TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ellen Loit 164742IABB

KASUTAJALIIDEST AUTOMAATSELT GENEREERIVA KOMPONENDI LOOMINE DATAHUB CONNECTORI NÄITEL

Bakalaureuse töö

Juhendaja: Viljam Puusep
Magistrikraad

Tallinn 2020

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: Ellen Loit

22.05.2020

Annotatsioon

Bakalaureusetöö eesmärgiks oli seadistada arenduskeskkond, vajalikud ühendused ja luua esialgne connectori kasutajaliides ettevõttele Synerall AS. Antud rakendus on mõeldud kommunaalteenuseid, ehk vett, gaasi, elektri- ja soojusenergiat pakkuvate ettevõtete infosüsteemide haldamiseks.

Datahub on elektrienergia jaemüügituru tsentraliseeritud andmevahetussüsteem, mis sisaldab 3,7 miljoni Soome elektrienergia mõõtepunkti andmeid. Synerall AS aga on ettevõte, mis pakub CIS ja arvelduslahendus tarkvara kommunaalteenuseid pakkuvatele ettevõtetele ning antud ettevõtte eesmärgiks on rakendada Datahubi protsessi külgloogikat. Datahubi ja Syneralli vahelise andmevahetuse jaoks luuakse connector, mis pakub Synerallile liidestusvõimalust, mille abil saab veebiteenuste kaudu edastada sõnumeid.

Arendusvahendina kasutatakse Microsoft Visual Studio 2019 ning projektis kasutatakse C# programmeerimiskeelt. Veebirakenduse arenduseks kasutatakse Angular'i esikomponendi raamistikku ning ASP.NET Core 3.0 tagakomponendi loomiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 25 leheküljel, 4 peatükki, 5 joonist, 0 tabelit.

Abstract

Development of the component designed to automatically generate user interfaces based on the example of Datahub Connector

The goals of this bachelor's thesis are to configure the development environment and the necessary connections, as well as to create an initial connector user interface for Synerall AS. This application is intended for the management of information systems of utility companies providing water, gas, electricity and heat.

Datahub is a centralized data exchange system for the retail electricity market, which contains data from 3.7 million Finnish electricity metering points. Synerall AS, on the other hand, is a company that offers customer information system and billing solution for utility companies. The purpose of Synerall AS is to implement the side logic of the Datahub process. For the exchange of data between Datahub and Synerall, a connector is developed that provides Synerall with an interface that allows messages to be transferred via web services.

Microsoft Visual Studio 2019 is used as the development tool and the back-end of the project is written in the C# programming language. The author used Angular framework for front-end development and ASP.NET Core 3.0 for creating the back-end.

The thesis is in Estonian and contains 25 pages of text, 4 chapters, 5 figures, 0 tables.

Lühendite ja mõistete sõnastik

API	Application Programming Interface, rakendusliides
Back-end	Eestikeelne vaste ingliskeelsele IT-s kasutatavale teriminile "backend"
CIS	Customer Information System tähendab kliendi infosüsteemi
Datahub	Andmekeskus
DOM	Document Object Model on rakendusliides HTML'i ja XML'i kirjutamiseks, mis defineerib dokumendi struktuuri
Front-end	Andmete teisendamine graafiliseks liideseks kasutades selleks HTML, CSS ja JavaScripti, et kasutajad saaksid andmetega suhelda ja neid vaadata
Git	Versioonihaldustarkvara
MBaaS	Mobile backend as a service
MDM-lahendus	Mobiilseadmete keskhaldusteenus
Npm	Node Package Manager on JavaScripti programmeerimiskeele jaoks loodud paketihaldur
TFS	Team Foundation Server on tarkvaraarendusriistade komplekt
TypeScript	Microsofti arendatav tüübikirjeldust võimaldav JavaScriptil baseeruv ja selleks kompileeriv programmeerimiskeel
WCF-teenus	Windows Communication Foundation on teenus, mida kasutatakse teenustele orienteeritud rakenduste arendamiseks

Sisukord

Jooniste loetelu

1 Sissejuhatus

Käesoleva lõputöö raames ehitatakse connectori veebirakenduse põhi Eesti ettevõtte Net Group OÜ tütarettevõtte Synerall AS jaoks. Rakenduse eesmärgiks on hallata kommunaalteenuseid, ehk vett, gaasi, elektri- ja soojusenergiat, pakkuvate ettevõtete infosüsteeme. Synerall AS on ettevõte, mis pakub klientidele CIS ja arvelduslahendus tarkvara.

Synerall AS kliendi andmete haldamine toimub läbi Eesti andmekeskuse. Kuna ettevõte on suunatud Soome turule, siis tuleb ehitada data connector, mis ühenduks Soome andmekeskusega. Soome ja Eesti andmehalduses on erinevusi, mis on tingitud erinevustest seaduse määrustikudes riikides [1].

Bakalaureusetöö teoreetilises osas võtab autor vaatluse alla kolm populaarsemat front-end raamistikku ning toob välja peamised eelised ja puudused. Samuti kirjeldatakse rakenduse arendamisel kasutatavat ASP.NET Core raamistikku ja analüüsitakse Web API ja MVC tehnoloogiaid veebiteenuste loomiseks.

Käesoleva lõputöö praktilises osas tutvustab autor loodud veebirakendust, selle funktsionaalsuseid ning kasutatud tehnoloogiaid.

1.1 Eesmärk

Antud lõputöö eesmärgiks on arenduskeskkonna seadistamine, vajalikude ühenduste loomine ja esialgse kasutajaliidese arendamine ettevõttele Synerall AS. Selle jaoks tuleb luua tagarakendus ASP.NET Core'i baasil ja kasutajaliides.

Front-end'i poolel tuleb võrrelda JavaScript'i raamistikke ning leida sobiv antud veebirakenduse jaoks, lähtudes sellest, milline raamistik oleks kõige optimaalsem. Lõputöö tulemusena peab valmima rakendus koos back-end'i ning front-end'iga, võimalusel ka andmebaasiga.

Kasutajaliidese põhiülesanneteks on andmekeskusest tulevate sõnumite haldamine, mis tähendab sõnumite filtreerimist, sorteerimist ja muutmist. Seal hulgas on oluline kuvada

kasutajale kõik sisse- ja väljaminevad sõnumid ning vajadusel ka sõnumite detailandmed. Lahendus peab olema dünaamiline, mis tähendab automaatset kasutajaliidese loomist vastavalt etteantud sõnumitüübile.

