TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ain Kivimägi 206678IAIB

PÄIKESEPARKIDE PROJEKTIDE TEENINDUSPLATVORMI TAGARAKENDUS

Bakalaureusetöö

Juhendaja: Gert Kanter
PhD

Tallinn 2023

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: Ain Kivimägi

21.05.2023

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua kliendile Smartecon, kes on Eesti üks suurimaid päikeseelektrijaamade paigaldajaid, veebiplatvormi Elos minimaalse töötava toote tagarakendus ning see koos esirakendusega avalikustada.

Lõppeesmärk selle lõputöö väliselt on kujundada Elosest võimekas rahvusvaheline päikesepaneelide iseteeninduslatvorm, mis abistaks ja suunaks eri riikidest pärit erakliente läbi tellimuse esitamise, pakkumiste ja lepingute, maksmiste ja paigaldamise faaside kiiresti ja arusaadavalt.

Antud lõputöö tulemusena on valmis selle rakenduse minimaalne töötav versioon, mis koordineerib tellimuste esitamist ja kooskõlastamist ning genereerib automaatselt pakkumisi ja haldab lepinguid, tehes ära vajaliku eeltöö, et selle alusel edasi minna arveldamiste ja paigaldusprotsessiga.

Töö esimeses osas käsitletakse praegust olukorda, analüüsitakse konkurentide lahendusi ja seletatakse projekti visiooni ja arendusmetoodikat. Seejärel esitletakse rakenduse funktsionaalsust ning lahatakse eesmärkide saavutamiseks valitud tehnoloogiaid ja valikute põhjuseid ning tagarakenduse tehnilist lahendust. Järgmisena antakse ülevaade nii tagarakenduse kui ka paralleelselt arendatava esirakenduse jaoks pilvekeskkondade rajamisest ning rakenduse avalikustamise protsessist. Viimasena analüüsitakse tehtud töö tulemusi ning defineeritakse järgmised sammud jätkuarendusteks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 52 leheküljel, 6 peatükki, 16 joonist, 0 tabelit.

Abstract

Solar Park Service-platform Backend Application

The purpose of this thesis is to develop the minimum viable product of backend application of a portal called Elos as a request from Smartecon, who is one of the largest solar panel park providers and installers in Estonia and to also publish it together with parallelly developed frontend application.

The ultimate goal outside the scope of this thesis is to develop Elos to be a powerful and international self-service platform, which would assist and direct clients from different countries through the phases of order, offer, contract, payment and installation.

The functionality implemented in the scope of this thesis includes creating and coordinating orders, generating offers and handling contracts, which creates a solid project foundation for continuing with billing and installment phases.

The first part of the thesis consists of the analysis of already implemented solutions of some competititors and explains the project vision and development methodology. After that, the author presents the implemented functionality of the application and dissects the chosen technologies and reasons behind them, as well as the technical solution of the backend application. Then the overview of publishing the backend and frontend applications is given. Finally, the results of the completed work are analyzed and next steps for future developments are defined.

The thesis is written in Estonian and is 52 pages long, including 6 chapters, 16 figures, 0 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendusliides
Autentimine	Protsess kontrollimaks teise olemi identideedi tõesust
Autoriseerimine	Protsess, mis jagab või keelab olemitele ligipääsu veebiressurssidele
AWS	*Amazon Web Services*, üks enam-levinuid pilveteenuste pakkujaid
CI	*Continuous Integration*, pidevintegratsioon, muudatuste pidev ühildamine
CSRF	*Cross-Site Request Forgery*, ründemetoodika sessiooni äärandamiseks
CSS	*Cascading Style Sheets*, märgistuskeel veebilehtede kujundamiseks
Deemon	Taustal töötav protsess, mis sooritab ettemääratud toiminguid
DNS	*Domain Name System*, domeeninimede süsteem, mis teisendab domeeninimed kasutavateks IP-aadressiteks
Dockeri konteiner	Tarkvarapakett, mis sisaldab kõike rakenduse käivitamiseks
Dockeri kujutis	Juhised, mida kasutatakse Dockeri konteineri loomiseks
Esirakendus	*Frontend*, kasutajale nähtav osa rakendusest
HTML	*HyperText Markup Language*, märgistuskeel veebilehtede sisu defineerimiseks
HTTP	*Hypertext Transfer Protocol*, protokoll teabe edastamiseks arvutivõrkudes
HTTPS	*Hypertext Transfer Protocol Secure*, turvalisem protokoll teabe edastamiseks arvutivõrkudes
IDE	*Integrated Development Environment*, integreeritud programmeerimiskeskkond
Integratsioonitest	Test, millega kontrollitakse komponentide omavahelist liidestust
IP-aadress	*Internet Protocol address*, internetiaadress
JRC	*Joint Research Centre*, Teadusuuringute Ühiskeskus
Konveier	Sammude kaupa käivitav kood rakenduse testimiseks, ehitamiseks ja tarnimiseks

