemist on välja toodud Joonisel 27.



Joonis 27. Näidis sisestatud eelarve põhjal arvutatud juhistest.

4.4.1 Eelarvete ajalugu

Kasutaja saab vastava nupu alt vaadata oma varasemaid eelarveid. Varasemaid eelarveid saab soovi korral kustutada, kuid mitte muuta. Eelarvete ajaloo põhjal kuvatakse aladiagramm, kuhu on märgitud püsikulud, mõlemat liiki muutuvkulud ja kogumiseks

eraldatud summad läbi ajaloo. Näidis eelarvete ajaloo diagrammist on välja toodud Joonisel 28.

Joonis 28. Kuvatõmmis eelarvete ajaloo diagrammist.

4.4.2 Kontode haldus

Kasutaja saab vastava nupu alt vaadata oma kontode teoreetilist seisu. Siin kuvatakse summad, mis peaksid olema kasutaja püsimaksete kontol (jaotatuna konkreetsete püsikulude kaupa), hädareservis ja kogumiskontodel. Kui mõne konto teoreetiline seis ei vasta reaalsele seisule, saab kasutaja summat käsitsi parandada.

4.4.3 Kaaslase lisamine

Kasutaja saab vastava nupu alt otsida kasutajanime järgi teisi kasutajaid. Teisele kasutajale on võimalik saata kaaslaseks liitmise kutse. Kui teine kasutaja kutse vastu võtab, luuakse kontode vahele ühendus.

Ühendatud kasutajatel kuvatakse edaspidi mõlema kasutaja eelarvetel (ja diagrammil) kaaslase panust vältimatute vajaduste kulutuskontole. Ühtlasi kuvatakse edaspidi nende eelarvetel üksteise püsikulude koormust protsendina sissetulekust. Kui püsikulude koormus on kaaslaste vahel liiga erinev, soovitab rakendus püsikulude omavahelist jaotust muuta.

4.4.4 Konto haldamine

Kasutajal on võimalik vastava nupu alt muuta oma kasutajakonto seadeid, täpsemini kasutajanime, salasõna ja salasõna vihjet. Samuti saab konto halduse alt kustutada oma konto, millega kaasneb kõikide kontoga seotud andmete, sealhulgas eelarvete kustutamine.

4.5 Edasine arenguplaan

Rakendusele on plaanis lisada võimalus lubada ühendatud kontodega kasutajatel püstitada ühiseid finantseesmärke. See tähendaks ühiste kogumiskontode loomist ja jälgimist sarnasel viisil nagu on praeguses arenduses ühise muutuvkulude kontoga.

Rakendusse võiks lisada võimaluse salvestada ja kuvada rahade liikumise diagramme reaalse pangaväljavõtte alusel. See lisaks uusi võimalusi rahavoolu visualiseerimiseks. Samuti võiks see asendada vajaduse kontode halduses käsitsi korrektuure teha. Täienduse liitmiseks tuleb esmalt välja mõelda, kuidas tagada pangakonto väljavõtte töötlus selliselt, et oleks välistatud konto omaniku isiku tuvastamine.

Rakenduse kasutamiseks on broneeritud veebiaadress *rahapäevik.ee*. 2022 aasta suve jooksul on plaan käesoleva töö käigus valminud rakendus sellel aadressil üldsusele tasuta kasutamiseks avaldada. Rakenduse tulevaste kasutajate hulka kuulub ka töö autor.

5 Kokkuvõte

Lõputööle oli seatud kaks eesmärki – luua ülevaatlik võrdlus teekidest, mille abil saab Blazor raamistikus diagramme kuvada ning valmistada minimalistlik, kuid tõhus rakendus eraisiku kulude haldamiseks. Mõlemad eesmärgid said töö raames täidetud.

Töö sisust leiab tutvustava ülevaate kõikide tuntud Blazor diagrammiteekide kohta. Lisaks üldisele tutvustusele on iga teegi juures toodud välja visuaalne näidis teegiga loodud diagrammist koos diagrammi joonistamiseks kasutatud programmikoodi tööpõhimõttega. Võrdluse lõpus on koostatud kokkuvõtlik tabel võrreldud teekide eeliste ja puudustega. Valminud ülevaade on kasulikuks allikaks programmeerijatele, kes soovivad Blazor raamistikus valmistada rakendust, kus kuvatakse diagramme.