1.2 Probleem ja taust

Datahub on elektrienergia jaemüügituru tsentraliseeritud andmevahetussüsteem, mis sisaldab 3,7 miljoni Soome elektrienergia mõõtepunkti andmeid. Andmekeskuses sisalduvat teavet kasutab umbes 100 elektritarnijat ja 80 elektritarbijaid teenindavat jaotusvõrguettevõtet. Aastal 2022, kui Soome tarbija soovib vahetada elektrienergia tarnijat, saadetakse kogu vajalik teave elektrimüüja ja jaotusvõrguettevõtte vahel andmekeskuse ehk tsentraliseeritud teabevahetussüsteemi kaudu. Andmed asuvad hetkel erinevate ettevõtete süsteemides [2].

Datahubi eesmärgiks on koondada kogu oluline teave elektritarbimise kohta ühte kohta, et kiirendada, lihtsustada ja täiustada andmekeskuse kõigi osapoolte tegevust. Lisaks sellele pakub tsentraliseeritud lahendus kõikidele osapooltele võrdset ja samaaegset juurdepääsu andmetele. Tarnijate vahetamine, uuele aadressile kolimine ja muud tarbija elektrilepingut mõjutavad muudatused on vaid põhjus andmevahetuseks. Näiteks uues süsteemis arveldatakse jaotusvõrguga seotud tasakaalustamatust. Datahub saab ka sellese salvestatud andmeid töödelda ja täpsustada. Nutikad, kaugloetavad elektrimõõdikud on Soomes laialdaselt kasutusel ja iga mõõtepunkti kohta genereeritakse päevas suur hulk andmeid. Need andmed, nagu ka kõik tulevikus kasutusele võetavad mobiilirakendused, võimaldavad elektritarbijatele täiesti uusi teenuseid. Näiteks, rakendus suudab tulevikus jälgida kasutaja nii linna kodu kui ka suvekodu elektritarbimise näitajaid. Olenemata sellest kas need mõõtepunktid asuvad kõrvuti või riigi erinevates osades [2].

Syneralli eesmärgiks on rakendada Datahub'i protsessi külgloogikat. See hõlmab näiteks lepingute loomist, pikendamist, eemaldamist ja protsesside ülesütlemist. Mõõteandmed edastatakse Datahub'i kliendi MDM-lahenduse abil ja Synerall loeb kõik Datahub'i mõõtmisandmed (ka oma võrgupiirkonna tarbimispunktide jaoks). Luuakse eraldi Syneralli arenduskeskkond, mis on integreeritud CGI Datahub'i arenduskeskkonda, kus kõik andmed on sünteetilised.

Syneralli Datahub'i projekt on jaotatud seitsmeks etapiks, järgides peamiselt Fingridi sidusrühmade testimise etappe. Iga etapp lõpeb rakendatud funktsioonide testimisega.

Etapid on järgmised:

1. Ettevalmistusetapp – Arenduskeskkonna rakendamine, vajalikke ühenduste loomine ja nõutavate andmemudelite muudatuste tegemine sisemistesse liidestesse.
2. 1. etapp – Tarbimispunktide rakendamine ja lepingute ning ühenduste loomine.
3. 2. etapp – Kliendi- ja tarbimispunktide värskenduste ja katkestuste rakendamine.
4. 3. etapp – Lepingute värskendamise, lõpetamise ja ülesütlemise rakendamine, toodete ja arvete haldamine.
5. 4. etapp – Mõõtmisandmete, automatiseerimise ja turu andmehalduse rakendamine.
6. Küpsemise etapp – Funktsionaalsuse ning protsesside testimine. Vigade parandus.
7. Ettevalmistus Go-Live'i jaoks – Lõpliku toote testide käivitamine, proovid ja lõplik Go-Live.

Antud lõputöös pööratakse tähelepanu Synerall Datahub projekti esimesele etapile, mis hõlmab endas peamiselt arenduskeskkonna loomist. See tähendab esimeste vaadete ja veebiteenuste arendamist.

1.3 Metoodika

Lähtudes töö eesmärgist luua toimiv veebirakendus, millel on front-end, back-end ja andmebaas, valitud töö metoodikaks osutus arendusuuring. See tähendab, et alguses tuleb eelseisva töö vajaduste analüüs ja kavand. Järgmisena teostatakse arendus ja rakendamine ning töö lõpus hinnatakse valminud rakendust [3].

Arendusuuringud on laialt levinud tehnoloogia ja tootmise valdkondades ning nende eesmärgiks on analüüsida ja uurida projekteerimisprotsessi. Algselt oli arendusuuringu metoodika fookus suunatud toodete ja teenuste kavandemise protsessile, kuid tänapäeval

on see laienenud nii kaugele, et hõlmab endas ka toote ja tootmise hindamist. Põhiline küsimus millele tuleb vastata antud uuringu metoodikat kasutades on: kuidas luua tõhusat toodet, protsessi või mudelit [3]?

Ettevõtte poolt saadud sisendina oli teada, et veebirakenduse tagarakend peab olema realiseeritud C# keeles, kasutades ASP.NET Core'i. Toetudes kirjandusele ja raamistikude võrdlusele tuleb valida üks raamistik ning realiseerida esitluskiht ehk front-end.

Juhul kui arendustööd lähevad plaanijärgselt ning ajakava järgides tuleb luua andmebaas. Andmebaasi tehnoloogia valitakse pärast veebirakenduse esialgset versiooni loomist.

Antud töös kasutatakse TFS versioonihaldust, mis pakub arendajatele võimalust hoida kogu arendatav kood Git keskkonnas. TFS versioonihaldus on tsentraliseeritud versioonikontrollisüsteem, mis on arendatud Microsofti poolt. Antud lahendus on osa Azure DevOps Services ja Team Foundation Server'ist [4].

Versioonihaldust kasutatakse eelkõige selleks, et vältida koodi kustumist või juhul kui tekivad koodis vead on võimalik iga hetk tehtud muudatused tagasi võtta. Versioonihalduse valik olenes eelkõige sellest, mis on ettevõttes kasutusel ning millega on tiimil kõige rohkem kogemusi.

2 Teooria

Antud peatükis kirjeldatakse töövahendeid ja raamistike, mida kasutatakse rakenduse loomiseks. Samuti analüüsitakse enim kasutatavamaid ja populaarsemaid raamistikke, selgitatakse mille alusel tehti valik ning kirjeldatakse autori poolt valitud töövahendeid.

2.1 Front-end raamistikud

JavaScript on aastal 1995 Brendan Eich'i loodud programmeerimiskeel, mille eesmärgiks oli veebilehtede skriptimine, mis lubas esitluskihi genereerimist etteantud koodist. Programmeerimiskeele loomisel oli suureks eeskujuks C keel, mida iseloomustavad looksulud, avaldised ning „if", „while" ja „for" laused [5].

Node.js on käituskeskkond, mis võimaldab serveripoolseid rakendusi kirjutada JavaScriptis. Seega on võimalik luua täiuslik JavaScript rakendus, kus nii esikomponent kui ka tagarakendus on kirjutatud samas programmeerimiskeeles. Node.js avaldab märkimismäärset mõju JavaScripti ökosüsteemile, tutvustades arendajatele npm paketihaldurit ja populariseerides CommonJs mooduleid. Arendajad hakkasid looma uuenduslikumaid tööriistu ja arendama uusi lähenemisviise, et hägustada piirid brauseri, serveri ja rakenduse vahel [6].