Küpsis	*Cookie*, väike tekstifail, mis salvestatakse brauserisse selleks, et identifitseerida kasutajat või koguda infot tema eelistuste kohta
Lõpp-punkt	Kindla aadressiga liides, mille pihta tehakse päringuid rakendusega suhtlemiseks
Mikroteenused	Arhitektuuristiil, mis jagab rakenduse ühendatud kindla ülesandega teenuste kogumiks
MVP	*Minimum Viable Product*, minimaalne töötav toode
PDF	*Portable Document Format*, failiformaat
Pilv	Hõlmab servereid ja tarkvara, mis on internetist ligipääsetav
Plugin	Pistikprogramm süsteemile täiendava funktsionaalsuse lisamiseks
Raamistik	Platvorm abistamaks arendajat koodi kirjutamisel
REST	*Representational state transfer*, veebirakendustes kasutatav tarkvara arhitektuurilaad
Retro	Kord sprindis teostatav koosolek, mille käigus vaadatakse tagasi möödunud sprindile
SQL	*Structured Query Language*. Standardiseeritud andmebaasi päringukeel
Stand-up	Regulaarselt korraldatav lühike koosolek
Tagarakendus	*Backend*, serveri poolne tarkvara, mis haldab andmeid ja nende töötlust
Teek	*Library*, korduvkasutamiseks mõeldud kollektsioon funktsioonidest ja rutiinidest
TLS	*Transport Layer Security*, transpordikihi turve
Token	Pikk sõne, mida kasutatakse näiteks kasutaja tuvastamisel
URL	*Uniform Resource Locator*, internetiaadress
VPC	*Virtual Private Cloud*, virtuaalne privaatpilv
Üksustest	Test, millega kontrollitakse väikse alamprogrammi individuaalset töötamist
XSS	*Cross-site-scripting*, ründemetoodika, mille käigus suunatakse rakendust käivitama pahatahtlikku koodi

Sisukord

Jooniste loetelu

1. Sissejuhatus

1.1 Taust

Smartecon[1] on Eestis üks suurimaid päikeseelektrijaamade paigaldajaid ning siiani on nende fookus olnud eelkõige äriklientidel ja suurte päikeseparkide rajamisel kasutades peamiselt projektijuhtimise tarkvara Scoro[2].

Maailmas järjest populaarsemaks muutuva rohepöörde tõttu on aga eraklientide huvi järjest kasvamas ning seetõttu on nende visioon vallutada eraklientide turg esialgu naaberriikides ning hiljem mujal Euroopas. [1]

1.2 Hetkeolukord

Praegusel hetkel on põhiprobleem see, et Smarteconi töötajad peavad väga palju manuaalset tööd tegema näiteks projekti ja pakkumiste koostamisel, kliendiga kooskõlastamisel, dokumentide ja arvete valmistamisel ja paigalduspartnerite koordineerimisel, mis on hallatav vähema arvu väga suurte projektide puhul, kuid muutub ebapraktiliseks paljude väikeste projektide puhul ning seab piirangud sellele, kui palju projekte suudetakse korraga vastu võtta ja teostada.

Smartecon tunnetab Scoro puhul mitmeid probleeme. Esiteks, kuna tegemist on üldise projektijuhtimise tarkvaraga, on rakendus ülemäära mahukas ning sellest tulenevalt ka keerulisem kasutada. See ei paku tuge automaatsete projekti- ja hinnakalkulatsioonide tegemiseks ning etteantud andmete põhjal pakkumise automaatseks koostamiseks, vaid seda tuleb teha uuesti käsitsi sisestades. Probleemne on ka andmete automaatne ülekanne erinevate vaadete ja vormide vahel ning alltöövõtjatele ei ole võimalik faile valikuliselt näidata. Projekti koordineerimine projektijuhi ja kliendi poolte pealt käib enamjaolt vaba suhtluse teel, mis nõuab palju vaeva ja aega ning põhjustab tegelikult välditavaid viiteid infovahetuse puhul, mis omakorda pärsib klientide uudishimu ja innukust.

Rakenduse jaoks ideede kogumisel ja nõudmiste paika panemisel uuriti konkurentide pakutavaid lahendusi ja meetodeid. Rakenduse üheks oluliseks osaks on piisavalt täpne ja kõiki osapooli rahuldav automaatne kalkulatsioonimootor, mida teeb päris hästi ja

[1] Smartecon `https://smartecon.ee/`
[2] Scoro `https://www.scoro.com`

põhjalikult Areva Solar Estonia [3], kuid kus ka portaali võimekus abistada projekti algusest lõpuni lõpeb.

Eesti Energial [4] on fikseeritud loetud arv pakette, milles arvestatakse hinna ja tootlikkusega, kuid puudub detailsem kohandamise võimalus pindalade, katusekallete ja teiste oluliste tegurite kohta, mis mõjutavad nii paigaldust kui ka tootlikkust. Samuti hakkab Elos paigaldusi pakkuma alltöövõtjate kaudu, mis tõttu saab olema rakenduse ülesanne koordineerida ka nende tööd, jagades sobivaid projekte ning pakkudes piisavalt infot edukaks paigaldmiseks.

Konkurente on ka Leedus [2, 3] ja Saksamaal [4, 5], kuid ei pakuta sellist rahvusvahelist täislahendust projekti haldamiseks esmasest kontaktist kuni paigaldatud projekti üleandmiseni kõigi rollide vaatest.

1.3 Eesmärk

Püstitatud eesmärgi täitmiseks on nad tellinud lahenduse autori ettevõttelt Net Groupilt[5], mis muudaks palju-arvuliste väiksema-mahuliste eraprojektide halduse efektiivseks ja mugavaks vähendades projektijuhtide koormust tellimuse, pakkumise, lepingute, maksmise ja paigalduse faasides.

Projekti tulevikuvisioon on rakendus, mis on kohandatud suunama ja abistama klienti kogu paigaldusprotsessi vältel läbi selliste etappide nagu kliendi poolt projekti tellimuse koostamine, automaatse kalkulatsiooni- ja hinnamootori abil pakkumise genereerimine, lepingute koostamine ja allkirjastamine, väljaostu ja rendi puhul tasumise võimalused, paigaldusprotsessi koordineerimine ja projekti üleandmise vormistamine niimoodi, et Elose projektijuht peab üldjuhul vähe sekkuma ja on pigem monitoorija ja kinnitaja rollis. Sealjuures seisneb rõhk ja ka äriline keerukus esiteks sellel, et kuidas muuta seni keeruline ja aeganõudev protsess kliendile piisavalt arusaadavaks, mugavaks ja kiireks, et ta suudaks läbida seda valdavalt iseseisvalt ning teiseks kujundada selline hinnamootor, mis üldjuhul suudaks automaatselt genereerida kõikidele osapooltele rahuldava pakkumise. Paigaldusprotsessi jaoks hakkab rakendus koondama enda alla erinevaid paigalduspartnerite ettevõtteid, kelle tööd rakendus koordineerida aitab ning kellele peab rakendus suutma pakkuda vajamineva informatsiooni paigaldusprotsessiks.

