TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Sander-Sven Annula 164473 IABB

LIKVIIDSUSE PLANEERIMISE RAKENDUSE MIGREERIMINE VEEBI — ANDMETE SISESTAMISE VAADE

Bakalaureusetöö

Juhendaja: Deniss Kumlander

Doktorikraad

Tallinn 2019

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: Sander-Sven Annula

20.05.2019

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on ettevõtte Unit4 töölauarakenduse Cash Flow Planning: Liquidity Plan Data Input vaate loomine veebirakendusele.

Eesmärgi realiseerimiseks kasutas autor esikomponendi arenduseks Angular'i raamistikku, programmiliideste loomiseks ASP.NET Web API 2.0 raamistikku ning andmebaasisüsteemi Microsoft SQL Server. Loodud vaate põhikomponendiks olev tabel on loodud kasutades Handsontable poolt pakutud arvutustabeli raamistikku.

Antud töö annab ülevaate Cash Flow Planning rakendusest ning tehnoloogiatest, mida autor kasutas eesmärgi saavutamiseks. Samuti kirjeldab autor eesmärgi saavutamiseks läbitud protsesse nagu analüüs, esikomponendi arendus, tagakomponendi arendus ja turvalisuse realiseerimine. Töö viimases peatükis kirjeldab autor valminud vaadet ning analüüsib tehtud tööd ning tehnoloogia valikut.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 25 leheküljel, 5 peatükki, 10 joonist, 1 tabelit.

Abstract

Migrating Liquidity Plan Application to Web — Data Input View

The goal of this thesis is to migrate Unit4 desktop application Cash Flow Planning: Liquidity Plan Data Input view to web application view.

To reach this goal, user has used Angular framework for front-end development, ASP.NET Web API 2.0 framework for creating API's and database system Microsoft SQL Server. The main component of the view under development is a table, which has been created by using Handsontable spreadsheet grid.

Current thesis gives overview of Cash Flow Planning application and technologies which user has used to reach its goals. Author also has described processes for reaching its goal such as analyzes, front-end development, back-end development and realization of security. In the last part of this thesis, author has described final view and analyzes the developemnt and technologies he had chosen.

The thesis is in estonian and contains 25 pages of text, 5 chapters, 10 figures, 1 tables.

Lühendite ja mõistete sõnastik

Angular	Tarkvaraarenduse platvorm
API	*Application Programming Interface* ehk rakendustarkvara liides
ASP.NET	.NET tarkvaraplatvormi laiendav tööriistade ja ressurside kogum, mis on mõeldud arendamaks veebirakendusi
CSS	Kaskaadlaadistik, mida kasutatakse veebilehe kujundamisel
C#	Objekt-orienteeritud programmeerimiskeel
HTML	Hüperteksti märgistuskeel, milles märgendatakse veebilehti
HTTP	Hüperteksti edastusprotokoll
JSON	Andmevahetusvorming
JWT	JSON Web Token, standardne JSON struktuuriga veebitõend
Likviidsus	Maksevalmidus ehk vara muudetavus rahaks või muudeks varadeks.
LINQ	Language Integrated Query, .NET raamistiku laiendav andmepäringukeel
.NET	Microsoft'i tarkvaraplatvorm
Npm	Node Package Manager
REST	Tarkvaraarhitektuur, milles kasutatakse kindlaks määratuid päringuid
RFC	Interneti standard
SQL	Structured Query Language, andmebaasi päringukeel
Token	Veebitõend, millega saab kasutaja identsust kontrollida

Sisukord

Jooniste loetelu

Tabelite loetelu

1. Sissejuhatus

Tänapäeva tehnoloogia on kiiresti arenev. Sellega kaasnevalt koguvad suurt populaarsust just veebirakendused ning töölaua rakenduste kasutamine on harvjuhus. Töölauarakenduste puhul kasutatakse üldiselt suuremahulisi rakendusi või rakendusi millega tegeletakse igapäevaselt näiteks ettevõtte siseselt. Küll aga on ettevõtte siseselt uue töölauarakenduse kasutusele võtmine küllaltki aeganõudev protsess, kuna nende kasutamiseks tuleb läbi viia erinevaid installeerimisi ning samuti võib tekkida ühilduvuse probleeme. Sellega kaasnevalt võib tekkida juhus, kus uus tarkvara ei jõua kliendini, kuna puudub teadmine, kas rakendus on üldse sobilik antud ettevõttele ning ei soovita võtta seda riski. Vältimaks antud olukorda, oleks hea lahendus luua töölauarakendusest ka veebirakendus. Veebirakenduse puhul ei ole vajalikud erinevad installeerimised ja ei teki ka ühilduvuse probleeme, luues ideaalse keskkonna ettevõtetele, kes soovivad alustada uue rakenduse kasutamist.

Viies töölauarakenduse ka veebi, tagab see ligipääsu kliendile erinevatelt seadmetelt ilma oluliste installeerimisteta. Antud lahendus võimaldab kliendil pääseda kiirelt ligi rakendusele olenemata asukohast, piisab vaid internetiühendusest. Samuti veebirakendus võimaldaks kliendil kasutada rakendust nii tahvelarvutitest, kui ka nutitelefonidest.

1.1 Taust ja probleem

Unit4 üks pakutavatest rakendustest on „UNIT4 Financial Performance Management“, mille abil on kliendil võimalik jälgida ja automatiseerida raha planeerimise protsesse. Antud rakendus on ettevõttel realiseeritud Windows platvormile, mis töötab töölaua rakendusena. Kuna antud rakendus on kasutatav vaid arvutites, kuhu see rakendus on installeeritud, tekitab see ebamugava olukorra klientidele juhul, kui puudub ligipääs sellele seadmele. Lisaks rakenduse installeerimisele tekivad probleemid ka juhul, kui kasutaja operatsioonisüsteem on erinev kui Windows. Et kasutada antud rakendust näiteks Mac operatsioonisüsteemiga arvutil, tuleb installeerida veel lisavahendeid, nagu näiteks Virtual Machine, et kasutada antud rakendust [1].

Samuti tekivad probleemid rakenduste uuendamisel. Juhul kui kliendil ei ole automaatsed uuendused aktiveeritud, peab töölaua rakenduse kasutaja uuendama installeeritud rakendust kõikides oma arvutites.

Kui rakendus viia üle ka veebiplatvormile, on ligipääs rakendusele lihtsam ja kättesaadav kliendile olenemata tema asukohast, kasutades ükskõik mis erinevaid seadmeid ilma vajalike installeerimisteta. Lisaks sellele välistab see kohustuse rakenduse kasutajal uuendamast rakendust, kuna piisab vaid ühekordsest uuendamisest süsteemi haldaja poolt.

1.2 Eesmärk ja oodatud tulemus

Käesoleva bakalaureusetöö eesmärk on ettevõtte Unit4 töölauarakenduse Cash Flow Planning: Liquidity Plan Data Input vaate loomine veebirakendusele. Vaatega kaasnevalt on eesmärgiks kõik töölaua rakenduse vaate funktsionaalsused realiseerida ka veebirakenduse vaates. Kuna kõik vaate elemendid põhinevad andmebaasi andmetele, siis luuakse programmiliidesed, mille kaudu on võimalik teha päringuid andmetele andmebaasist. Lisaks vaate loomisele on eesmärk tagada veebirakenduse turvalisus kasutades veebi *token'it* ehk veebitõendit millega on võimalik kasutaja identsust kontrollida.

2 Ülevaade rakendusest ja kasutatud tehnoloogiatest

Käesolev peatükk kirjeldab lähemalt eksisteerivat rakendust ja sellega kaasnevat likviidsusplaani andmete sisestamise vaadet, mida lõputöö käigus veebirakendusele luuakse. Lisaks tehakse ülevaade autori poolt kasutatud tööriistadest ja tehnoloogiatest arendustegevuse jaoks.

2.1 Unit4 Cash Flow Planning rakendus

Unit4 Cash Flow Planning rakendus kasutab üksikekraani keskkonda, et koostada andmete aruandlust ilma keeruliste protsesside või erinevate valikuteta. Finantsaruandluse loomise vahend on standardiseeritud kasutusega, kuid siiski on tegu paindliku süsteemiga koos kasutajasõbraliku lohistamise funktsionaalsuse ja automaatse graafilise kujutlemisega. Antud rakendusega on võimalik anda otsene kontroll varade- ja finantsüksusele grupi likviidsusprognoosi, raha netopositsiooni ja valuutakursi üle. Samuti on süsteemil sisseehitatud efektiivsuse aruandlus koos graafikutega, mis näitab kiirelt ära valuutariskid ja avatud positsioonid vabastades kasutaja kohustusest teha rutiinset aruandlust[2].

Antud rakendusel on ka kasutajasõbralik importimise moodul. Standariseeritud importimise visandid nagu SWIFT MT940 muudavad panga bilansi ja muud raha planeerimisandmed automaatseks, samas olles kuluefektiivne. Samuti on võimalik kasutada seda kui on kasutusel erinevad pangad koos erinevate raamatupidamissüsteemidega[2].