Töö käigus valmis eraisiku kulude haldamise rakendus. Rakendus erineb teistest omataolistest, sest selle eesmärgipärane kasutamine võtab väga vähe aega – vaid ühe lühikese kasutuskorra kuu jooksul. Rakenduse eripära seisneb ka selles, et see ei põhine üksikkulutuste sisse kandmisel ja jälgimisel, vaid üldise päevaeelarve arvutamisel. Töö sisus tutvustatakse Eesti pankade võimaldatud viisi rakendusega leitud päevaeelarve jõustamiseks.

Käesolev töö koos kõigi selle osadega on autori loodud. Autor on rakenduse avaldamiseks broneerinud domeeninime Rahapäevik.ee ja kavatseb rakenduse sellel aadressil lähitulevikus avaldada.

Kasutatud kirjandus

[1] MIT License, „The MIT License (MIT),“ [Võrgumaterjal]. Available: https://mit-license.org/. [Kasutatud 22 aprill 2022].

[2] DevIq, „Magic Strings,“ [Võrgumaterjal]. Available: https://deviq.com/antipatterns/magic-strings/. [Kasutatud 22 aprill 2022].

[3] K. Lust, "Pärisorjast päriskohaomanikuks: talurahva emantsipatsioon eestikeelse Liivimaa kroonukülas 1819-1915", Tartu: Eesti Ajalooarhiiv, 2005.

[4] H. Relve, "Kiviaja puudutus", Tallinn: Varrak, 2017.

[5] Rahandusministeerium, „Rahatark Eesti - Eesti elanike rahatarkuse enendamise strateegia aastateks 2021-2030,“ Tallinn, 2021.

[6] Saar Poll, „Finantskirjaoskuse ja finantsteenuste alase teadlikkuse uuring Eesti elanike hulgas: teadmised, oskused, käitumine ja hoiakud isiklike rahaasjade korraldamisel,“ Tallinn, 2012.

[7] Turu-Uuringute AS, „Eesti elanike finantskirjaoskuse ehk rahatarkuse uuring,“ 2019.

[8] A. Shavrina, „Raha paigutamise harjumused Eestis elavate eestlaste ja venelaste näitel,“ *Tallinna tehnikaülikool, majandusteaduskond, majandusanalüüsi ja rahanduse instituut, bakalaureusetöö,* 2018.

[9] G. Karman, „Finantsolukord ja rahaasjade planeerimine eesti üliõpilaste seas,“ *Tallinna tehnikaülikool, majandusteaduskond, rahanduse ja majandusteooria instituut, bakalaureusetöö,* 2016.

[10] Finantsinspektsioon, NASDAQ Tallinn AS, "Finantsaabits: Rahaasjade korraldamise käsiraamat", Tallinn: Finantsinspektsioon, NASDAQ OMX Tallinn AS, 2016.

[11] T. T. T. V. Kristjan Liivamägi, "Rahaedu põhimõtted: Kuidas haarata kontroll oma rahaasjade üle ja saavutada rahaline vabadus", Tallinn: Argo kirjastus, 2021.

[12] Forbes, „Your Guide To The 50/30/20 Budgeting Rule,“ [Võrgumaterjal]. Available: https://www.forbes.com/advisor/banking/guide-to-50-30-20-budget/. [Kasutatud 19 aprill 2022].

[13] J. Roosaare, "Rikkaks saamise õpik", Tallinn: Müügiguru OÜ, 2014.

[14] C. Laantee, „Rahaplaneerimisrakenduste kasutajaliidesed ja kasutatavus,“ *Tallinna Tehnikaülikool, infotehnoloogia teaduskond, arvutiteaduse insituut, bakalaureusetöö,* 2015.

[15] Investopedia, „Best Budgeting Apps,“ [Võrgumaterjal]. Available: https://www.investopedia.com/best-budgeting-apps-5085405. [Kasutatud 19 Aprill 2022].

[16] LHV Pank, „Hinnakiri,“ [Võrgumaterjal]. Available: https://www.lhv.ee/et/hinnakiri. [Kasutatud 19 aprill 2022].

[17] SEB pank, „Hinnakiri,“ [Võrgumaterjal]. Available: https://www.seb.ee/hinnakiri. [Kasutatud 19 aprill 2022].

[18] Swedbank pank, „Hinnakiri,“ [Võrgumaterjal]. Available: https://www.swedbank.ee/private/home/more/pricesrates?language=EST. [Kasutatud 19 aprill 2022].

[19] W. Jager, "Breaking ‘bad habits’: a dynamical perspective on habit", Groningen: University of Groningen, 2003.