[3] Areva Solar Estonia `https://paikesepaneel.ee/`
[4] Eesti Energia päikesepaneelide lahendused `https://www.energia.ee/era/taastuvenergia/paikesepaneelid`
[5] Net Group `https://netgroup.com/`

Kõike seda peab terviklik rakendus suutma hallata erinevates riikides (esialgu Leedu, Eesti, Läti, Saksamaa), mis lisab keerukust nii keeletugede, valmislahenduste (näiteks makse- ja allkirjastamise lahendused) kui ka projektide manageerimise (erinevad projektid, tooted, partnerid, maksustamine) näol.

Käesoleva töö puhul on aluseks võetud minimaalse töötava toote skoop, mis sisaldab tellimuse vormistamise ja pakkumise ning lepingute koostamise faase, mida hakatakse Leedus rakendama. Lisaks tagarakenduse arendamisele vastutab autor rakenduse avalikustamise ning pideva integratsiooni ja pideva tarnimise eest.

1.4 Arendusmetoodika

Autori vastutada oli rakenduse tagakomponendi eest, paralleelselt arendati teise arendaja poolt kasutajaliidest, kellega käis tihe suhtlus töö, tähtaegade ja implementatsioonide koordineerimisel. Tiimi tööd koordineeris tiimijuht ning lisaks kasutati perioodiți ka analüütik-disaineri abi, kes aitas defineerida kriitilisemaid nõudeid ning disainis Figma prototüüpi. [6]

Agiilsetele metoodikatele kohaselt korraldati tööd jooksvalt kahenädalaste sprintidena, mil üritati võimalikult kiiresti kokkulepitud funktsionaalsused mingil kujul valmis saada, et seda kliendiga tagasisidestada ja testida ning seejärel vajadusel muudatusi sisse viia.

Otsustasime, et meil tasub oma igapäevastesse *stand-up* kohtumistele kaasata ka kliendipoolne esindaja, sest märkasime, et vajame sujuvamaks arendamiseks ning edasiste tegevuste suunamiseks pidevat kliendipoolset ärilist sisendit ning nii saime varakult tõstatada arendusprotsessi käigus tekkivad küsimused, millele klient sai hakata koheselt vastust otsima. Vastavalt vajadusele lepiti eraldi kokku pikemaid teemapõhiseid koosolekuid, kus võeti mingi konkreetne teema (näiteks kalkulatsioonid, tooted või lepingud) põhjalikult arutluse alla.

Iga sprindi lõpus toimus möödunud sprindi *retro* ning uue sprindi planeerimise koosolek, kus arutati läbi sprindi skoop üldisemate nõudmiste näol. Kokkulepitu info põhjal täpsemate arenduspiletite kirjeldamine ning sprindi sisese arendusplaani koostamine jäi arendaja enda koostada.

Arendusprotsessi teekonnal loogilistel hetkedel toimusid demod ja kasutusmugavuse testimised, et valideerida ja testida tehtud tööd koos tulevaste rakenduse kasutajatega. Hiljem hakati regulaarselt korraldama ka juhtkonna koosolekuid, kuhu kaasati Smarteconi Eesti ning Leedu esindajaid, kes tagasisidestasid rakenduse funktsionaalsusi ning kus arutati

järgmisi samme ja tähtaegu.

Täiesti uue projektina ei olnud ei kliendi ega ettevõtte poolt esitatud ühtegi nõudmist valitud tehnoloogiatele või lahendustele, nii et autoril oli vaba valik, mida ta kasutab, kui see täidab kliendi nõudmisi rakenduse funktsionaalsusele ja toimimisele.

2. Rakenduse ülevaade

Järgmisena seletatakse lahti rakenduse voog ja kasutajate funktsionaalsus selle läbimisel, mida üritati paika kalibreerida testimiste käigus päris mitu korda. Arvestama pidi sellega, et millal ja kuidas kuvada rollidele informatsiooni nii, et kõik oleks neile piisavalt hallatav, mis infot ja kui detailselt küsida, millal keegi mis infot muuta saab, et säiliks vajalik paindlikkus, aga samas ei tekiks vigu ega ka konflikte põhjustavaid liiga hiliseid muudatusi.

2.1 Avaliku portaali funktsionaalsus

Avalik portaal, kuhu suunatakse uus potentsiaalne klient, sisaldab endas mitmete sammudega sisendandmete vormi, mille eesmärk on saada kliendilt piisav hulk informatsiooni projekti piisavalt täpselt hindamiseks ja pakkumise koostamiseks, kuid samal ajal tuleb hoida see hoida piisavalt lihtne, et selle läbimine oleks kiire, arusaadav ja jõukohane.

2.1.1 Sisendinfo valikute ja toodete serveerimine

Tagarakenduse ülesanne on serveerida klientidele vajalikku infot sisendandmete sisestamiseks. Näiteks päritakse tagarakendusel tarbimisharjumuste ja katusekallete valikuid, kuna pidi arvestama, et need sõltuvad kasutaja asukohast – ühes riigis võib tarbimine sama suurte leibkondade puhul olla märgatavalt erinev, katusekalle mõjutab päikesekiirte langemisnurka ning paigalduspindala, mis mõjutab märgatavalt tootlikkust ning võib samuti riigiti erineda.

Samuti päritakse tagarakenduselt kombineeritud pakettide valikuid, mis koosnevad mitmetest komponentidest: paneeli mudel, inverteri bränd, garantiid ja lisatoodete (akud, laadimisjaamad, lisagarantiid) valik ning mis võib samuti sõltuda asukohast ja hiljem isegi kliendi valitud projekti mõõtmetest.