Lisaks on antud tarkvaraga võimalik analüüsida ja võrrelda prognoose kohalikus ja võõras valuutas, leides erinevusi kuude, nädalate ja päevade lõikes, võimaldades kasutajal saada kiire ülevaade ja arusaam rahavoogudest. Samuti tagab antud rakendus lihtsad ja selged graafiliselt esitatud prognoosid, mis aitavad ettevõtetel lahendada likviidsuse prognoosimisega tekkivaid raskusi[2].

Antud rakendusel on kolm peamist kasutust: sisestamine ja protsessi kontroll, valuutariski ja riskimaandamise efektiivsus, aruandluse tegemine ja analüüsimine.

Sisestamise ja protsessi kontrolli abil on võimalik:

- näha detailselt raha aruandlust vastavalt valuutadele ja pangakontodele;
- vaadata nii päevade, nädalate kui ka kuude kaupa;
- piiramatu arv versioone nagu prognoos, tegelik, eelarve;
- reeglipõhine teavitussüsteem, mis suunatleb kasutajat lisama selgitusi ja kirjeid, kui on prognoosis tegu suuremate muudatustega;
- visuaalse oleku aruandlus, tagades protsessi täieliku kontrolli[2].

Valuutariski ja riskimaandamise efektiivsus tagab:

- sisseehitatud loogika arvutamaks valuutariski valitud ettevõttes või grupis;
- täieliku läbipaistvuse ettevõttele, valuutapaaridele ja prognoositud raha tehingutele;
- standardiseeritud aruanded prognooside maandamise või tegelike tehingute riskide kohta koos graafikute ja analüüsidega;
- grupi tasemel maandatud tehingute sisestamine ja väljastamine[2].

Aruandlus ja analüüsimine võimaldab kasutada:

- standardiseeritud visandeid aruandluse tegemiseks koos graafikute ja joonestikuga;
- erinevate struktuuride rahade automaatset konsolideerimist;
- aruannete koostamist vabalt valitud valuutas;
- aruannete avalikustamist koos ligipääsuga vastavalt kasutajaõigustele[2]

Käesolevas bakalaureusetöös loodav vaade rahuldab väljatoodud rakenduse esimest kasutuse juhtumit- sisestamine-, tagades võimalikkuse näha aruandlust vastavalt valuutadele ja pangakontodele, vaadata nii päevade, nädalate kui ka kuude kaupa ning vaadata andmeid vastavalt valitud versioonile.

2.2 Töövahendid ja tehnoloogiad

Autor valis esikomponendi arenduseks Angular’i, mis on TypeScripti põhine avatud lähtekoodiga raamistik ja platvorm veebirakenduste arendamiseks. Angular põhilised arenduse osad on NgMoodulid, mis on Angulari enda moodulite süsteem. Need koosnevad ühtlustatud koodi plokkidest, mis on spetsialiseeritud rakenduse domeenile, töövoole või lähedaselt seotud võimaluste kogumile[3].

Autor valis antud raamistiku, kuna tegu on väga populaarse raamistikuga, ning autor tahtis teha sellega tutvust. Lisaks sellele valis autor Angular raamistiku, kuna alates Angular 5'st pakub raamistik ka kasutajaliidese komponente Angular Material, mida on mugav ja lihtne kasutada ilmestamaks veebirakendust[4].

Kaasnevalt Angulari kasutamisega, kasutas autor Angular'i käsurea kasutajaliidest Angular CLI, mille abil on lihtne genereerida ja ehitada Angular rakendusi[5].

Rakenduse programmiliidesed on loodud kasutades ASP.NET Web API 2.0 raamistikku, mis võimaldab ehitada lihtsaid HTTP teenuseid, mida saavad kasutada nii brauserid kui ka mobiilseadmed[6]. ASP.NET on arendusplatvorm, mis laiendab .NET raamistiku. Antud raamistikuga on võimalik kirjutada programmeerimiskeeltes nagu Visual Basic, F# ja C#[7]. Antud lõputöö raames on kasutusel objektile orienteeritud programmeerimiskeel C#, mis oli ettevõtte Unit4 üks omapoolsetest nõudmistest.

Projekti raames kasutatud andmebaas Microsoft SQL Server on Microsofti poolt loodud relatsiooniline andmebaas[8]. Antud andmebaasisüsteem on kasutusel ka töölauarakenduse versioonil ning sellega kaasnevalt kasutas autor seda enda projekti tegemisel. Samuti oli ka antud andmebaasisüsteemi kasutamine ettevõtte Unit4 üks omapoolsetest nõudmistest. Andmebaasi keskkonna haldamiseks kasutas autor Microsoft SQL Server Management Studio't.

Kuna andmete sisestamise vaate peamine element on suures mahus tabel koos erinevate funktsioonidega, siis otsustas autor kasutada Handsontable komponenti, mis on kirjutatud kasutades JavaScript'i ja töötab erinevate raamistikega nagu Angular, Vue ja React[9].

Handsontable'i valis autor peamiselt erinevate funktsioonide tõttu, mis loodavas vaates kasuks tulevad, nagu näiteks ridade ja veergude pealkirjad, ridade peitmine ja valemite plugin[10]. Samuti on antud komponendil põhjalik dokumentatsioon ning aktiivne veebipõhine tugi, kus on võimalik esitada küsimusi komponendi kohta ja teavitada vigadest[9]. Lisaks sellele valis autor Handsontable, kuna viimase aastaga on Handsontable versioon neljalt jõutud versioon seitsmeni, mis näitab pidevat ja kiiret arengut antud komponendil, mille viimane versioon 7.0.3 lasti välja 13.mai 2019[11]. Samuti oli üheks plussiks see, et antud komponent on saadaval tasuta litsentsiga mittekaubanduslikel eesmärkidel kasutamiseks[12].

Veebirakenduse turvalisuse tagamiseks kasutas autor rakenduse siseseks autoriseerimiseks JSON Web Token'it ehk JWT. JWT on standardne (RFC 7519) viis turvalise andmevahetuse tagamiseks erinevate osapoolte vahel. JWT saab kasutada nii autoriseerimiseks, kui ka informatsiooni vahetuseks. Autoriseerimise puhul lisatakse JWT igale päringule tagades kasutajale ligipääs vaid lubatud ressurssidele. Turvalise informatsiooni vahetuse jaoks kasutatakse allkirjastamist kasutades avaliku või privaatse võtme paare[13].

Rakenduse tagakomponendi arenduseks kasutas autor Microsoft Visual Studio integreeritud arenduskeskkonda ja esikomponendi arenduseks koodi redigeerijat Visual Studio Code. Projekti versioonihalduseks on autor kasutanud hajutatud versioonihaldustarkvara Git[14], ning veebipõhist hoiustamise keskkonda GitLab[15]. Rakenduse tagakomponendi poolt loodud programmiliideste testimiseks ja katsetamiseks kasutati rakendust Postman, mis on abivahend RESTful API'de testimiseks[16].

3 Arendustegevus

Käesolev peatükk kirjeldab lähemalt arendustegevuse protsesse nagu analüüs, kasutajaliidese ja tagakomponendi arendamine ning turvalisuse realiseerimine.

3.1 Rakenduse analüüs

Rakenduse arendus algas esmalt analüüsi protsessiga, kus autorile tehti demo üldisest rakendusest ja likviidsuse plaani andmete sisestamise vaatest, mida peaks veebirakenduses realiseerima. Ettevõtte Unit4 poolt loodi autorile Virtual Image, mille abil sai autor ligipääsu rakendusele personaalsest arvutist.

3.1.1 Funktsionaalsed nõuded

Esmalt pani autor paika funktsionaalsed nõuded vaatele. Funktsionaalsed nõuded pärinesid töölauarakenduse funktsionaalsustest.

Funktsionaalsed nõuded:

- Kasutaja peab saama valida ettevõtte, versiooni ja ajaperioodi alguse millega seoses soovib kasutaja andmeid sisestada.
- Kasutaja peab saama filtreerida kontode tüüpe, mida kuvatakse andmete sisestamise vaates.
- Kasutaja peab saama filtreerida andmeid valuuta ja pangakontode põhjal.
- Kasutaja peab saama lisada andmeid vaid väljadele, mis kuuluvad sisestamiseks.
- Kasutaja peab saama salvestada sisestatud andmeid.
- Tabeli esmasel avamisel peab olema kuvatud kõikide rahavoogude summa valuutas, millega on ettevõtte seotud.
- Kasutaja peab saama vaadata andmeid nii päevade, nädalate kui ka kuude kaupa.
- Kasutaja sisestatud andmete põhjal peavad olema tagatud arvutused kindlate kontode väärtusteks.