Analüüsitavad esitluskihi raamistikud valiti mitme kriteeriumi põhjal. Oluline on, et raamistik toetaks C# programmeerimiskeelt, oleks jätkusuutlik ning pidevalt uuendatav loojate poolt. Kindlasti väga olulist rolli mängis raamistiku dokumentatsiooni olemasolu, kvaliteet ning kättesaadavus. Kuna autoril on võrdlemisi vähe kogemusi front-end raamistikega, siis õppematerjalide ja -ressursside olemasolu oli tõhusa töö tegemiseks kohustuslik. Siinkohal võib mainida ka seda, et populaarsust hinnati võrreldes veebis olemasolevaid statistikaid, mis näitavad antud raamistiku kasutajate arvu.

Ülaltoodud kriteeriumite alusel valiti edaspidiseks analüüsiks välja Vue.js, AngularJS ja React esitluskihi raamistikud. Samuti räägitakse järgmistes peatükkides ka TypeScriptist, mis on JavaScripti rakenduse keel. TypeScript lisab JavasScripti valikulised tüübid juurde, mis toetavad suuremahuliste JavaScripti rakenduste tööriistu mis tahes veebibrauseri, masina või operatsioonisüsteemi jaoks [7].

2.1.1 TypeScript ja JavaScript võrdlus

JavaScripti arendati algusest peale kliendipoolseks raamistikuks, kuid arendajad avastasid, et see töötab ka serveripoolse raamistikuna. JavaScripti arenemise tulemusena kasvas see liiga mahukaks ja keeruliseks, mistõttu ei suutnud see enam objekt-orienteeritud programmeerimiskeele nõudeid täita. Mis tõttu pidid Microsofti arendajad täiustama olemasolevat JavaScripti nii palju, et see täidaks nõudeid, kuid samas ei tahetud kaotada tavalist JavaScripti. Tänu sellele arendati TypeScript, mis vastab objekt-orienteeritud programmeerimiskeele nõuetele [8].

TypeScript on loodud Microsofti poolt ja ei ole loetav veebibrauserite poolt, seega pärast koodi kompileerimist tõlgendatakse ja konverteeritakse TypeScript JavaScriptiks. Samuti võimaldab TypeScript kasutada tüübikirjeldust ning vastab kõikidele ECMAScript standarditele. Kokkuvõttes võib öelda, et TypeScripti moodustavad JavaScripti skriptide teegid ja funktsionaalsused [9].

TypeScripti peamisteks eelisteks on:

1. Klasside ja moodulite tugi. Märksõnad nagu klass, liides, laiend ja moodul on TypeScriptis saadaval [10].

2. Staatiline trükkimine aitab valideerida kirjutatud koodi enne rakenduse kompileerimist. Selline lahendus toob kõik trüki- ja tüübivead välja koheselt. Varasemalt kasutades JavaScripti antud tüüpi vigade leidmiseks tuli esitada päring back-end'i [10].

3. ES6 komponendi ja ES madalamate versioonide tugi [10].

4. Lihtsustatud veebiteenused ja komponendid. TypeScript võimaldab korduvalt kasutada samu komponente läbi viitamise koodis. Selleks tuleb luua .d.ts-fail, kus deklareeritakse antud teegi tüübid ja veebiteenused [10].

TypeScripti puudusteks võib nimetada aeglasemat kompileerimist ja abstraktsete klasside mitte toetamist [10]. Aeglane kompileerimine toob endaga kaasa arendaja töö efektiivsuse ja kiiruse langust. Selline veebirakenduse käitumine on tingitud sellest, et enne kompileerimist teisendatakse TypeScript JavaScriptiks veebibrauseris.

2.1.2 Vue.js

Vue.js on progressiivne raamistik kasutajaliidese ehitamiseks. Võrreldes teiste monoliitsete raamistikega, näiteks AngularJS, on Vue.js loodud algusest peale järk-järguliselt vastuvõetavaks. Raamistik keskendub ainult vaadete kihile ning seda on kerge integreerida olemasolevasse projekti või kasutada koos teise raamistikuga. Tänapäeval koguvad populaarsust üheleherakendused ehk Single-Page Applications, mis tähendab, et korraga laaditakse alla tervik veebirakendus (JavaScripti ja HTML'i failid). Vue.js on sellist tüüpi rakenduste loomiseks võimeline ja jätkusuutlik tuleviku mõttes [11].

Vue.js rakendab virtuaalset DOM funktsionaalsust, see tähendab, et ilma tervet lehte taas laadimata on võimalik muuta valitud lehe osa. Vue.js on eelkõige mõeldud väiksemate prototüüpide või kasutajaliidesele fokuseeritud rakenduste loomiseks, kuna sellel puudub suur dokumentatsioon ning valikus on piiratud arv võimalusi. Ühest küljest võib eeldada, et Vue.js kasutamine on lihtsam ning loodud rakendus kiirem, sest ei ole integreeritud suuri komponente või seoseid. Teisest küljest võib öelda, et see raskendab mahukate projektide loomist, sest arendaja on piiratud võimalustes [12].

2.1.3 AngularJS

AngularJS on struktureeritud raamistik dünaamiliste veebirakenduste loomiseks. Antud raamistik võimaldab kasutada HTML'i mallina, kuhu saab lisada meelepärast funktsionaalsust. AngularJS'i andmeside tehnoloogia (*data binding*) ja sõltuvuste süstimine kõrvaldab suurema osa tühjast koodist, mida muul juhul kirjutatakse nullist. Antud raamistik on ehe näide sellest milline oleks HTML, kui see oleks disainitud rakendustele, mitte veebilehtedele [13].

AngularJS'i eeliseks React'i ees on eelkõige kahesuunaline andmete sidumine. AndularJS on ehitatud Model-View-Controller arhitektuuri järgi ning raamistik sünkroniseerib mudeli ja vaate automaatselt ka kompileerumise käigus. JavaScripti väärtuste muutmisel silutavas rakenduses reageerib vaade koheselt antud muudatusele. See võimaldab arendajatel oluliselt vähendada kulutatud aega arendamisele tänu sellele, et ei pea pärast igat muudatust rakendust uuesti siluma panema. Olulist rolli mängib ka sõltuvuste süstimise võimalus. Sellega määratakse erinevate koodi lõikude suhtlust üksteisega ning kuidas ühe komponendi muutused mõjutavad teisi. Sõltuvuste süstimine

muudab komponendid taaskasutatavateks, kergemini hallatavateks ning testitavateks [14].