[20] Microsoft, „Blazor,“ Blazor raamistiku ametlik tutvustus, [Võrgumaterjal]. Available: https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor. [Kasutatud 19 aprill 2022].

[21] E. A. Scott, "SPA Design and Architecture", New York: Manning Publications, 2015.

[22] Medium, „Why JavaScript is Popular Despite Being a Crappy Illogical Language,“ [Võrgumaterjal]. Available: https://medium.com/@trungluongquang/why-javascript-is-popular-despite-being-a-crappy-illogical-language-a8be98b20779. [Kasutatud 19 aprill 2022].

[23] Ant Design Blazor Charts, lähtekood ja dokumentatsioon, [Võrgumaterjal]. Available: https://github.com/ant-design-blazor/ant-design-charts-blazor/. [Kasutatud 19 aprill 2022].

[24] Ant Group, „G2Plot,“ G2Plot teegi tutvustus ja juhised, [Võrgumaterjal]. Available: https://g2plot.antv.vision/en/docs/manual/introduction. [Kasutatud 2022 aprill 19].

[25] Blazly, lähtekood ja dokumentatsioon, [Võrgumaterjal]. Available: https://github.com/lqdev/Blazly/. [Kasutatud 19 aprill 2022].

[26] Plotly, „Plotly JavaScript Open Source Graphing Library,“ Plotly JavaScript teegi dokumentatsioon, [Võrgumaterjal]. Available: https://plotly.com/javascript/. [Kasutatud 19 aprill 2022].

[27] Blazor ApexCharts, lähtekood ja dokumentatsioon, [Võrgumaterjal]. Available: https://github.com/apexcharts/Blazor-ApexCharts. [Kasutatud 19 aprill 2022].

[28] Blazor ApexCharts Demo, näidislehed, [Võrgumaterjal]. Available: https://apexcharts.github.io/Blazor-ApexCharts/. [Kasutatud 19 aprill 2022].

[29] ApexCharts.js, ApexCharts JavaScript teegi dokumentatsioon, [Võrgumaterjal]. Available: https://apexcharts.com/docs/installation/. [Kasutatud 19 aprill 2022].

[30] Blazorise Component Library, ametlik koduleht, [Võrgumaterjal]. Available: https://blazorise.com/. [Kasutatud 19 aprill 2022].

[31] Megabit d.o.o., „Megabit software license agreement," Blazorise kasutuslitsents. [Võrgumaterjal]. Available: https://commercial.blazorise.com/files/licences/SLA-2021-09.pdf. [Kasutatud 19 aprill 2022].

[32] Blazorise, lähtekood, [Võrgumaterjal]. Available: https://github.com/Megabit/Blazorise. [Kasutatud 19 aprill 2022].

[33] ChartJs.Blazor, dokumentatsioon ja lähtekood, [Võrgumaterjal]. Available: https://github.com/mariusmuntean/ChartJs.Blazor. [Kasutatud 19 aprill 2022].

[34] ChartJs.Blazor, „ChartJs.Blazor Client-Side Sample," näidisdiagrammid, [Võrgumaterjal]. Available: https://www.iheartblazor.com/welcome. [Kasutatud aprill 19 2022].

[35] ChartJs.Blazor, „Difficult times #160," [Võrgumaterjal]. Available: https://github.com/mariusmuntean/ChartJs.Blazor/issues/160. [Kasutatud 19 aprill 2022].

[36] GrapeCity, „ComponentOne," ComponentOne koduleht, [Võrgumaterjal]. Available: https://www.grapecity.com/componentone/. [Kasutatud 19 aprill 2022].

[37] GrapeCity Inc., „End-User License Agreement For GrapeCity ComponentOne Software," ComponentOne kasutuslitsents. [Võrgumaterjal]. Available: https://global-cdn.grapecity.com/en/eula/c1/20220315b-c1-eula.pdf. [Kasutatud 19 aprill 2022].

[38] GrapeCity, „Blazor Overview," ComponentOne dokumentatsioon, [Võrgumaterjal]. Available: https://www.grapecity.com/componentone/docs/blazor/online-blazor/overview.html. [Kasutatud 19 aprill 2022].

[39] Developer Express Inc., „DevExpress ASP.NET," DevExpress ASP.NET teegikogumik, [Võrgumaterjal]. Available: https://www.devexpress.com/products/net/controls/asp/. [Kasutatud 20 aprill 2022].