Kuna enamus nõuetest on realiseeritavad vaate põhikomponendi- tabeli- kaudu, siis tegi autor funktsionaalsete nõuete baasil kriteeriumid, mida peab arvestama tabeli komponendi valimisel. Kriteerimuid olid järgnevad :

- Tabel peab võimaldama ridade peitmist.
- Tabeli andmete muutmine peab olema lihtne.
- Tabel peab võimaldama muuta lahtrid vaid lugemiseks.
- Tabeli andmeid on lihtne kokku koguda.
- Tabeli andmete põhjal on võimalik sooritada arvutusi.

Lähtudes antud kriteeriumitest otsustas autor kasutada Handsontable poolt pakutud arvutustabeli komponenti, kuna antud komponendil on kõik vajalikud funktsionaalsused olemas. Ridade peitimise võimaldamiseks on Handsontable funktsioon *HiddenRows*, mis peidab defineeritud read. Tabeli andmete muutmiseks piisab selle andmeallika väärtuste muutmisest ning vastavalt muudatustele kuvatakse uued väärtused automaatselt. Tagamaks vajaduse määrata lahtrid vaid lugemiseks, saab panna Handsontable lahtritele omaduse *ReadOnly.* Tabeli andmete kogumiseks piisab vaid Handsontable meetodist *GetData* ning tabeli andmete põhjal on võimalik arvutusi sooritada kasutades pluginit *Formulas.* Lisaks nende oluliste kriteeriumite täitmisele võimaldab Handsontable nii ridadadele kui ka veergudele lisada pealkirju, mille abil on lihtne kuvada ridadele konto nimesid ja veergudele kuupäevi.

3.2 Kasutajaliidese arendamine

Järgnevas peatükis kirjeldatakse lähemalt kasutajaliidese projekti loomist ja sellega kaasnenud struktuuri ning üldist arendusprotsessi andmete sisestamise vaate loomiseks.

3.2.1 Struktuur

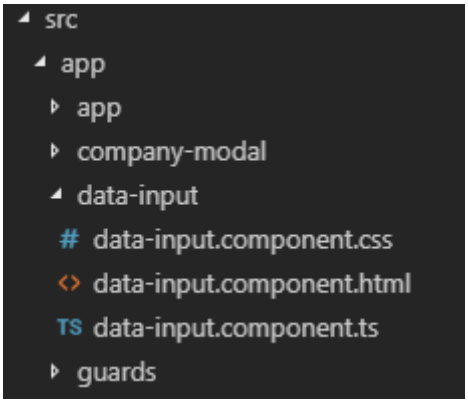
Käesoleva projekti esikomponendi realiseerimiseks kasutas autor Angular raamistikku, mis on täpsemalt kirjeldatud peatükis 2.2. Projekti loomiseks kasutas autor Angular CLI käsku:

```
ng new Unit4Web
```

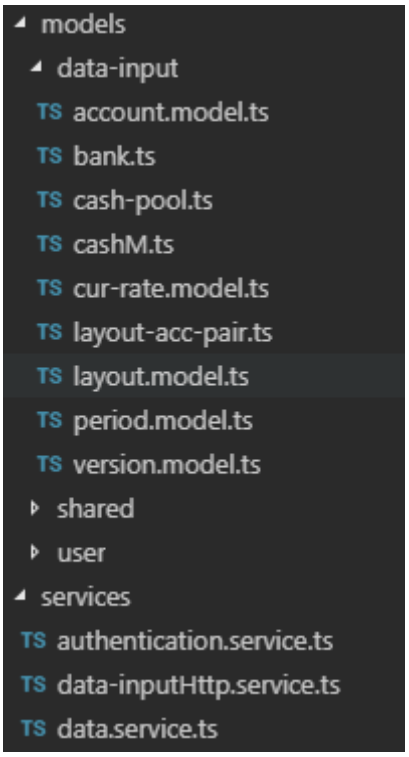
Antud käsk loob Angular'i tööruumi, põhikomponendi App ja Angular rakenduse nimega „Unit4Web". Tänu Angular CLI'le lisatakse rakendusele kõik vajalikud npm pakid ja muud olulised sõltuvused Angular rakenduse jaoks[18]. Rakenduse komponentide, teenuste ja mudelite genereerimiseks on kasutatud Angular CLI käsku:

```
ng generate
```

Antud käsuga luuakse vastavale loodud elemendile ka testimise fail. Genereeritud komponentidele lisatakse nii CSS fail antud komponendile kui ka HTML fail. Joonisel 1 on kujutatud vaate peakomponendi, *data-input*, genereerimisest tekkivaid faile ning joonisel 2 on kujutatud mudelite ja teenuste faile.



Joonis 1. Esikomponendi komponentide failistruktuur näidatuna Visual Studio Code koodiredigeerijas



Joonis 2. Esikomponendi andmemudelite failistruktuur näidatuna Visual Studio Code koodiredigeerijas

Services kaustas sisalduv *authentication service* tegeleb esikomponendi poolse autentimisteenusega. Kõik HTTP teenused, mis on kasutusel andmete sisestamise vaates on grupeeritud kokku *data-inputHttp* teenuse failis. *Data* teenuse fail sisaldab endas teenuseid, mida kasutavad eri komponendid andmete vahetamiseks. Samuti on andmemudelite klassid eraldi struktureeritud vastavalt nende kasutusele.

3.2.2 Andmete sisestamise vaade

Tagamaks funktsionaalne nõue: võimalus valida ettevõte, versioon ja ajaperioodi algus, mille kohta andmeid sisestada, kasutas autor Angular moodulit MatDialog. Antud moodul võimaldab kuvada kasutajale *pop-up* vaadet, milles kasutaja saab oma valikut teha andmete sisestamise jaoks.

Andmete sisestamise vaate jaoks loodi komponent nimega *data-input*. Antud vaate üheks peamiseks elemendiks on tabel, mille loomiseks on kasutatud Handsontable komponenti, mis on ka kirjeldatud peatükis 2.2. Antud komponendi andmeallikaks on massiiv, mis sisaldab endas iga rea kohta eraldi massiivi, mille iga element massiivis kujutab veeru väärtust[17].

Kontode sorteerimise funktsionaalsuse realiseerimiseks kasutas autor Handsontable pluginat *HiddenRow*. Kuna antud plugin kasutab väärtusteks rea indekseid, lõi autor massiivi, mis koosnes rea indeksitest, kontode unikaalsete numbrite ja kontode põhjal defineerimaks, mis kontod kuuluvad erinevatele kontode paigutusviiside alla. Konto paigutuse valimisel saab muutuja *hiddenRow* väärtused loodud massiivist paigutuse põhjal ja peidetakse read, mis ei kuulu valitud kontode paigutusse.

Kuna üheks funktsionaalseks nõudeks andmete sisestamise vaatel oli olemasolevate andmete põhjal kalkulatsioonid, kasutas autor Handsontable poolt pakutud pluginat *Formulas*. Kasutamaks seda pluginat, peavad valemid olema sisestatud tabeli andmete allikaks. Kuna valemid põhinevad relatiivsetel tabeli lahtrite viidetel nagu A1, $A1, A$1 või A1[19], tegi autor funktsiooni millega loodi massiivide massiiv kontode põhjal, millega on vaja teha arvutusi ja millega mitte.

Kuigi kasutusele võetud *Formulas* plugin funktsioneeris nagu oodata võis, siis autor pidi pettuma selle jõudluses. Kui autor kasutas antud pluginat kõikide tabeli lahtrite põhjal aastapikkuse perioodi jaoks, siis tabeli laadimine valemitega võttis väga kaua aega. Selle põhjal muutis autor kuvatud perioodi pikkuse 31 päeva pikkuseks ning sellega kaasnevalt paranes jõudlus tunduvalt, kuid siiski ei olnud see piisavalt kiire. Autor katsetas erinevaid viise, mida Handsontable meeskond on soovitanud jõudluse parandamiseks nagu näiteks tabeli konstantse suuruse määramine, kuid polnud kasu ka nendest[20]. Autor tegi ka kindlaks, et põhjustajaks on *Formulas* plugina kasutamine. Eemaldades plugina realiseerimise, oli tulemuseks vaid mõne millisekundi pikkune tabeli laadimine.

Lõpuks otsustas autor eemaldada valemid, mis arvutasid *opening balance* konto väärtust eelneva päeva *closing balance* väärtusest, kuna just need tekitasid kõige suurema ooteaja ning leppis sellega, et tabeli laadimine võtab ligikaudu viis sekundit. Kompenseerimaks antud valemite kasutust *opening balance* kontol, realiseeris autor selle kuvades vaid sisestamise järgsel päeval *opening balance* väärtust.