2.1.2 Projekti suuruse, komponentide ja hinna kalkuleerimine

Tagarakendus kalkuleerib kliendile pärast nõutud sisendandmete kogumist projekti suuruse selliste väärtustega nagu katuse ja paigaldamise alla minev pindala, süsteemi võimsus ja aastane tootlikkus, arvestades ka paneelide efektiivsuse hääbumist ajas. Samuti arvutatakse veel projekti potentsiaalsed minimaalsed ja maksimaalsed suurused, et klient saaks nende põhjal liugkomponendi abil oma projekti suurust lubatud vahemikus vähendada või su-

urendada vastavalt enda soovile. Ühtlasi valideeritakse, kas projekti elluviimine tasuks äriliselt ära – vastasel juhul suunatakse klienti sisendandmeid muutma (näiteks vahetama paketti või otsima suuremat ala paneelide paigutamiseks).

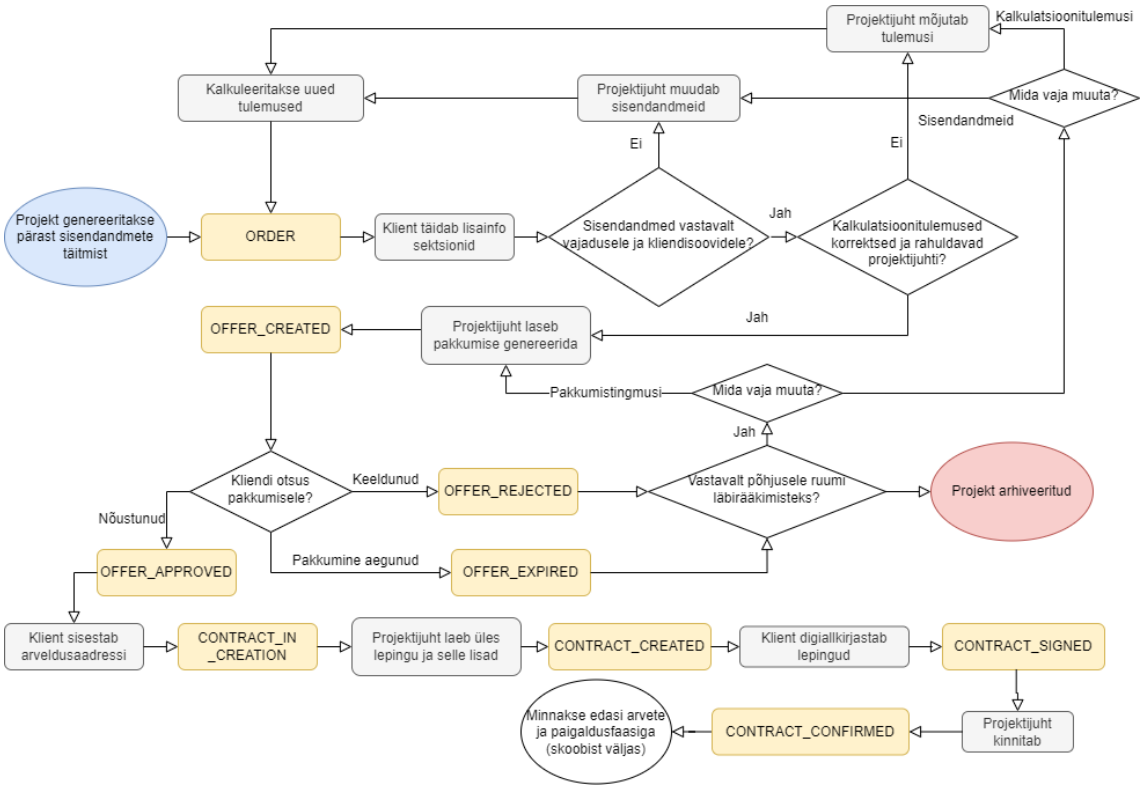
Vastavalt projekti suurusele ja sisendandmetele arvutatakse vajaminevate komponentide arv ning automaatselt valitakse juurde vastava inverteri brändi konkreetne inverter, mis rahuldaks antud süsteemi võimsust. Klient saab valida ka tasumisviisi - kas soovib projekti välja osta või rentida mingiks ajaperioodiks. Sõltuvalt sellest ning projekti mõõtmetest arvutatakse detailne projekti osade maksumus, rakendades juurde ka kasumimarginaali. Seejärel arvutatakse lõpphinnad, kus arvestatakse ka käibemaksu ning rentimise valiku puhul liitintressid ja kuutasud.

2.1.3 Kasutaja ja projekti loomine ning kokkuvõtte meilide saatmine

Pärast tellimuse koostamist ja esitamist tehakse viimased validatsioonid ning õnnestumise korral luuakse projekt, mille külge lisatakse veel vajalikud andmed (näiteks lisainfo sektsioonid, mida klient on ohustatud hiljem portaalis täitma) ning luuakse suvalise parooliga kasutaja kliendile. Seejärel saadetakse Elose ametlikule emailile kiri koos projekti esmaste andmetega PDF kujul ning kliendi poolt sisestatud aadressile kinnitav email koos sisselogimiseks vajalike andmetega (vt Lisa 2).

2.2 Autenditud portaali funktsionaalsus

Järgmisena kirjeldatakse autentitud portaalide (kliendi ja Elose projektijuhi) lõputöö skoopi jäävat funktsionaalsust. Joonisel 1 on kujutatud hetkel implementeeritud vooskeem, mida mööda projektijuht ja klient liiguvad



Joonis 1. *Projekti voog*

2.2.1 Portaaliga autentimine ja projektide ülevaade

Autenditud (kliendi ja Elose projektijuhi) portaalidesse pääseb ligi autentides ennast õige kasutajatunnuse ning parooliga, mis on talletatud andmebaasi. Klient saab portaali sisse logides muuta enda parooli ja kasutaja andmeid. Elose töötaja saab samuti muuta enda parooli ja nii enda kui ka kliendiandmeid. Sessiooni lõppedes saab kasutaja ennast välja logida.