AngularJS'i puuduseks võib välja tuua jõudluse langemise. Dünaamilised rakendused nõuavad palju ruumi, mistõttu kulub rohkem aega laadimiseks. AngularJS'i mahuka dokumentatsiooni tõttu muutub antud raamistiku õppimine tunduvalt keerukamaks, mis tõstab ka ajakulu töö tegemisel. Samuti on AngulariJS'il mis tahes ülesande täitmiseks alati rohkem kui üks viis, mis võib olla nii positiivne kui ka negatiivne külg antud raamistiku kasutades [14].

2.1.4 React

React on on Facebooki'i poolt arendatud JavaScripti teek kasutajaliidese loomiseks. Antud raamistik on loodud eesmärgiga lihtsustada kasutajaliidese loomist tänu taaskasutatavate komponentide kasutamisele. React esindab MVC arhitektuurist V osa ehk vaadet, mis on omakorda jaotatud komponentideks [15].

Algaja arendaja vaatepildist on React'il oluline eelis teiste raamistike ees tänu korrektsele dokumentatsioonile, suurele hulgale juhendeid ning ressurssidele harjutamiseks. Samuti on React'i veebirakendus loodud mitmest komponendis ja igal komponendil on oma loogika ja kontrollerid. Komponendid vastutavad väikese korduvkasutatava HTML-koodi tüki väljastamise eest ning neid saab kasutada rakenduses kõikjal. Selline arendusviis muudab töö tegemise kiiremaks ja tõhusamaks. React on kiiresti uuenev JavaScript'i raamistik ning seetõttu peavad arendajad omalt poolt koguaeg juurde õppima ja liikuma uuendustega kaasa, mis võib tekitada üleliigseid probleeme arenduses. Oluline puudus võrreldes teiste raamistikega seisneb selles, et React hõlmab ainult front-endi osa ehk kasutajaliidese kihte rakenduses, mistõttu täieliku tööriistakomplekti saamiseks tuleb võtta kasutusele veel mõned tehnoloogiad [16].

2.2 Back-end

Lähtudes ettevõtte soovidest ning tiimi kogemustest otsustati, et rakendus tuleb ASP.NET Core baasil C# programmeerimiskeeles.

2.2.1 ASP.NET Core

ASP.NET Core 3.0 on uus platvormiülene ja avatud lähtekoodiga raamistik modernsete pilvepõhiste rakenduste loomiseks. Taoliste rakenduste hulka kuuluvad veebirakendused, IoT rakendused ja MBaaS põhised rakendused [17].

ASP.NET Core 3.0 rakendused käivitatakse nii .NET Core raamistikul kui ka .NET raamistikul. ASP.NET Core raamistik on loodud pakkuma optimeeritud arendusraamistikku rakendustele, mida saab kasutada kas pilves või lokaalselt. See koosneb moodulkomponentidest, millel on minimaalsed kulutused, nii et säiliks maksimaalne paindlikkus tarkvaralahenduste kavandamisel ning rakendamisel. ASP.NET Core 3.0 rakendusi saab arendada ja käivitada Windows'il, Linux'il ja macOS'il [17].

Klassikalisel .NET raamistikul põhineva rakenduse loomisel tuleb valida kuue rakendusmudeli vahel, mis võib tekitada segadust ja alandada produktiivsust. Koos uue ASP.NET Core 3.0'ga on rakendusmudelite arv langenud kaheni ning kood on nüüd jagatav, mistõttu arendajad saavad taaskasutada 90% koodist [17].

Kliendipoolse arenduse osas on ASP.NET Core 3.0 disainitud sujuvalt integreerima mitmesuguste kliendipoolsete raamistikega, sealhulgas AngularJS, KnockoutJS ja Bootstrap. [17]

2.2.1.1 MVC

Suurem osa tänapäeva veebirakendusi on teostatud lähtudes „Model-View-Controller" mustrist, mis tähendab teisisõnu MVC'd.[18] MVC on arhitektuuri muster, mis jagab rakenduse kolmeks kihiks – mudel, vaade ja kontroller. Kasutaja tegevus MVC mustrit järgivas rakenduses täidab ühte tsüklit: kasutaja teostab toimingu, mille järel rakendus muudab andmemudelit ja edastab kasutajale uuendatud vaate. Tsükkel kordub. Antud käitumisviis on iseloomulik HTTP päringute kaudu andmeid edastatavatele veebirakendustele. MVC puhul on tavaline, et kogu andmevahetus ja kasutajaliidese kokku panemine toimub serveris [19].

2.2.1.2 Web API

ASP.NET Web API on HTTP teenuste loomise raamistik, millele pääseb ligi igast kliendist, sealhulgas brauseritest ja mobiilseadmetest [20]. API ehk rakendusliides on

alarutiini mõistete, protokollide ja tööriistade komplekt tarkvara ja rakenduste loomiseks. Teisisõnu API on liides, millel on funktsionaalsuste kogum, mis võimaldavad arendajatel ligi pääseda rakenduse, opsüsteemi või muude teenuste konkreetsetele funktsioonidele või andmetele [21].

ASP.NET Web API töötab enam-vähem sama loogika järgi nagu ASP.NET MVC veebirakendus, välja arvatud, et andmed saadetakse vastusena, mitte HTML-i vaatena. See on justkui veebiteenus või WCF-teenus, kuid erandiks on see, et toetab ainult HTTP protokolli [21].

2.3 Lõplik tehnoloogia valik

Lähtudes eelnevalt toodud raamistike võrdlusest ja analüüsist otsustas autor TypeScripti kasuks. Antud otsus oli tingitud eelkõige taaskasutatavate komponentide kasutusvõimalusest ning sarnasusest back-end programmeerimiskeelte loogikaga, kuna autoril on üsna väike kogemus esitluskihtide arendamises. TypeScripti kasutamine vähendas raamistike valikut. Vue.js tundus autorile väheste võimalustega raamistikuna, tuginedes eelmises peatükis välja toodud raamistiku kirjeldusele. Arvestades sellega, et projekt on tuleviku mõttes üsna mahukas ja mõeldud suure andmemahu haldamiseks, tekkisid autoril kahtlused Vue.js raamistiku jätkusuutlikuse osas, mis tõttu jäi Vue.js võrdlusest välja.

Valides React'i ja Angular vahel otsustas autor Angulari kasuks, sest React on pidevalt uueneva dokumentatsiooniga ja nooremarendaja jaoks võib see oluliselt raskendada tööd. Lähtudes samuti enda ja tiimi kogemusest oli Angulari eelis, sest sellega oli autor juba varasemalt kokku puutunud.

Seoses sellega, et tekkis huvi TypeScripti vastu ja AngularJS ei toeta seda tuli teha ootamatu otsus AngularJS'i uuema versiooni kasuks. Mis tõttu tuli autoril veebirakenduse arendamisel kasutada uuemat versiooni Angularist ehk Angular 8.