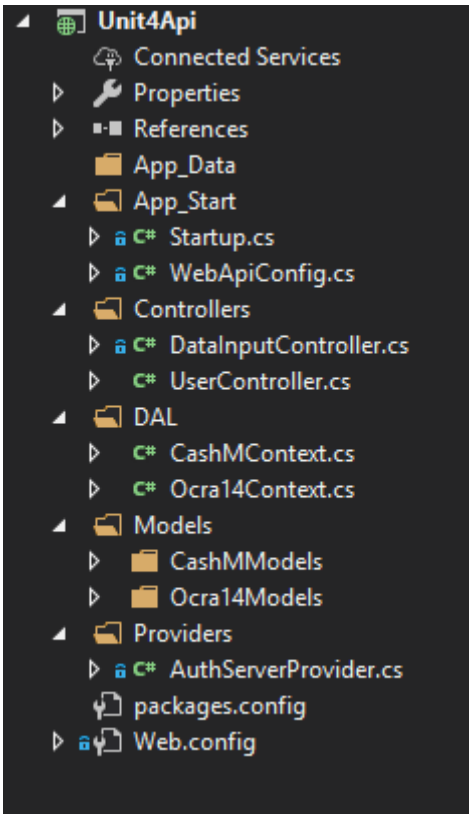
Funktsionaalsus „sisestatud andmete filtreerimine valuutade ja pangakontode põhjal“ realiseeriti kasutades tabeli andmeallika seondumist loodud massiivide massiiviga. Filtri realiseerimiseks uuendas autor vastavalt filtrile massiivi sobilike andmetega, mille tulemusel muutusid ka väärtused tabelis.

Sisestatud andmete salvestamiseks kasutas autor Handsontable meetodit *getData*. Antud meetod tagastab sisestatud andmed samasuguse massiivide massiiviga nagu kasutab Handsontable andmete allikaks. Sobitamaks saadud väärtused kontodega ja unikaalsete perioodi väärtustega lõi autor massiivi, kuhu oli lisatud iga andme kohta tabelis temaga seonduv unikaalne konto number ja unikaalne perioodi väärtus. Antud massiivi väärtused filtreeris autor läbi, et ei saadetaks tühjasid andmeid ja lisas andmetele vajalikud lisaandmed ning moodustas nendest uue massiivi, mis saadeti üle API serverile.

Tagamaks seda, et tabeli esmasel laadimisel oleks kuvatud kõik rahavood ettevõtte kohalikus valuutas, kasutas autor arvutamiseks andmebaasis olevaid valuutakursse. Leidmaks nende väärtusi lõi autor funktsiooni, millega iga konto kohta käiv rahavoog teatud perioodil arvutati ümber ettevõtte baasvaluutasse ja liideti kokku ning sisestati andmeallikaks olevasse massiivi.

3.3 Tagakomponendi arendus

Tagakomponendi loodud projekt on Unit4Api, mis on loodud ASP.NET Web API 2.0 raamistikul. Antud raamistikku on kirjeldatud lähemalt peatükis 2.2. Joonisel 3 on kujutatud rakenduse tagakomponendi failistruktuur.



Joonis 3. Tagakomponenti failistruktuur näidatuna Visual Studio's

DataInputController tegeleb kõikide HTTP päringutega, mis on seotud andmete sisestamise komponentidega ning *UserController* HTTP päringuga uue kasutaja registreerimiseks.

Tagakomponendi arendamisel on kasutatud andmebaasiga suhtluseks Entity raamistiku. See tagab automatiseeritud mehhanismi andmebaasi andmete päringuteks ja salvestamiseks[21]. Kasutades Entity raamistiku kaasab see ka võimaluse kasutada andmebaasi päringute tegemiseks Linq päringukeelt, mida autor ka kasutas.

Kuna projekti raames uut andmebaasi ei loodud, siis lõi autor tagakomponendi andmemudelid lähenemisega andmebaas esimesena, mille raames luuakse projekti mudelid olemasoleva andmebaasi põhjal[22]. Selle käigus kasutati *ADO.NET Entity Data Model'it*, mille abil loodi andmebaasi ühendus ja andmemudelid automaatselt vastavalt andmebaasile[22].

Tagakomponendi arenduse käigus loodi REST teenused, mille kaudu on võimalik kliendiserveril suhelda tagakomponendiga, mis omakorda võimaldab küsida andmeid

andmebaasist. Tagakomponendi tegevusmeetodite nimetusel on lähtutud Web API võimekusest välja lugeda HTTP meetod otse nimest, näiteks meetodi *GetAllProducts* meetodi puhul oskab Web API välja lugeda HTTP meetodi GET ja sarnaselt näiteks *PostProduct* puhul saab aru, et tegu on POST meetodiga ning samamoodi ka teiste meetoditega nagu uuendamine ja kustutamine[23].

Loodud programmiliideste lõpp-punktid andmete sisestamise vaate jaoks on esitatud tabelis 1 koos päringu HTTP meetoditega ja selle kasutusjuhuga.

Tabel 1. Programmiliideste lõpp-punktid, HTTP meetodid ja kasutusjuhud

Lõpp-punkt	HTTP meetod	Kasutusjuht
/api/datainput/GetCompaniesList	GET	Kõikide ettevõtete andmete pärimine
/api/datainput/GetVersionList	GET	Kõikide andmesisestus versioonide pärimine, mis on aktiivsed ja kuuluvad kategooriasse varad
/api/datainput/GetActivePeriodList	GET	Kõikide aktiivsete perioodide pärimine
/api/datainput/GetAccounts	GET	Kõikide kontode pärimine, mis ei ole peidetud kujul
/api/datainput/GetListOfLayouts	GET	Kontode paigutuste võimaluste pärimine
/api/datainput/GetCurRates?versionId=	GET	Valuutade pärimine vastavalt kasutaja valitud versioonile
/api/datainput/GetLayoutAccs	GET	Kontode ja omavahel seotud kontode paigutuse paaride pärimine
/api/datainput/GetCashpoolForCompany?CompanyId=1	GET	Valitud konto pangakontode pärimine
/api/datainput/GetCashMData?CompanyId=1 &versionId=1&start=20120101&end=20120102	GET	Ettevõtte rahaliste andmete pärimine vastavalt ettevõttele ja versioonile
/api/datainput/GetListOfBanks?CompanyId=1	GET	Kõikide pankade pärimine, millega on valitud ettevõtte seotud.

/api/datainput/PostSaveCashMData	POST	Sisestatud andmete saatmine serverile andmebaasi salvestamiseks. POST kehandiks CashM andmemudelil põhinev JSON.

3.4 Turvalisuse realiseerimine

Antud projekti üheks eesmärgiks oli tagada ka veebirakenduse turvalisus. Selle jaoks kasutas autor JSON Web Token'it, mida on pikemalt kirjeldatud peatükis 2.2. Autentimise ehk sisselogimise käigus tehakse tagakomponendile POST päring, kasutades programmiliidese lõpp-punkti /token. Päringule antakse kaasa kasutaja sisestatud vormi andmed ehk kasutajanimi ja parool ning ja *grant_type,* milleks on "*password*". Tagakomponendis kontrollitakse kasutaja andmete korrektsust ja positiivse kontrolli tulemusena genereeritakse ligipääsu *token* eluajaga 10 tundi ning tagastatakse esikomponendile ligipääsu *token* järgneval kujul:

```
{
    "access_token": <token>
    "token_type": "bearer",
    "expires_in": <aegumis_aeg>
}
```

Antud *token* salvestatakse sessiooni mällu edaspidiseks kasutamiseks ning lubatakse kasutajal jätkata toiminguid rakendusega. Vastasel juhul, kui sisestatud andmed on valed edastatakse kasutajale veateade.

Järgnevate HTTP päringute korral esikomponendi poolt kasutatakse autori poolt loodud *JwtInterceptor* klassi, mis implementeerib *HttpInterceptor'it. HttpInterceptor* kutsutakse välja HTTP päringu korral ning võimaldab käsitleda peatatud päringut[24]. Päringute korral välja kutsutud *JwtInterceptor'i* käsitleb päringut ning lisab päringule autoriseerimise pealkirja järgneval kujul:

```
Authorization: Bearer <token>
```

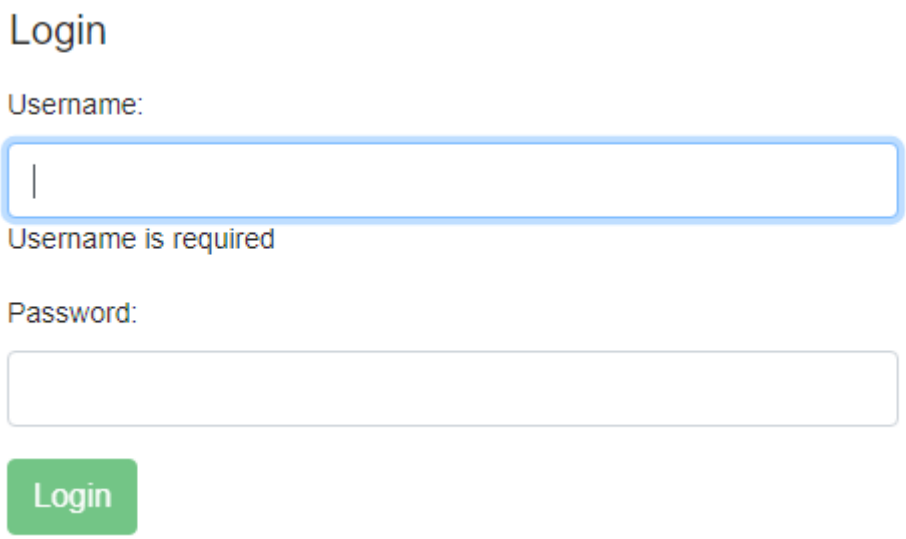
Edasiselt kontrollitakse autoriseerimist tagakomponendi poolt. Selle jaoks on lisatud igale API lõpp-punktile Web API poolt sisseehitatud filtri autoriseerimise atribuut *Authorize.* Antud atribuut kontrollib kas kasutaja on autenditud või mitte. Kui kontroll on edukas, jätkub tegevus, vastasel juhul tagastatakse HTTP staatuse kood 401, mis tähendab, et on tegu autoriseerimata päringuga[25]. Kui esikomponent saab vastuseks staatuse koodi 401, logitakse kasutaja rakendusest välja.