Esimese vaatena päritakse vastavalt rollile projektide nimekirja vaade. Projektijuhil on võimalik näha kõiki projekte kokkuvõtliku infoga ning neid filtreerida, klient näeb vaid oma projekte temale mõeldud lühiandmetega (vt Lisa 3). Kuna kuvatakse vaid ülevaatlikku versiooni, kuid infot päritakse paljudest tabelitest, on kasutatud projektsioone, et muuta need päringud efektiivsemaks.

2.2.2 Tellimuse faasi funktsionaalsus

Valides mingi konkreetse projekti, tehakse detailne päring projekti sisendinfo (objekti paiknemine Google Mapsil, kliendiandmed, hoone andmed, toodete andmed) ja kalkulatsioonitulemuste (projekti suurus, hind, vajaminevad lisatooted) kohta (vt Lisa 4).

Kui projekt on tellimuse faasis, saab projektijuht muuta nii projekti sisendandmeid, mille tulemusel käivitatakse uuesti ka kalkulatsioonide protsess, kui ka vajadusel mõjutada otse kalkulatsioonitulemusi (arvutatud mõõtmed, hind), mis mõjutavad omakorda üksteist. Varasemas versioonis sai klient samuti muuta enda sisendandmeid, kuid ärilistel põhjustel on see funktsionaalsus hetkel lukustatud.

Lisasektsiooni juures saab projektijuht lisada uusi kategooriaid (näiteks „Katuse pildid“), mille valik koos lisainfo ja näidistega päritakse andmebaasist (vt Lisa 4). Enne projektiga edasi liikumist peab klient vastavad kategooriad nõuetekohaselt täitma (antud näite puhul lisama pildifailid katusest).

2.2.3 Pakkumise faasi funktsionaalsus

Kui projektijuht ja klient on tellimuse andmed kooskõlastanud ja kinnitanud, saab projektijuht lasta genereerida automaatse pakkumise koos kehtivusajaga ning soovi korral täienda seda garantiide ja lisamärkustega.

Kui pakkumine on genereeritud, saab seda näha ka klient, millega ta saab kas nõustuda või keelduda. Viimase puhul küsitakse temalt põhjust, mida projektijuht saab kasutada kas tagasisidena või vajadusel teha muudatusi projekti sisendandmetes või rakendada allahindust.

Mõlemad saavad soovi korral lasta ka pakkumisest PDF versiooni genereerida ning selle alla laadida, mida on kujutatud joonisel 2. Vastavalt seatud keelele saab seda teha hetkel nii inglise kui ka leedu keeles.

elos OFFER

DETAILS

Offer number:	2
Valid from:	2023/03/21
Valid until:	2023/03/28
System power:	8.00 kW
Address:	Tammsaare tee 31, Tallinn, 12612
Package:	Design
Warranty:	3 years
Notes:	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Client:	Ain Kivimägi

ITEMS	AMOUNT
Panel *WST-MGX-P1/P3 FULL BLACK, GEMINI*	20
Inverter *SUN2000*	1
Ev charger *11 kW EV charger*	1
Battery *10 kWh battery*	1
Installation	1

Smartecon OÜ
info@smartecon.ee
+3725128115

Läike tee 32/3, 75312 Peetri, Eesti
Rg-kood: 12382136
KMKR: EE101626532

www.smartecon.ee
LHV: EE307700771002490196

elos OFFER

Installation accessories	20
Design and documentation	1

PRICE

Rent period:	10 years
Interest rate:	5 %
Compounding periods per year:	12
Price without VAT:	54787.68 €
VAT price:	10957.54 €
Total price	65745.22 €
Monthly rent price:	547.88 €
Monthly kW price:	68.49 €

CONTACT PERSON

First name:	Elos
Last name:	Company
Email:	test-elos@gmail.com
Phone:	987654321

Smartecon OÜ
info@smartecon.ee
+3725128115

Läike tee 32/3, 75312 Peetri, Eesti
Rg-kood: 12382136
KMKR: EE101626532

www.smartecon.ee
LHV: EE307700771002490196

Joonis 2. *Genereeritud pakkumise PDF fail*

2.2.4 Lepingu faasi funktsionaalsus

Pärast pakkumise genereerimist algab lepingute faas, milles kõigepealt peab klient sisestama oma arveldusaadressi. Kuna lepingud on jäigemad ja olulisemad dokumendid, siis hetkel jäetakse nende sisu kirjutamine projektijuhtidele, hiljem genereeritakse ka nende sisu automaatselt.

Küll aga saavad nad põhilepingu ning sellega kaasnevad osad failidena üles laadida, pärast mida loob tagarakendus nendest ASICE konteineri [7], mida mõlemad osapooled saavad alla laadida ning pärast käsitsi digiallkirjastamist uuesti üles laadida. Projektijuht saab pärast valideerimist lepingu kinnitada. Hiljem lisandub Elose poolne automaatne digiallkirjastamine. Sellega lõppeb ka antud lõputöö implementeeritud voog, kust edasi liigutakse arvelduste ja paigaldusfaasiga käsitsi.

2.2.5 Kommentaarium

Läbi kõigi faaside on projektiga seotud isikutel (esialgu projektijuht ja klient, hiljem ka partnerid) implementeeritud esmaseks suhtlemiseks ning juhiste ja informatsiooni jagamiseks

kommentaarium, kus kuvatakse ajalises järjekorras lehekülgedena kommentaaride sisu ja nende autor ning kuhu saab vaba tekstiväljana kommentaare juurde lisada (vt Lisa 4).