3 Rakenduse arendamine

Käesolev peatükk sisaldab arendustegevuse kirjeldust, mis sisaldab dokumentatsiooni, kasutajaliidese ning tagakomponendi arendamist.

3.1 Struktuur

Veebirakenduse arendamine algas kasutajaliidese loomisest kasutades Angular raamistikku. Lähtudes sellest, et autori ametikoht on .net arendaja oli tiimi poolt tehtud otsus, et rakenduse loomisel kasutatakse juba olemasolevat Angular'i malli. Malli kasutusõigused on ostetud ning selleks malliks osutus „Metronic theme pack", kuna selle kujundus sobis ideega kokku ja kasutajate tagasiside oli enamjaolt positiivne. Antud malli on ostetud 2020 aasta aprilli seisuga 80 000 korda ja keskmine hinnang on 4,9. Selleks, et alustada projekti loomisega tuli installeerida Node.js, npm ja Angular CLI [22].

Antud rakenduse versioonihaldus teostatakse TFS'is, mis on ettevõtte sisene ning antud informatsiooni tohib jagada vaid osadena.

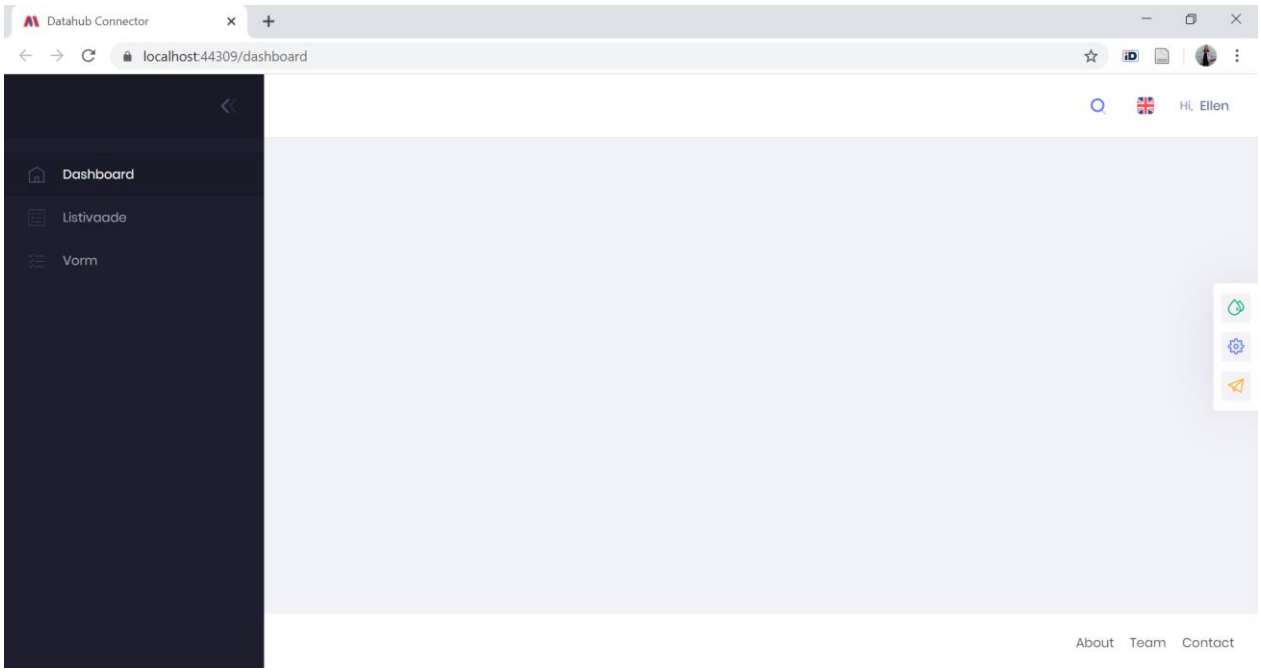
Projekti ja arenduskeskkonna seadistamiseks kasutas autor eelnevalt mainitud malli ja tühjendas selle üleliigsetest komponentidest ja vaadetest [22]. Kohalike sõltuvuste paigaldamiseks kasutas autor npm käsku:

```
npm install
```

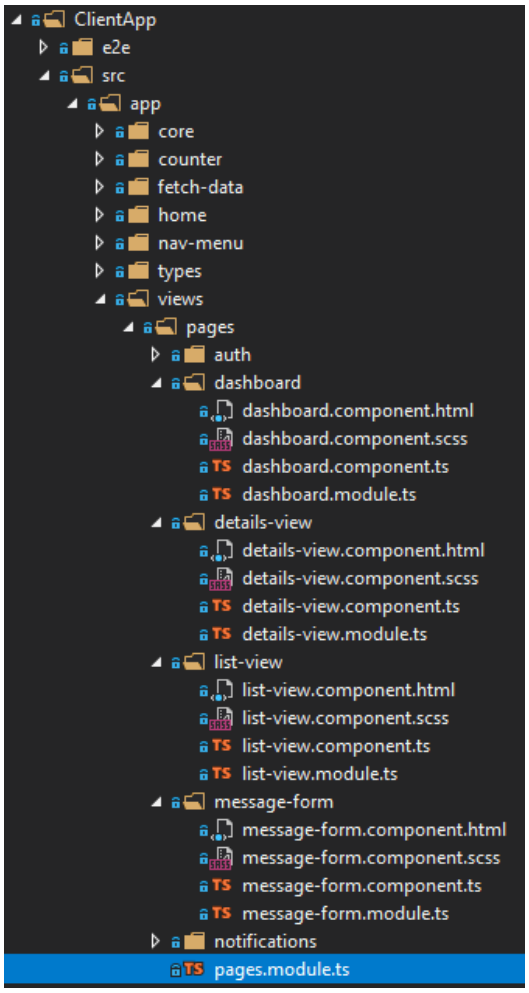
Rakenduse vaadete loomiseks kasutas autor käsku:

```
ng generate component views/pages/my-page
```

Joonisel 2 on kujutatud olemasolevate vaadete komponentide failistruktuur. Veebirakenduses on hetkel loodud avaleht, listi-, detail- ning vormivaade. Avaleht on hetkel tühi, sest sinna genereeritakse hiljem olemasolevate andmete põhjal trendid.