[40] Developer Express Inc., „DevExpress Blazor UI Components End-User License Agreement," [Võrgumaterjal]. Available:

https://www.devexpress.com/support/eulas/blazor.xml. [Kasutatud 20 aprill 2022].

[41] Developer Express Inc., „Blazor UI Documentation,“ DevExpress Blazor teegi dokumentatsioon, [Võrgumaterjal]. Available: https://docs.devexpress.com/Blazor/400725/blazor-components. [Kasutatud 20 aprill 2022].

[42] Infragistics Corporation, „Infragistics Product Pricing,“ [Võrgumaterjal]. Available: https://www.infragistics.com/how-to-buy/product-pricing. [Kasutatud 21 aprill 2022].

[43] Infragistics Corporation, „Infragistics Ultimate Software License Agreement,“ Ignite UI kasutajatingimused, [Võrgumaterjal]. Available: https://www.infragistics.com/legal/license/igultimate-eula. [Kasutatud 21 aprill 2022].

[44] Infragistics Corporation, „Terms of Use,“ [Võrgumaterjal]. Available: https://www.infragistics.com/legal/terms-of-use. [Kasutatud 21 aprill 2022].

[45] Infragistics Corporation, „Blazor Data Visualization Tools,“ Ignite UI Blazor teegi dokumentatsioon, [Võrgumaterjal]. Available: https://www.infragistics.com/products/ignite-ui-blazor/blazor/components/general-getting-started. [Kasutatud 21 aprill 2022].

[46] MudBlazor, „MudBlazor - Blazor Component Library,“ MudBlazor ametlik koduleht, [Võrgumaterjal]. Available: https://www.mudblazor.com/. [Kasutatud 22 aprill 2022].

[47] MudBlazor, „Explore MudBlazor,“ Mudblazor dokumentatsioon, [Võrgumaterjal]. Available: https://www.mudblazor.com/docs/overview. [Kasutatud 22 aprill 2022].

[48] Syncfusion Inc., „Buy Blazor Components,“ Syncfusion teegi hinnakiri, [Võrgumaterjal]. Available: https://www.syncfusion.com/sales/products/blazor. [Kasutatud 22 aprill 2022].

[49] Syncfusion Inc., „Syncfusion Essential Studio License,“ [Võrgumaterjal]. Available: https://origin2.cdn.componentsource.com/sites/default/files/resources/syncfusion/551546/syncfusion_license.pdf. [Kasutatud 22 aprill 2022].

[50] Syncfusion Inc., „Syncfusion downloads,“ [Võrgumaterjal]. Available: 24.

[51] Syncfusion Inc., „Introduction Documentation to Blazor Component Library,“ Syncfusion teegi dokumentatsioon, [Võrgumaterjal]. Available: https://blazor.syncfusion.com/documentation/introduction. [Kasutatud 24 aprill 2022].

[52] Progress Software Corporation, „Buy UI for Blazor | Telerik,“ Telerik hinnakiri, [Võrgumaterjal]. Available: https://www.telerik.com/purchase/blazor-ui. [Kasutatud 24 aprill 2022].

[53] Progress Software Corporation, „End user license agreement for Telerik UI,“ [Võrgumaterjal]. Available: https://origin2.cdn.componentsource.com/sites/default/files/resources/progress-telerik/778216/telerik-blazor-ui-sep2021-eula.pdf. [Kasutatud 24 aprill 2022].

[54] Progress Software Corporation, „Blazor Chart Overview,“ Telerik teegi dokumentatsioon, [Võrgumaterjal]. Available: https://docs.telerik.com/blazor-ui/introduction. [Kasutatud 24 aprill 2022].

[55] Progress Software Corporation, „First Steps with Client-Side UI for Blazor,“ Telerik paigaldusjuhis, [Võrgumaterjal]. Available: https://docs.telerik.com/blazor-ui/getting-started/client-blazor#add-the-telerik-nuget-feed-to-visual-studio. [Kasutatud 24 aprill 2022].

[56] Euroopa Parlament ja Euroopa Liidu Nõukogu, „GDPR,“ [Võrgumaterjal]. Available: https://eur-lex.europa.eu/legal-content/ET/TXT/PDF/?uri=CELEX:32016R0679&from=EN. [Kasutatud 25 aprill 2022].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Pärtel Relve

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Eraisiku kulude haldamise rakenduse loomine Blazor raamistikus“, mille juhendaja on Lembit Viilup
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

16.05.2022

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.