4 Valminud rakenduse kirjeldus ja analüüs

Antud peatükk kirjeldab käesolevas töös loodud veebirakendust jooniste ja vastavate kirjeldustega. Peatüki teises pooles analüüsib autor loodud veebirakendust.

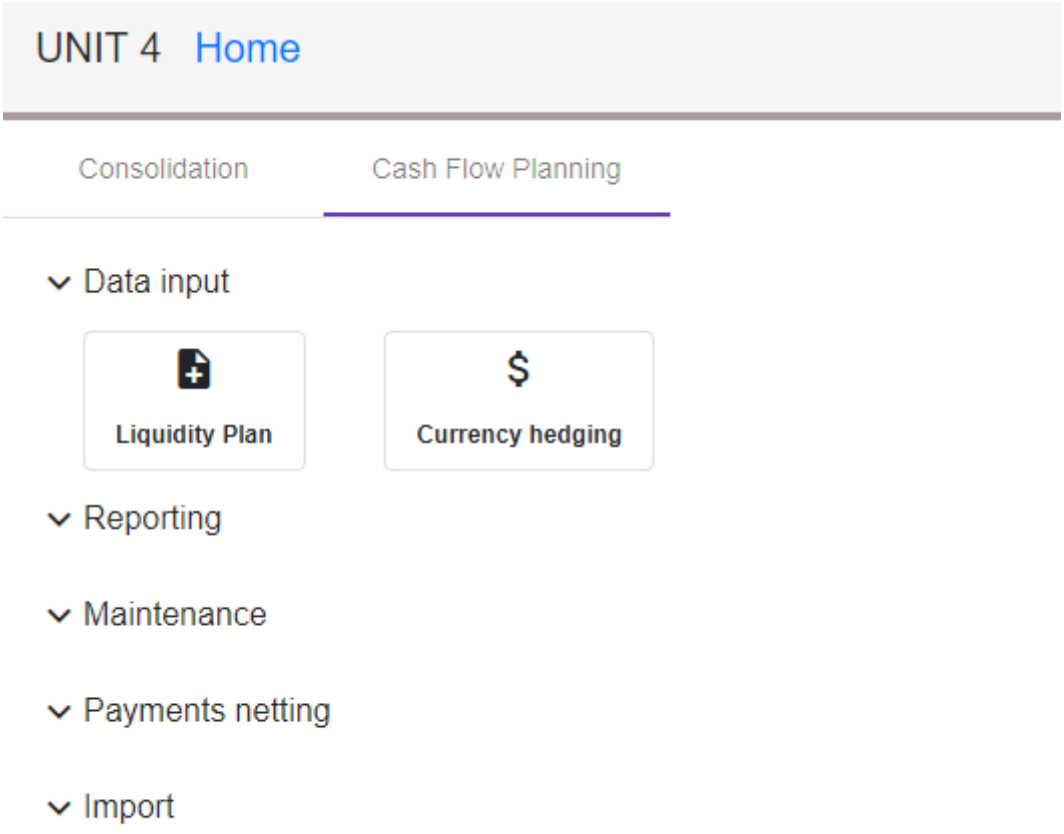
4.1 Sisselogimine

Kogu veebirakenduse kasutuse elutsükkel algab sisselogimise vaatest, võimaldades kasutajal autentida end veebirakendust kasutama. Autentimata kasutajatel puudub teistele vaadetele ligipääs. Joonisel 4 on veebirakenduses kasutusel olev sisselogimise vorm. Aksepteeritavaks sisendiks on kasutajanimi ja parool ning mõlemad väljad on kohustuslikud. Eduka sisselogimise korral antakse kasutajale ligipääs veebirakendusele, vastasel juhul edastatakse veateade "*invalid credentials*".



Joonis 4. Sisselogimise vaade

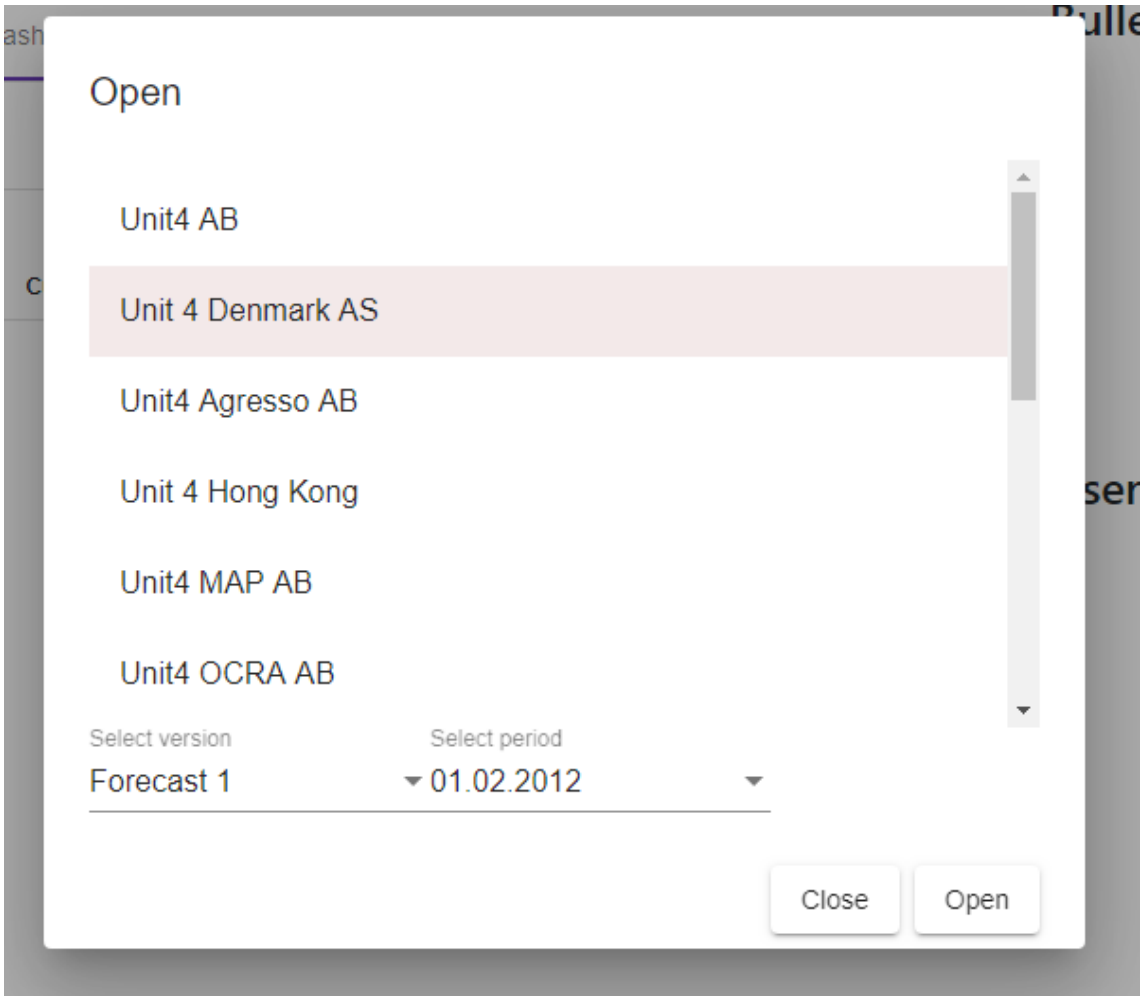
Kui kasutaja sisselogimine on õnnestunud, saadetakse kasutaja automaatselt edasi peamenüü lehele, mis on kujutatud joonisel 5. Antud vaatest edasi on võimalik minna rahavoogude planeerimise paneelile ja sealt andmete sisestuse valikusse, kus on võimalik valida vaid *Liquidity Plan*. Valik *Currency hedging* funktsionaalsust ei paku ning on lisatud ainult visuaalselt sobivaks töölauarakenduse vaatele. Peamenüü lehel olevate paneelide ülejäänud valikud nagu nt *Reporting*, *Maintenance* mingit funktsionaalsest ei paku, kuna antud vaated ei kuulunud projekti skoopi nagu ka eelnevalt mainitud *Currency hedging*.