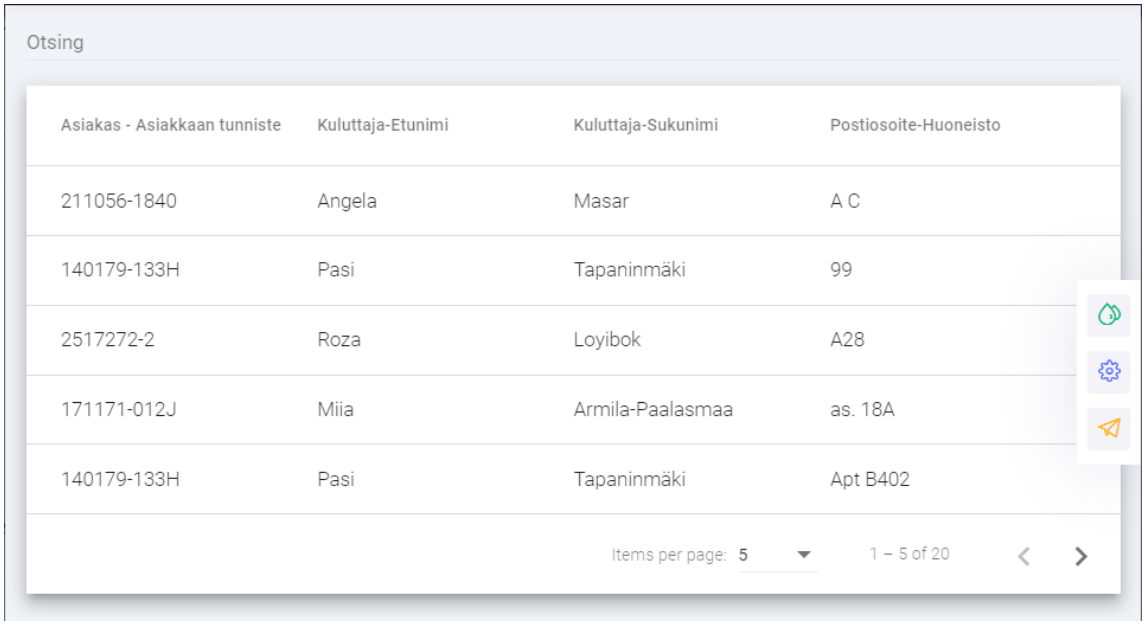
Joonis 1 Avaleht



Joonis 2 Esikomponendi failistruktuur Visual Studio Code koodiredigeerijas

3.1.1 Listivaade

Vaate eesmärgiks on sõnumite kuvamine ja haldamine. Vaade võimaldab kuvada kõik sissetulevad sõnumid ühes tabelis ning haldada neid. Listivaatele on lisatud sorteerimise, filtreerimise, andmete lisamise funktsionaalsused. Selleks on kasutatud Angulari Mat komponenti.



Joonis 3 Listivaade

3.1.2 Detailvaade

Listivaates on võimalus valida spetsiifiline sõnum ning valitud sõnumi detailandmetega saab kasutaja tutvuda eraldi vaates. Selleks on loodud detailvaade, kuhu saab ligi vajutades listivaates reale sobiva sõnumikoodiga. Detailvaates on kuvatud kõik andmed, mis on seotud valitud sõnumiga.

Joonis 4 Detailvaade

3.1.3 Vormivaade

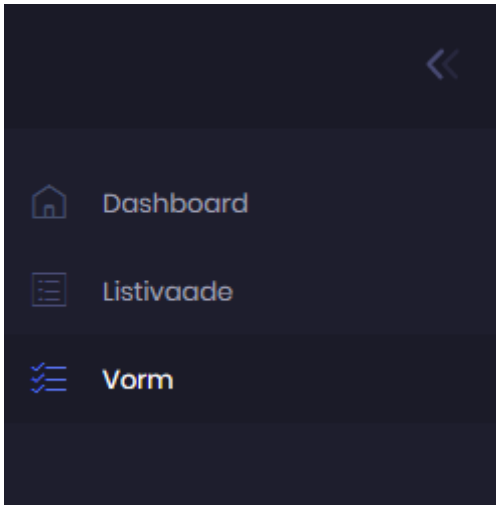
Vormivaate loomisel on kasutatud komponente võimaldamaks dünaamilist vormi loomist olenevalt sisendist. Antud vormi genereerimisel kasutatakse komponenti, mis genereerib json schema'st vormi. Komponendi nimi on formly ning selle installeerimiseks kasutasin järgmist käsku:[23]

```
ng add @ngx-formly/schematics --ui-theme=bootstrap
```

Pärast komponendi installeerimist genereerisin uue vaate kasutades Angular CLI käsku:

```
ng generate component views/pages/message-form
```

Selleks, et pääseda antud vaatele ligi tuleb valida menüüst „vorm":



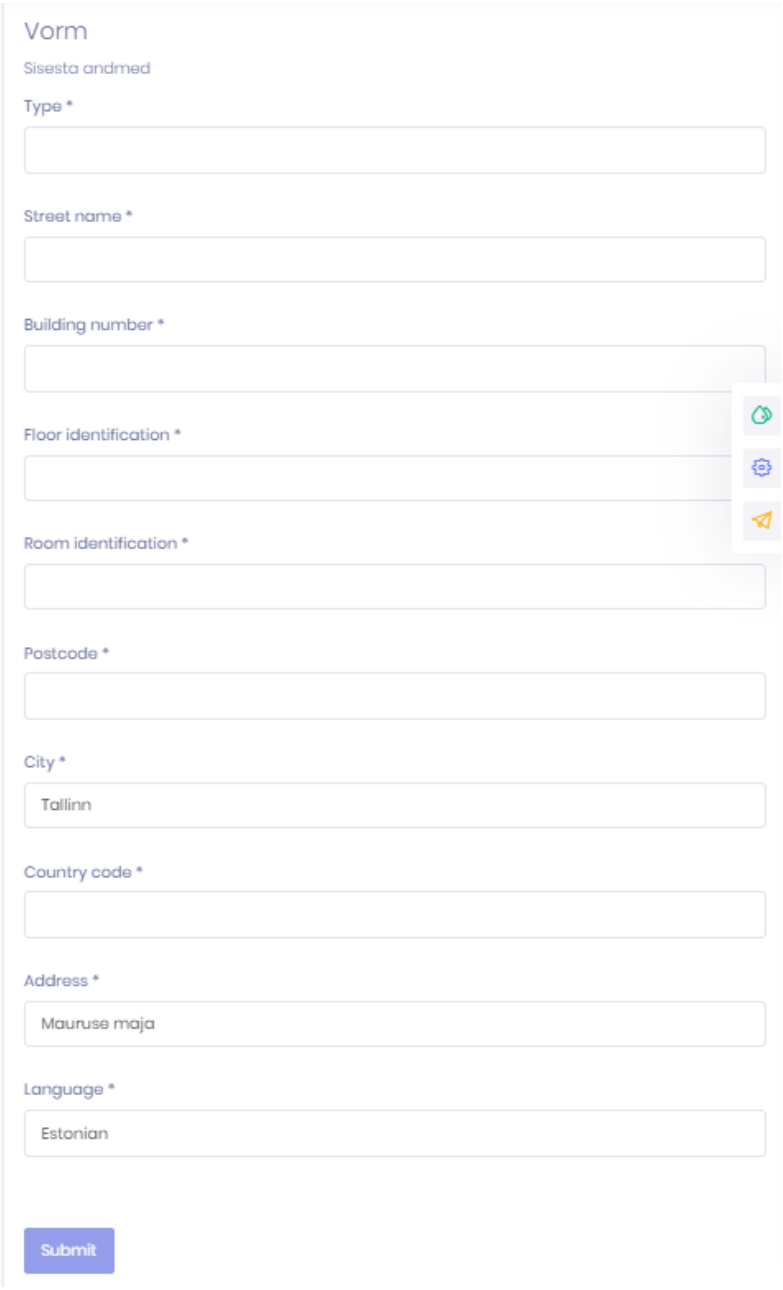
Selle järel avaneb aken koos dünaamiliselt genereeritud vormiga. Lähitulevikus saadetakse vormi genereerimiseks Syneralli andmebaasi päring, mille kaudu saadakse soap'ist xsd sõnum. Tänapäeval need andmed puuduvad Synerall'i andmebaasis seoses sellega, et teisel osapoolel esinevad raskused andmete saatmisega andmebaasi.