2.2.6 Muudatuste logi

Kuna projekti käigus võib andmetega toimuda palju muudatusi mõlemate osapoolte tegevuste tagajärjel, on oluline, et nendest muudatustest oleks teadlik ka vastasosapool. Kui klient on valinud näiteks paketi mingite lisadega ning projektijuht neid muudab, genereeritakse nii sellest kui ka muudatusest tingitud uutest kalkulatsioonitulemustest inimloetav muudatustelogi, kus on välja toodud muudetud objekt, eelnevad ja uued väärtused, muudatuse põhjustaja ning muudatuse aeg (vt Lisa 5).

Selle üle, kas vastav osapool on muudatusi näinud ning need kinnitatud, peetakse samuti arvet, et oleks võimalik seda osapoolt teavitada ja suunata teda muudatusi kinnitama, enne kui jätkatakse järgmiste tegevustega. Muudatuslogisid päritakse esmalt kinnitamata ning seejärel ajalises järjekorras efektiivsuse eesmärgil lehekülgedena. Rakendus on suuteline tõlkima muudatuslogisid nii inglise kui ka leedu keeles päringu tegemise hetkel, s.t hoolimata muudatuslogi genereerimise ajal kehtivast keelest päritakse talle need hetkel kehtivas keeles.

Lisaks kuvatavale muudatustelogile logitakse ka projekti edenemisi staatuste vahel, mille põhjal saab projektide kulgemisi ajas jälgida ning teha statistikat, näiteks mis staatustes on pudelikael suure ajakulu mõttes ning kelle tõttu ning mis tingimustes projektidest loobutakse.

2.2.7 Teavituste süsteem

Selleks, et projekti edenemine läbi eri faaside kulgeks võimalikult kiirelt ja sujuvalt, on implementeeritud ka e-maili põhine teavituste süsteem, mis saadab vastavale osapoolele emaili, kui temalt oodatakse mingisugust tegevust. Näiteks, kui projektijuht on geneerinud pakkumise või seda muutnud, saadetakse kliendile teavitus, et temalt oodatakse otsust. Vastupidiselt, kui klient on pakkumise suhtes otsuse teinud, saadetakse projektijuhile teade, et temalt oodatakse edasi lepingute vormistamist. Rakendus on võimeline kirju saatma nii inglise kui ka leedu keeles sõltuvalt sättest.

3. Tehnoloogiline lahendus

Järgmisena esitatakse eelnevalt välja toodud funktsionaalsuse implementeerimiseks analüütilisel teel valitud tehnoloogiad, implementeeritud arhitektuur ja lahendusviisid.

3.1 Kasutatud tehnoloogiad

Rakendus on kirjutatud Javas, mis on üks populaarsemaid veebiplatvormide serverirakenduste arendamiseks kasutatavaid keeli, olles objekt-orienteeritud keel, platvormist sõltumatu, turvaline, skaleeritav, paljude APIdega, põhjalikult dokumenteeritud ning milles on autoril ka kõige suurem kompentents. [8] Rakenduse loomiseks on kasutatud Spring raamistikku, mis on Java veebirakenduste *de facto* standard, olles lihtsasti ülesseadistatav, kiire, turvaline, paindlik ning hõlmates võimsa funktsionaalsusega raamistikke, millest autor kasutab Spring Web'i, Spring Data JPA'd, Spring Security't ja Spring WebFluxist WebClientit. [9, 10]

Ehitustööriistana kasutab autor Gradlet. Kuigi Maven on vanem ning autorile koolist tuttavam variant, eelistab järjest rohkem arendajaid Gradlet näiteks selle optimeerituse (kordades kiirem), personaliseerimise ja pluginate poolest. [11, 12]

Andmebaasi tasandil kasutatakse PostgreSQL andmebaasi manageerimise süsteemi, mida peetakse tänapäeval üheks turvaliseimaks valikuks, olles tasuta, kiirelt arenev, turvaline, efektiivne keeruliste andmelugemiste ja -muudatuste korral ning pakkudes palju hästi implementeeritud lisaväärtusi võrreldes teiste turul olevate SQL ja NoSQL gigantidega (mõneks näiteks suur valik andmetüüpe, mitte-relatsiooniliste andmete salvestamine, ulatuslikud laiendusvõimalused). [13, 14, 15] Kiireks ja mugavaks arendusprotsessiks ning testide jooksutamiseks kasutatakse veel mälupõhist H2 andmebaasi, mis luuakse iga kord rakenduse käivitamisel, lihtsustades ja kiirendades arenduse, katsetamiste ja testide jooksutamise protsessi. [16]

Andmebaasi skeemi implementeerimiseks ja migratsioonideks kasutab autor Liquibase'i. Liquibase töötab muudatuslogide põhimõttel, mis võimaldab andmebaasi muudatusi jälgida ning vajadusel taastada andmebaasi oleku mingisse ajahetkesse. See pakub tuge paljude erinevate andmebaaside jaoks, mis võimaldas kasutada arendusprotsessis H2 andmebaasi ning sujuvalt üle minna PostgreSQL kasutamisele päriskeskkondades. Alternatiivne lahendus oleks olnud Flyway, kuid autor otsustas Liquibase kasuks tema rohkemate võimaluste

ja paindlikkuse tõttu. [17, 18]