Joonis 5. Veebirakenduse peamenüü

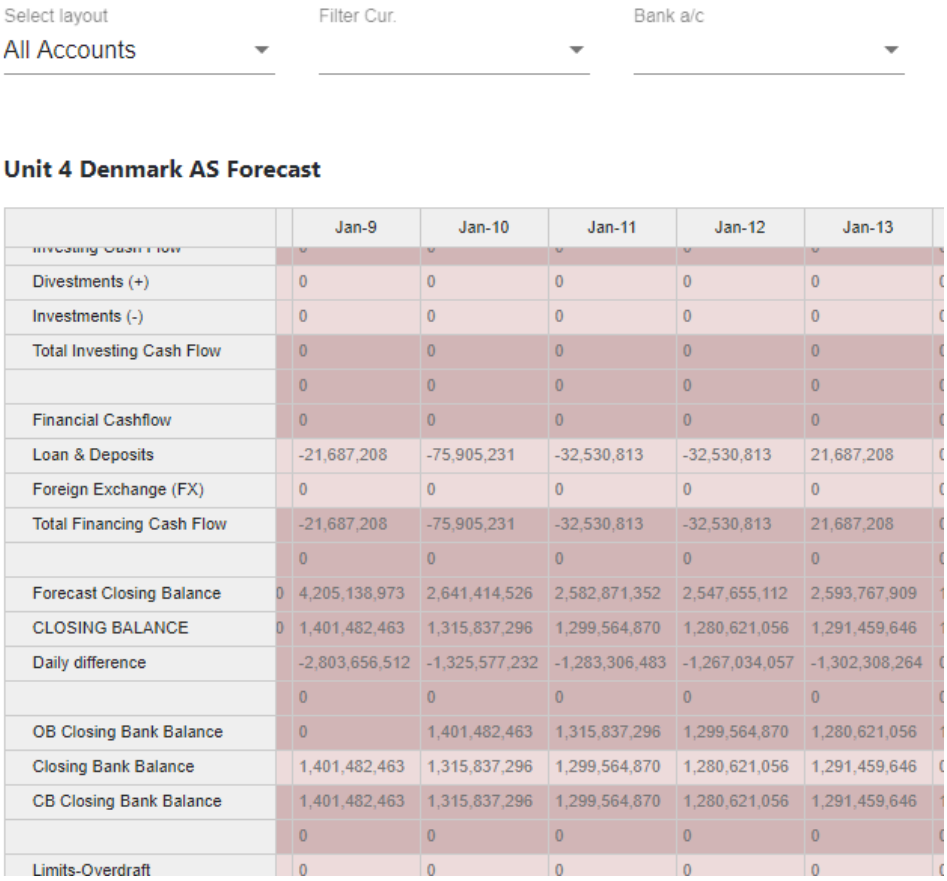
4.2 Andmete sisestamine

Andmete sisestamise vaatele eelnevalt tuleb kasutajal valida ettevõte, versioon ja ajaperioodi algus, mille kohta andmeid sisestada. Selle valiku tegemise jaoks kuvatakse kasutajale MatDialog komponent koos valikuvõimalustega, mis on kujutatud joonisel 6.



Joonis 6. Andmete sisestamis vaate jaoks vajalike valikute hüpikaken

Kui kasutaja on valikud teinud, suunatakse kasutaja edasi andmete sisestamise vaatesse, mis on kujutatud joonisel 7. Antud vaates kuvatakse talle tabel kõikide kontodega ja rahavoogudega. Esmasel laadimisel, kus ei ole valitud pangakonto ega valuuta filtrit, kuvatakse kasutajale informatsioon kõikide kontode kohta ja arvutatakse ettevõtte kõik rahavood valuutasse, mis on määratud ettevõtte vaikimisi valuutaks ja liidetakse kokku. Rahavoogude valuutade konverteerimiseks on kasutatud andmebaasi tabelis *curRate* olevaid valuutakursside suhet valitud versiooni kohta.



	Jan-9	Jan-10	Jan-11	Jan-12	Jan-13
Divestments (+)	0	0	0	0	0
Investments (-)	0	0	0	0	0
Total Investing Cash Flow	0	0	0	0	0
	0	0	0	0	0
Financial Cashflow	0	0	0	0	0
Loan & Deposits	-21,687,208	-75,905,231	-32,530,813	-32,530,813	21,687,208
Foreign Exchange (FX)	0	0	0	0	0
Total Financing Cash Flow	-21,687,208	-75,905,231	-32,530,813	-32,530,813	21,687,208
	0	0	0	0	0
Forecast Closing Balance	4,205,138,973	2,641,414,526	2,582,871,352	2,547,655,112	2,593,767,909
CLOSING BALANCE	1,401,482,463	1,315,837,296	1,299,564,870	1,280,621,056	1,291,459,646
Daily difference	-2,803,656,512	-1,325,577,232	-1,283,306,483	-1,267,034,057	-1,302,308,264
	0	0	0	0	0
OB Closing Bank Balance	0	1,401,482,463	1,315,837,296	1,299,564,870	1,280,621,056
Closing Bank Balance	1,401,482,463	1,315,837,296	1,299,564,870	1,280,621,056	1,291,459,646
CB Closing Bank Balance	1,401,482,463	1,315,837,296	1,299,564,870	1,280,621,056	1,291,459,646
	0	0	0	0	0
Limits-Overdraft	0	0	0	0	0

Joonis 7. Andmete sisestamise vaade laetud andmetega

Antud joonisel kujutatud tabelis on tumehalliga määratud read, mis ei ole mõeldud kasutajapoolseks sisestamiseks vaid tulenevad arvutamisel sisestatud rahavoogudest. Antud joonisel helehallilt esitatud read on samuti määratud vaid lugemiseks, kuna antud päeva kohta on sisestus juba tehtud ja salvestatud. Vastasel juhul, kui valitud ettevõtte kohta ei ole tehtud varasemaid sissekandeid, siis avaneb kasutajale vaade, mis on kujutatud joonisel 8.

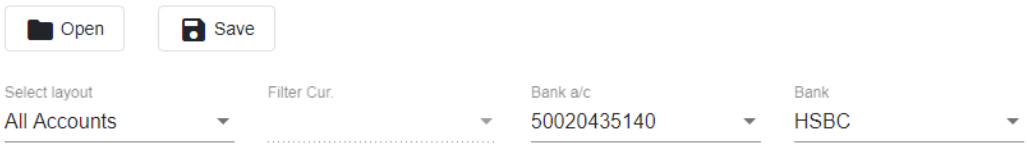
Unit4 Poland Test

	Jan-2	Jan-3	Jan-4	Jan-5	Jan-6	Jan-7	Jan-8	
OPENING BALANCE	0	5,963,567	5,964,131	6,497,741	15,168,714	0	0	0
	0	0	0	0	0	0	0	0
Payments IN	0	0	0	0	0	0	0	0
Cash collections	75,756	0	0	0	0	0	0	0
Accounts receivables imp acc	0	0	0	0	0	0	0	0
Acc receivables (Agresso)	0	0	0	0	0	0	0	0
Acc receivables adj	0	0	0	0	0	0	0	0
Taxes (corporate & social)	87,645	0	5,758	7,896,543	0	0	0	0
Other inflow	0	564	0	8,963	0	0	0	0
IC receipts: UNIT4 Denmark	0	0	0	0	0	0	0	0
IC receipts: UNIT4 MAP	58,745	0	527,852	0	0	0	0	0
IC receipts: UNIT4 Agresso AB	0	0	0	0	0	0	0	0
Total IC Receipts	58,745	0	527,852	0	0	0	0	0
IC Receipts	58,745	0	527,852	0	0	0	0	0
Total IN	222,146	564	533,610	7,905,506	0	0	0	0
	0	0	0	0	0	0	0	0
Payments OUT	0	0	0	0	0	0	0	0
Accounts Payable import acc	0	0	0	0	0	0	0	0
Acc payables (Agresso)	0	0	0	0	0	0	0	0
Acc payables adj	0	0	0	0	0	0	0	0
Rents	0	0	0	0	0	0	0	0