Seetõttu kasutas autor teise osapoole xsd näidisfaili, millest manuaalselt genereeriti C# klass. Antud C# klassis on kirjeldatud protsessid ja atribuudid.

Tänapäeval antud loogika vajab veel täiendamist, sest C#'ist json-schema faili genereerimine ei toimi. Autor genereeris manuaalselt C# klassist json-schema faili kasutades veebis olemasolevat komponenti. Seejärel lisas autor valmis genereeritud faili

rakenduse esitluskihi failide teeki. Vormi komponendis viitas autor antud json-schema failile, selle asemel, et genereerida igakord uus fail.

Antud C# klassi mudelist genereeritakse .json fail, mille sees on json schema, mida kasutatakse vormivaate genereerimiseks. Vormi genereerimiseks kasutas autor komponenti nimega Formly ja rakendas olemasolevat loogikat Formly dokumentatsioonist:[23]



Joonis 5 Json-schema’ga failist automaatselt genereeritud vormi vaade

4 Kokkuvõte

Käesoleva lõputöö eesmärgiks oli luua connectori veebirakenduse põhi, mis haldab kommunaalteenuseid, ehk vett, gaasi, elektri- ja soojusenergiat, pakkuvate ettevõtete infosüsteemi.

Autori poolt valitud metoodika eeldas populaarsete JavaScript'i raamistike analüüsi, esitluskihi ning rakenduskihi loomist ja arenduskeskkonna seadistamist. ASP.NET Core'i baasil veebirakenduse arendamiseks kasutas autor C# programmeerimiskeelt ning front-endi arendamiseks Angular'i raamistikku.

Töö autor ei saavutanud täies mahus püstitatud eesmärki lõputöö käigus. Bakalaureusetöö raames tehtud front-end raamistike analüüsi käigus selgus, et sobiv variant antud veebirakenduse loomiseks oli Angular kasutades komponentide arendamiseks TypeScripti. Rakenduse poolelt sai valmis esialgne versioon kasutajaliidesest ning mõned veebiteenused. Lõputöö alguses oli planeeritud andmebaasi integratsioon antud rakendusse, kuid ajapuuduse tõttu ei jõudnud autor seda realiseerida.

Siinkohal toon välja ka selle, et projekt valmis tööandja käe all ja oli oluline kooskõlastada arendusprotsess tiimiga. Kuna antud veebirakenduse arendamine on äärmiselt suur ja mahukas projekt, kuhu on kaasatud mitu tiimi nii Eestis, kui ka Soomes, siis tekkis probleeme ajakulu arvestamisega. Samuti võttis ka dokumentatsioonide lugemine ja arendamine eelnevalt planeeritust rohkem aega, sest autoril puudus kogemus nii Angular'i kui ka TypeScriptiga.

Kasutatud kirjandus

[1] ‘Synerall - Customer Information System & Billing Solution for Utilities’. https://synerall.com/ (accessed Mar. 01, 2020).

[2] ‘The Datahub project - towards centralised information exchange | EDIELfi’. https://ediel.fi/en/datahub (accessed May 10, 2020).

[3] H. L. T. Heikkinen, ‘Pedagoogiliste arendusuuringute suunad’, *Eesti Haridusteaduste Ajakiri. Estonian Journal of Education*, vol. 7, no. 2, pp. 6–22, Nov. 2019, doi: 10.12697/eha.2019.7.2.02a.

[4] ‘Integration With Microsoft Team Foundation Version Control | TestComplete Documentation’. https://support.smartbear.com/testcomplete/docs/working-with/integration/scc/tfvc/index.html (accessed May 22, 2020).

[5] R. Toal *et al.*, *Programming Language Explorations*. Chapman and Hall/CRC, 2016.

[6] S. B. Online, ‘Chapter 1: The Anatomy of a Modern JavaScript Application - Modern JavaScript’. https://learning.oreilly.com/library/view/modern-javascript/9781492023548/Text/modernjsant1_split_000.html (accessed May 04, 2020).

[7] *microsoft/TypeScript*. Microsoft, 2020.

[8] ‘Difference between TypeScript and JavaScript’, *GeeksforGeeks*, Jun. 14, 2018. https://www.geeksforgeeks.org/difference-between-typescript-and-javascript/ (accessed Mar. 24, 2020).

[9] ‘TypeScript - Overview - Tutorialspoint’. https://www.tutorialspoint.com/typescript/typescript_overview.htm (accessed May 03, 2020).

[10] R. Feng, ‘The Advantages of Typescript Over JavaScript | AppDynamics’, *Application Performance Monitoring Blog | AppDynamics*, Oct. 21, 2015. https://www.appdynamics.com/blog/engineering/the-benefits-of-migrating-from-javascript-to-typescript/ (accessed May 03, 2020).

[11] ‘Introduction — Vue.js’. https://vuejs.org/v2/guide/ (accessed May 03, 2020).

[12] ‘Comparison with Other Frameworks — Vue.js’. https://vuejs.org/v2/guide/comparison.html (accessed May 20, 2020).

[13] ‘AngularJS: Developer Guide: Introduction’. https://docs.angularjs.org/guide/introduction (accessed May 03, 2020).

[14] ‘The Good and the Bad of Angular Development’, *AltexSoft*. https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/ (accessed May 20, 2020).

[15] ‘ReactJS Tutorial - Tutorialspoint’. https://www.tutorialspoint.com/reactjs/ (accessed May 04, 2020).

[16] ‘Pros and Cons of ReactJS - javatpoint’, *www.javatpoint.com*. https://www.javatpoint.com/pros-and-cons-of-react (accessed May 20, 2020).

[17] S. B. Online, ‘What is ASP.NET Core 2.0? - Learning ASP.NET Core 2.0’. https://learning.oreilly.com/library/view/learning-aspnet-core/9781788476638/cover.xhtml (accessed May 04, 2020).