Funktsionaalsusnõuete kohaselt on vaja eri osapoolte vahel saata palju automaatselt genereeritud eriilmelisi teavitusmeile ning pakkudes lähitulevikus ka eri keelte jaoks tuge mille tõttu nägi autor vajalikuks võtta kasutusele mõni mallimootor. Spring ühildub hästi päris mitmete populaarsete mallimootoritega, millest kõige sagedamini on erinevates artiklites välja toodud ühiselt Thymeleaf, FreeMarker, Mustache ja Groovy. Thymeleaf, FreeMarker ja Mustache kasutavad kõik mallide kirjutamisel autorile tuttavat HTML sarnast süntaksit, Groovy'l on spetsiifilisem *builder* süntaks, Mustache'l jälle puuduvad sellised operatsioonid nagu *if-else* laused ja *for-loopid*. Thymeleaf ja FreeMarker on IDEde poolt kõige paremini toetatud ning Thymeleafil on kõigist tunduvalt põhjalikum dokumentatsioon ja näited ning pakub rohkem versatiilsust näiteks laienduste näol. Seetõttu tundus nende seast kõige sobivam Thymeleaf. [19, 20] Thymeleafi abil genereeritud emaile saadetakse JavaMail API abil. [21]

Lisaks oli vaja luua esinduslikum disain saadetavatele emailidele, mida peetakse arendajate seas keeruliseks ja ebamugavaks ülesandeks, kuna erinevad emaili-kliendid käituvad väga erinevalt, on turvatud failide ja fontide vastu, mõndasid elemente ei kuvata või kuvatakse neid ebaühtlaselt. [22] Nende probleemide lahendamiseks kasutab autor MJML raamistiku abi, mis võimaldab arusaadava süntaksiga üsna lühikeselt kirja kujunduse defineerida, kuid tagataustal genereeritakse väga põhjalik HTML ja CSS kood, mis defineerib kirja disaini arvestades ka eri ekraanisuuruste ja klientidega. Selle funktsionaalsus ei piirdu seal, vaid teoorias võimaldab lihtsasti luua ka keerulisemaid elemente ja animatsioone. [23]

Projektihaldustarkvarana moodustab funktsionaalsusest olulise osa PDF failide genereerimine. Üks populaarseimatest ja võimsamatest ning autori poolt varem kasutatud on iText teek, kuid selle kasutamist piiravad litsentsid. [24] Järgmisena tundus kõikidele rakenduse nõuetele vastav ja kõige lihtsamini kasutatav Apache PdfBox, mis on ka kommertslikult tasuta kasutatav. PdfBox pakub palju lisafunktsionaalsust ka näiteks mitmete PDF failide ühendamiseks, allkirjastamiseks ja olemasoleva PDF malli infoga täitmiseks, mis osutub tulevikuks kasulikuks näiteks pikkade kindlalt struktueeritud lepingute genereerimiseks. [25]

Lepingute ja muude failide jaoks, mida peavad mõlemad pooled digiallkirjastama, kasutatakse ASICE [7] failiformaati. Nende failikonteinerite loomiseks võttis autor kasutusele Java DigiDoc4j teegi. Hetkel genereeritakse sellega lihtsalt üks konteiner erinevatest lepingute osadest, mille saavad mõlemad osapooled allalaadida ning allkirjastada. Edaspidi, kui rakendusel on ligipääs digiallkirjadele, saab selle teegiga hõlpsasti implementeerida ka automaatse digiallkirjastamise kohe lepingufaili genereerimisel. [26]

Autenditud portaali muudatuslogi funktsionaalsuse implementeerimise üheks osaks on muudetavate erikujuliste objektide võrdlemine, et saada kätte erinevusi, mille jaoks oli vaja toetavat teeki. Üks nendest on Envers, mida kasutatakse objektide versioniseerimiseks ja millega saab implementeerida näiteks taastamist, kuid antud olukorra nõuete puhul on see liigne funktsionaalsus, genereerides väga palju tabeleid ja muutes üsna keeruliselt päritavaks sellist üldist logi. [27] Teise variandina leidis autor Javersi, millel on ka lisafunktsionaalsusi, kuid võrdlemismehhanism on suhteliselt lihtsakoeline ning kergesti konfigureeritav. [28]

Päikesepaneelide tootlikkuse arvutamine sõltub väga paljudest teguritest ning selle asukohapõhine arvutamine on keeruline protsess, mille rahuldav sooritus nõudis mingisuguse välise tööriista saavutamist. Alguses leiti vaid 3 tasulist varianti, millest kaaluti kasutada autorile tuttavat OpenWeatherMap APId, mis pakub koordinaatide põhiselt ajaloolisi andmeid päikeseradiatsiooni kohta, kuid see on tasuline ning lõviosa kalkuleerimisest võttes arvesse parameetreid nagu süsteemi võimsusus ja katuse nurk oleks vaja olnud ise ära teha. [29] Pärast teist otsimislainet avastati API toega Euroopa Teadusuuringute Ühiskeskuse (Joint Research Centre ehk JRC) poolt pakutava tasuta kasutatava tööriista, kus on väga täpselt võimalik anda lisaks koordinaatidele selliseid sisendandmeid nagu paneelide efektiivsustegur, kalle, asimuut, süsteemi nominaalvõimsus, paigaldusviis (katus või maapind), valida eelistatav andmebaas ning nende põhjal kalkuleerida kokkuvõtlikku ja kuupõhist tootlikkust, arvutada hinda ja muud lisainformatsiooni. Kõike seda võimaldab see tasuta teha 30 päringut sekundis, mis on esmaste kavatsuste juures rohkem kui küll. [30]

Nii selle API kui ka tulevikus mitmete teiste APIde pärimiseks oli vaja raamistikku, mis vajalikke HTTP päringuid teha suudaks. Springi sisseehitatud standard on eelnevalt olnud RestTemplate, mida nüüd loetakse aegunuks ning eelistatakse WebClientit, mis on Spring Webfluxi raamistiku osa. See võimaldab muuseas teha asünkroonseid päringuid, mis kauakestvate päringute puhul ei hoia ühte threadi kinni, vaid võimaldab sellel samal ajal täita muid tegevusi.[31]