Joonis 8. Andmete sisestamise vaade sisestatud andmetega

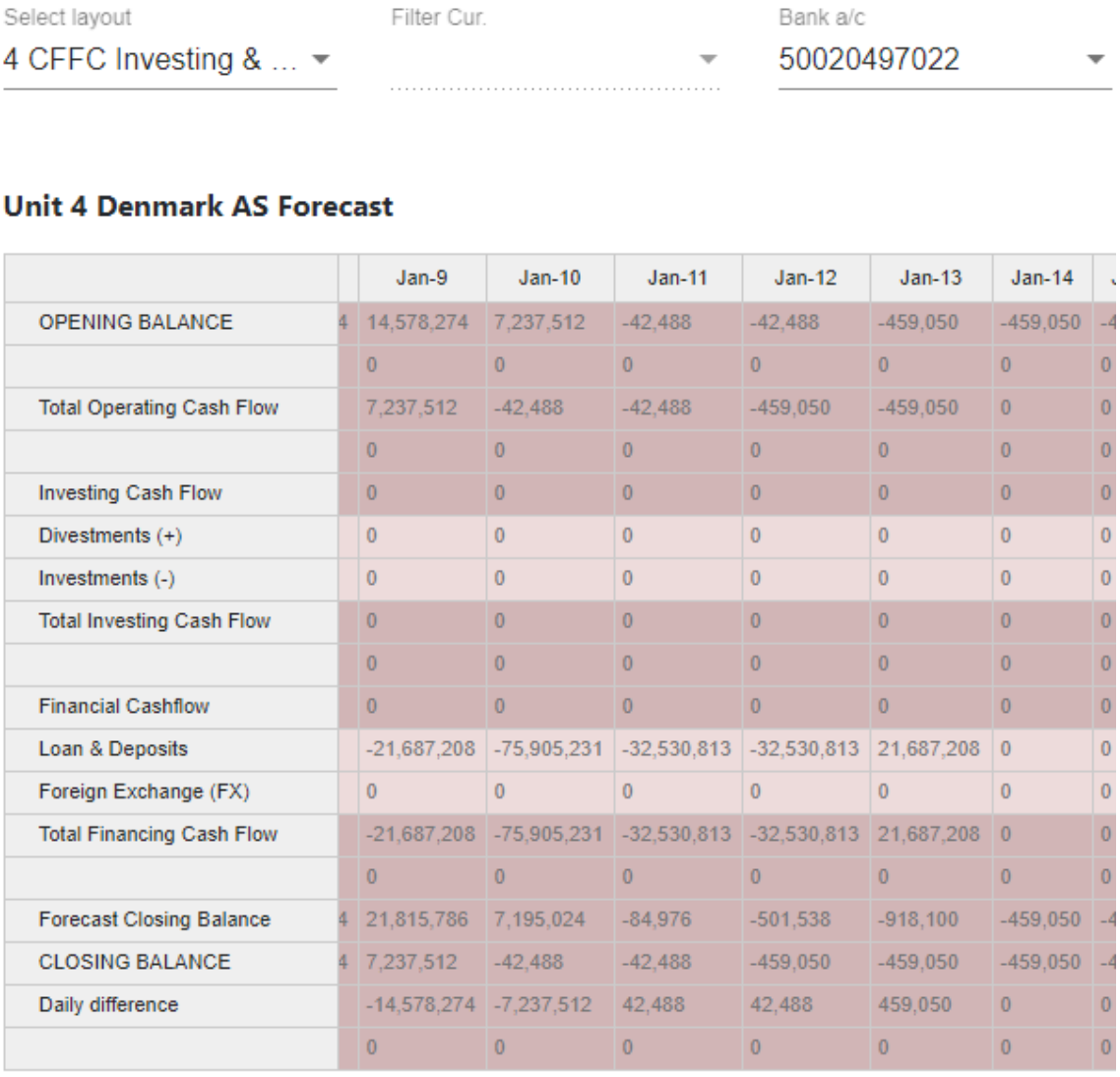
Joonisel 8 on valgelt kuvatud lahtrid mõeldud kasutaja poolseks sisestamiseks ning tumehallid on määratud vaid lugemiseks. Tumehallides kastides on sooritatud arvutused vastavalt kasutaja poolt sisestatud andmetele.

Andmete sisestamise vaates on võimalik filtreerida antud tabelit mitmel erineval viisil kasutades paneeli tabeli kohal mida on kujutatud joonisel 9.



Joonis 9. Andmete sisestamise vaate filtreerimis paneel

Kasutajal on võimalik filtreerida kontosid, mille kohta rahavoogu kuvatakse vähendades sellega tabelis kuvatud ridade arvu. Valides filtriks valuuta, kuvatakse tabelis vaid rahavood, mis on seotud valitud valuutaga. Kui kasutaja valib filtriks pangakonto, siis kuvatakse ainult raha liikumised, mis on seotud valitud pangakontoga, mis omakorda on seotud ühe kindla valuutaga. Sellega kaasnevalt eemaldatakse võimalus kasutajal filtreerida valuuta baasil andmeid. Panga filter võimaldab kasutajal filtreerida pangakontosid, mida kuvatakse pangakonto filtris. Joonisel 10 on kujutatud, kuidas rakendub kontode filtreerimine ning pangakonto baasil andmete filtreerimine. Antud juhul on andmed seotud pangakontoga, mille valuuta on eurodes.



		Jan-9	Jan-10	Jan-11	Jan-12	Jan-13	Jan-14	
OPENING BALANCE	4	14,578,274	7,237,512	-42,488	-42,488	-459,050	-459,050	-4
		0	0	0	0	0	0	0
Total Operating Cash Flow		7,237,512	-42,488	-42,488	-459,050	-459,050	0	0
		0	0	0	0	0	0	0
Investing Cash Flow		0	0	0	0	0	0	0
Divestments (+)		0	0	0	0	0	0	0
Investments (-)		0	0	0	0	0	0	0
Total Investing Cash Flow		0	0	0	0	0	0	0
		0	0	0	0	0	0	0
Financial Cashflow		0	0	0	0	0	0	0
Loan & Deposits		-21,687,208	-75,905,231	-32,530,813	-32,530,813	21,687,208	0	0
Foreign Exchange (FX)		0	0	0	0	0	0	0
Total Financing Cash Flow		-21,687,208	-75,905,231	-32,530,813	-32,530,813	21,687,208	0	0
		0	0	0	0	0	0	0
Forecast Closing Balance	4	21,815,786	7,195,024	-84,976	-501,538	-918,100	-459,050	-4
CLOSING BALANCE	4	7,237,512	-42,488	-42,488	-459,050	-459,050	-459,050	-4
Daily difference		-14,578,274	-7,237,512	42,488	42,488	459,050	0	0
		0	0	0	0	0	0	0

Joonis 10. Andmete sisestus vaate filtreeritud tulemus

Võrreldes jooniseid 7 ja 10, on näha kuidas andmed erinevad samadel kuupäevadel. See tuleneb sellest, et rakendatud on pangakonto filtrit. Kui joonisel 7 on arvutatud kõik

rahavood ettevõtte kohalikku valuutasse, milleks on Taani kroon siis joonisel 10 on kuvatud vaid rahavood, mis on seotud valitud pangakontoga, mille valuutaks on euro. Samuti on rakendatud joonisel 10 kontode sorteerimist. Tulenevalt sellest on peidetud kontod, mis ei ole seotud antud kontode paigutusega.

4.3 Tulemuste analüüs

Käesoleva lõputöö tulemusena sai loodud veebirakenduse põhi ning vaade likviidsus planeerimise andmete sisestuse jaoks, mis loodi töölauarakenduses kasutusel oleva vaate põhjal.

Loodud vaatega on realiseeritud funktsionaalsed nõuded, mis on kirjeldatud peatükis 3.1.1 peale ühe nõude, millest tuleb juttu järmises lõigus. Kirjeldatud nõuete realiseerimeks on kasutatud Handsontable tabeli komponendi erinevaid funktsioone nagu *HiddenRow, RowHeaders, ColHeaders,* F*ormulas* ning *Cells* millest viimase puhul on täpsemalt kasutatud võimalust määrata lahtreid vaid lugemiseks.

Loodud vaate kõige suurem puudus on vaate põhikomponendi ehk tabeli laadimiskiirus. Autor on kindlaks teinud, et antud probleemi põhjustajaks on tabeli loomiseks kasutatud Handsontable komponendi plugin *Formulas.* Antud probleem komponendi kasutamisel sundis autorit taganema funktsionaalsest nõudest tagada võimalus vaadata ja sisestada andmeid päevade, nädalate või kuude kaupa ning piirama *opening balance* konto väärtuse arvutamise loogikat. Selle asemel realiseeris autor vaid päevade kaupa andmete sisestamise ja vaatamise ning *opening balance* konto arvutatakse vaid järgnevale päevale. Kuid kuna antud plugin, *Formulas,* on veel algses arendusfaasis, siis autor on veendunud, et Handsontable tiim lahendab antud probleemi peagi ning siis piisab vaid väljakommenteeritud koodiridade taastamisest, et realiseerida funktsionaalne nõue kuvada andmeid nädalate kaupa ning kasutamaks korrektset arvutamisloogikat *opening balance* konto jaoks.