[18] S. B. Online, ‘Creating MVC Applications - Learning ASP.NET Core 2.0’. https://learning.oreilly.com/library/view/learning-aspnet-core/9781788476638/cover.xhtml (accessed May 04, 2020).

[19] S. B. Online, ‘1. ASP .NET Core MVC in Context - Pro ASP.NET Core MVC 2’. https://learning.oreilly.com/library/view/pro-aspnet-

core/9781484231500/A338050_7_En_1_ChapterPart1.html (accessed May 04, 2020).
[20] ‘ASP.NET Web API Tutorials’. https://www.tutorialsteacher.com/webapi/web-api-tutorials (accessed May 04, 2020).
[21] ‘What is Web API?’ https://www.tutorialsteacher.com/webapi/what-is-web-api (accessed May 04, 2020).
[22] ‘Metronic - Bootstrap 4 HTML, React, Angular 8 & VueJS Admin Dashboard Theme’, *ThemeForest*. https://themeforest.net/item/metronic-responsive-admin-dashboard-template/4021469 (accessed May 11, 2020).
[23] ‘Angular Formly’. https://formly.dev/guide/getting-started (accessed May 11, 2020).

Lisa 1 – Datahub sõnumitüüpide näited

Event	Message type	Message type description	Sender	Recipient
DH-111-2	F01	Structural data, customer	Datahub	DSO
DH-111-3	F01	Structural data, customer	Datahub	Supplier
DH-111-4	F01	Structural data, customer	Datahub	Third party
DH-112-1	F01	Structural data, customer	DSO	Datahub
DH-112-2	F01	Structural data, customer	Datahub	Supplier
DH-112-3	F01	Structural data, customer	Datahub	DSO
DH-112-4	F01	Structural data, customer	Datahub	DSO
DH-112-5	F01	Structural data, customer	Datahub	Third party
DH-113-1	F01	Structural data, customer	Third party	Datahub
DH-113-2	F01	Structural data, customer	Datahub	Supplier
DH-113-3	F01	Structural data, customer	Datahub	DSO
DH-121	E58	Structural data, accounting point	DSO	Datahub
DH-122-1	E58	Structural data, accounting point	DSO	Datahub
DH-122-2	E58	Structural data, accounting point	Datahub	Supplier
DH-122-3	E58	Structural data, accounting point	Datahub	Third party
DH-123	E58	Structural data, accounting point	DSO	Datahub
DH-124-1	E58	Structural data, accounting point	Supplier Third party	Datahub
DH-124-2	E58	Structural data, accounting point	Datahub	DSO
DH-131-1	F02	Accounting point identification request	Supplier Third party	Datahub
DH-131-2	F20	Accounting point list	Datahub	Supplier Third party
DH-132-1	F03	Accounting point data request	Current supplier	Datahub
DH-132-2	F21	Accounting point data	Datahub	Current supplier
DH-133-1	F03	Accounting point data request	Potential supplier	Datahub
DH-133-2	F21	Accounting point data	Datahub	Potential supplier

Lisa 2 – Näide xsd’st genereeritud C# klassist

```
public partial class MeteringPointAddress_E58_Type
    {

        private Type_S4_NFI_2_Type typeField;

        private string streetNameField;

        private string buildingNumberField;

        private string floorIdentificationField;

        private string roomIdentificationField;

        private string postcodeField;

        private string cityNameField;

        private CountryCode_S2_5_2_Type countryCodeField;

        private string addressFreeFormField;

        private Language_S2_5_2_Type languageField;

        /// <remarks/>
        public Type_S4_NFI_2_Type Type
        {
            get
            {
                return this.typeField;
            }
            set
            {
                this.typeField = value;
            }
        }

        /// <remarks/>
        public string StreetName
        {
            get
            {
                return this.streetNameField;
            }
            set
            {
                this.streetNameField = value;
            }
        }

        /// <remarks/>
        public string BuildingNumber
        {
            get
            {
                return this.buildingNumberField;
            }
            set
            {
                this.buildingNumberField = value;
            }
        }

        /// <remarks/>
        public string FloorIdentification
        {
            get
            {
                return this.floorIdentificationField;
            }
            set
            {
                this.floorIdentificationField = value;
            }
        }

        /// <remarks/>
        public string RoomIdentification
        {
            get
```

```
            {
                return this.roomIdentificationField;
            }
            set
            {
                this.roomIdentificationField = value;
            }
        }

        /// <remarks/>
        public string Postcode
        {
            get
            {
                return this.postcodeField;
            }
            set
            {
                this.postcodeField = value;
            }
        }

        /// <remarks/>
        public string CityName
        {
            get
            {
                return this.cityNameField;
            }
            set
            {
                this.cityNameField = value;
            }
        }

        /// <remarks/>
        public CountryCode_S2_5_2_Type CountryCode
        {
            get
            {
                return this.countryCodeField;
            }
            set
            {
                this.countryCodeField = value;
            }
        }

        /// <remarks/>
        public string AddressFreeForm
        {
            get
            {
                return this.addressFreeFormField;
            }
            set
            {
                this.addressFreeFormField = value;
            }
        }

        /// <remarks/>
        public Language_S2_5_2_Type Language
        {
            get
            {
                return this.languageField;
            }
            set
            {
                this.languageField = value;
            }
        }
    }
{
```

Lisa 3 – Näide C# klassist genereeritud json-schema failist


```
    "schema": {
        "type": "object",
        "title": "A registration form",
        "description": "A simple form example.",
        "required": [
            "type",
            "streetName",
            "buildingNumber",
            "floorIdentification",
            "roomIdentification",
            "postcode",
            "cityName",
            "countryCode",
            "addressFreeForm",
            "language"
        ],
        "properties": {
            "type": {
                "type": "string",
                "title": "Type",
                "default": null
            },
            "streetName": {
                "type": "string",
                "title": "Street name",
                "default": null
            },
            "buildingNumber": {
                "type": "integer",
                "title": "Building number",
                "default": null
            },
            "floorIdentification": {
                "type": "string",
                "title": "Floor identification",
                "default": null
            },
            "roomIdentification": {
                "type": "string",
                "title": "Room identification",
                "default": null
            },
            "postcode": {
                "type": "integer",
                "title": "Postcode",
                "default": null
            },
            "cityName": {
                "type": "string",
                "title": "City"
            },
            "countryCode": {
                "type": "integer",
                "title": "Country code",
                "default": null
            },
            "addressFreeForm": {
                "type": "string",
                "title": "Address"
            },
            "language": {
                "type": "string",
                "title": "Language"
            }
        }
    },
    "model": {
        "language": "Estonian",
        "addressFreeForm": "Mauruse maja",
        "cityName": "Tallinn"
    }
}
```