Sellegipoolest, et tabeli laadimiskiirus ei ole suurepärane ja sorteerimine vaid päevade kaupa, on loodud vaadet võimalik kasutada täielikult. Varasemad andmed laetakse vaates, tagatud on võimalus sisestada andmeid ja nende põhjal on realiseeritud arvutused üle tabeli ning samuti nende salvestamine, võimalus filtreerida tabeli andmeid

nii valuutade kui ka pangakontode baasil ning lisaks on tagatud ka kontode ja pangakontode sorteerimise võimalikkus.

Turvalisuse tagamiseks on autor kasutanud *JSON Web Token'it*. Sellega on tagatud ligipääs veebirakendusele ning programmiliidestele vaid määratud isikutel, kes on süsteemi sisseloginud.

Tagakomponendi poolelt loodi ühendus olemasolevatele andmebaasidele Ocra14 ja CashM. Sellega kaasnevalt loodi mõlema andmebaasi tabelite põhjal andmemudelid kõikide andmebaasi tabelite kohta. Samuti loodi vaate jaoks oluliste andmete päringuteks programmiliidesed, mis on ligipääsetavad vaid autenditud kasutajate poolt.

Kuigi Handsontable plugin *Formulas* valmistas autorile pettumust, on autor rahul enda poolt tehtud valikuga ning veendunud, et arendusmeeskond Handsontable tiimis on peagi tulemas uue versiooni väljastamisega, kus on antud plugini jõudlus märksa parem. Samuti on näha ka Handsontable arendustiimi *backlog'is*t[26], et antud plugini täiustamine on neil juba plaanis.

ASP.NET Web API 2.0 raamistiku valikuga on autor igati rahul. Web API 2.0 koos Entity raamistikuga lõi tõhusa ja kiire programmiliideste loomise keskkonna. Samuti on autor rahul valitud Angular raamistikuga. Antud raamistiku erinevad komponendid olid suureks abiks loodud vaate loomisel. Üheks valiku põhjusteks oli õppida antud raamistikku tundma. Seda autor ka tegi ning on positiivselt meelestatud ka edaspidi töötama antud raamistikuga.

Veebirakenduse edasiarengu vaatepunktist on loodud korralik põhi lisamaks funktsionaalsusi terviklikule rakendusele. Uute funktsionaalsustega kaasnevate andmepäringute programmiliideste loomiseks on loodud kõikide andmebaasi tabelite põhjal andmemudelid muutes uute API'de lisamise lihtsaks. Esikomponendi arendamiseks on koostatud lihtsasti hallatav failistruktuur, lisaks on esikomponendi keerulisemad funktsioonid varustatud kommentaaridega. Samuti on tagatud veebirakenduse turvalisus ning muudaduste korral, näiteks rollide realiseerimisel, on seda lihtne modifitseerida.

5. Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli ettevõtte Unit4 töölauarakenduse Cash Flow Planning: Liquidity Plan Data Input vaate loomine veebirakendusele. Eesmärgi realiseerimiseks kasutas autor esikomponendi arenduseks Angular'i raamistikku, programmiliideste loomiseks ASP.NET Web API 2.0 raamistikku ning andmebaasisüsteemi Microsoft SQL Server. Loodud vaate põhikomponendiks olev tabel on loodud kasutades Handsontable poolt pakutud arvutustabeli raamistikku.

Käesolevas kirjatöös on autor selgitanud täpsemalt rakendust, mille baasil vaade luuakse ning toonud ülevaate erinevatest kasutatud tehnoloogiatest, mida on kasutatud eesmärgi saavutamiseks. Samuti on autor kirjeldanud arendusprotsessi, milles selgitatakse täpsemalt nii projekti analüüsi, esikomponendi kui ka tagakomponendi arendusprotsesse ning kuidas on realiseeritud veebirakenduse turvalisus.

Töö tulemusena valmis veebirakenduse vaade, mis vastab osaliselt seatud funktsionaalsetele nõuetele. Täpsemalt on puudujääk funktsionaalsuses sorteerida andmeid nädalate ja kuude kaupa. Realiseeritud on vaid päevade kaupa sorteerimine. Lisaks kannatas ka *opening balance* konto väärtuse arvutamise loogika. Autor on kindlaks teinud, et antud puudujäägid tulenesid Handsontable komponendi ühest kasutatud pluginist, *Formulas,* nimelt antud pluginit kasutades muutub tabeli komponendi laadimisaeg pikaks ja sellega kaasnevalt ei realiseerinud autor pikemat ajaperioodi kui 31 päeva.

Sellegipoolest on autor rahul kasutatud komponendi valikust, kuna antud plugin, *Formulas,* on Handsontable tiimi poolt veel algses arendusfaasis, ning usub, et peagi lahendatakse jõudluse probleemid antud funktsionaalsusega. Samuti on autor teiste tehnoloogiliste valikutega igati rahul. ASP.NET 2.0 Web API ja Entity raamistiku abil lõi autor mugava keskkonna millega programmiliideseid teha ka edaspidistes arendustes ning esikomponendi arenduses kasutatud Angular'i erinevad moodulid lihtsustasid kliendirakenduse arendust.

Kasutatud kirjandus

[1] The Easiest Way to Run Windows Programs on Mac. [WWW] Available: https://www.makeuseof.com/tag/run-windows-programs-mac/ (20.05.2019)

[2] Cash Flow Planning Brochure. [WWW] Available: https://info.unit4.com/rs/900-SZD-631/images/U4-U4CFP-GEN-BR-Cash-Flow-Planning-Brochure.pdf (20.05.2019)

[3] Angular Architecture Overview. [WWW] Available: https://angular.io/guide/architecture (20.05.2019)

[4] Top Front End Frameworks In 2019. [WWW] Available: https://existek.com/blog/top-front-end-frameworks-2019/ (20.05.2019)

[5] Ng-cli. [WWW] Available: http://ngcli.github.io/ (20.05.2019)

[6] ASP.NET MVC – Web API [WWW] Available : https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_web_api.htm (20.05.2019)

[7] What is ASP.NET. [WWW] Avaliable: https://dotnet.microsoft.com/learn/web/what-is-aspnet (20.05.2019)

[8] Microsoft SQL Server 2016 Relational DBMS overview [WWW] Available: https://searchdatamanagement.techtarget.com/feature/Microsoft-SQL-Server-2016-relational-DBMS-overview (20.05.2019)

[9] Handsontable Introduction [WWW] Available: https://handsontable.com/docs/7.0.2/tutorial-introduction.html (20.05.2019)

[10] Handsontable Features. [WWW] Available: https://handsontable.com/docs/7.0.3/tutorial-features.html (20.05.2019)

[11] Handsontable Release Notes. [WWW] Available: https://handsontable.com/docs/7.0.3/tutorial-release-notes.html (20.05.2019)

[12] Handsontable Licensing. [WWW] Available: https://handsontable.com/docs/7.0.3/tutorial-licensing.html (20.05.2019)

[13] Introduction to JSON Web Tokens [WWW] Available: https://jwt.io/introduction/ (20.05.2019)

[14] Git Documentation [WWW] Available: https://git-scm.com/docs/git (20.05.2019)

[15] GitLab User Documentation [WWW] Available: https://docs.gitlab.com/ee/user/index.html (20.05.2019)

[16] Review: Postman Client Makes RESTful API Exploratino a Breeze [WWW] Available: https://www.programmableweb.com/news/review-postman-client-makes-restful-api-exploration-breeze/brief/2014/01/27

[17] Handsontale Data Sources [WWW] Available: https://handsontable.com/docs/7.0.3/tutorial-data-sources.html (20.05.2019)

[18] Angular Quickstart. [WWW] Available : https://angular.io/guide/quickstart (20.05.2019)

[19] Handsontable Formula Support. [WWW] Available: https://handsontable.com/docs/7.0.2/demo-formula-support.html (20.05.2019)

[20] Handsontable Performance Tips. [WWW] Available: https://handsontable.com/docs/7.0.2/tutorial-performance-tips.html (20.05.2019)

[21] Entity Framework Documentation. [WWW] Available : https://docs.microsoft.com/en-us/ef/#pivot=entityfmwk (20.05.2019)

[22] Entity Framework – Database First Approach [WWW] Available: https://www.tutorialspoint.com/entity_framework/entity_database_first_approach.htm (20.05.2019)

[23] Routing in ASP.NET Web API [WWW] Available: https://docs.microsoft.com/en-us/aspnet/web-api/overview/web-api-routing-and-actions/routing-in-aspnet-web-api (20.05.2019)

[24] Angular HttpInterceptor [WWW] Available: https://angular.io/api/common/http/HttpInterceptor (20.05.2019)

[25] Authentication and Authorization in ASP.NET Web API [WWW] Available: https://docs.microsoft.com/en-us/aspnet/web-api/overview/security/authentication-and-authorization-in-aspnet-web-api (20.05.2019)

[26] Handsontable backlog [WWW] Available: https://trello.com/c/6HMfUvR4/62-improvement-formula-engine (20.05.2019)