Konfiguratsioonifailides kasutatavate paroolide krüpteerimiseks kasutab autor Jasyptit. Jasypt on Springiga väga mugavalt integreeritav ning kasutatav vajamata krüpteerimispõhiseid teadmisi, kuid pakub samas suuri konfigureerimisvõimalusi, võimaldab krüpteerida kõiki paroole, tekste, arve, binaare ning omab palju muid kasulikke featuure. [32]

Kasutaja paroolide genereerimise ja parooli reeglite jõustamiseks kasutatakse Passay'd, mis võimaldab mugavalt defineerida parooli reeglid (pikkus, tähemärkide, sümbolite ja numbrite arv) ning selle põhjal genereerida suvalisi paroole või valideerida kasutaja poolt

sisestatud paroole. [33]

Osa andmeid, mida päritakse andmebaasist, on harva muudetava iseloomuga (minimaalse töötava toote skoobis näiteks toodete paketid, rentimise parameetrid, garantiid), kuid mida päritakse väga tihti ning millele oleks pidevalt kiiret ligipääsu vaja. Selliseid andmeid varundab autor vahemälus, mille jaoks võeti kasutusele Caffeine, mis on kõrge efektiivsuse ja paljude valikuvõimalustega. [34, 35]

Koodistiili ja -puhtuse jaoks on autor kasutanud Lomboki teeki ning Spotless ja Errorprone pluginaid. Lombok on ülimalt populaarne teek, mis vähendab koodi võimaldades paari annotatsiooniga implementeerida *Getter*, *Setter*, *Constructor* ja *Builder* meetodeid, kahandades hulgaliselt koodiridade arvu ning võimaldades tähelepanu pöörata põhifunktsionaalsusele ja ärireeglitele. [36] Spotless on lisatud koodi ühtse ja puhta formaadi tagamiseks. Sellele pluginale on võimalik defineerida mitmeid reegleid, mille põhjal käsku sisestades ta terve koodibaasi vormindab. Ühtlasi pakub ta kaitset vormindamata koodi ehitamise vastu, s.t projekti ehitamise protsess ebaõnnestub, kui Spotless leiab, et kood ei ole reeglitele vastav (näiteks leidub kasutamata *import* lauseid). [37] Errorprone on samuti vastavalt vajadustele konfigureeritav plugin, mis aitab koodis tuvastada veaohtlikke kohti, andes märku, kui mingi rida on kirjutatud ebamõistlikult/veaohtlikult ning pakkudes alternatiivi. [38]

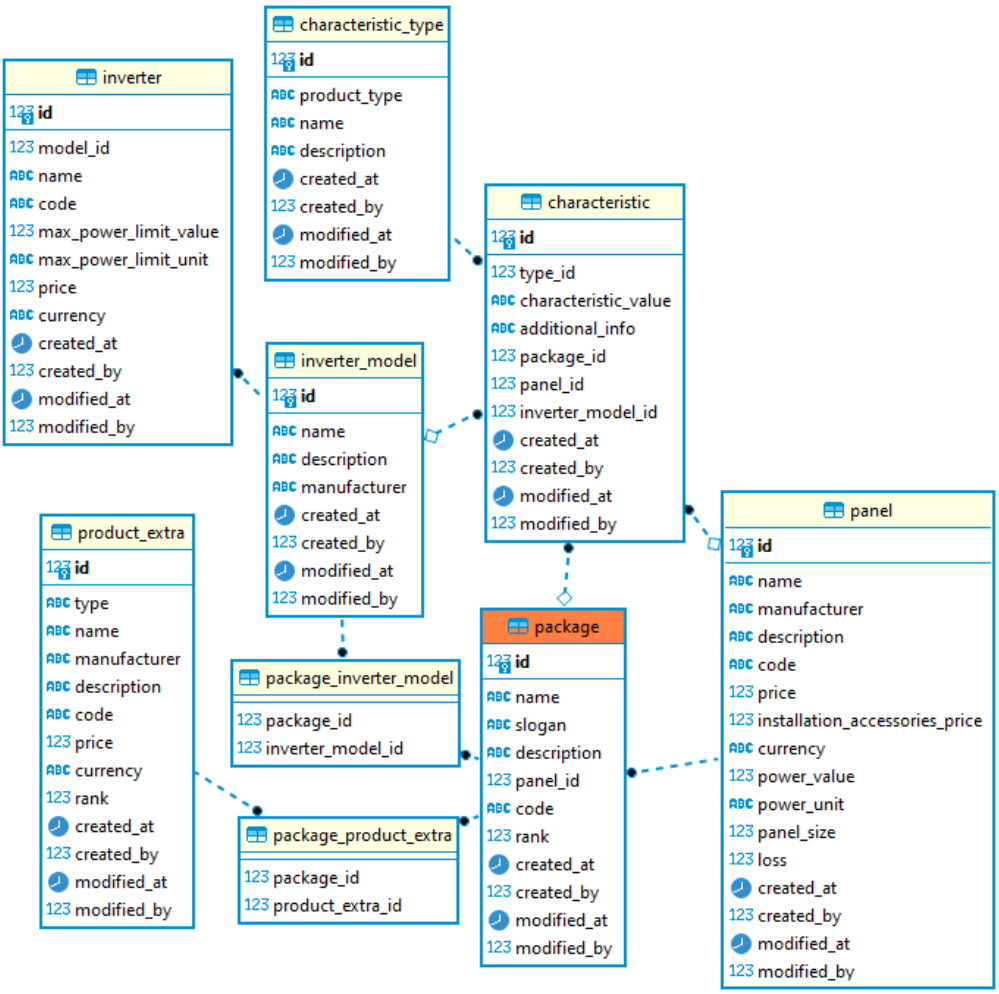
Arvukate automaattestide kirjutamiseks võttis autor kasutusele AssertJ teegi (leidub palju alternatiive, näiteks Google Truth, kuid kõik on sarnased ja üldjuhul erinevad lihtsalt maitseelistuse poolest), mis võimaldab laialdaselt mugavaid funktsioone erinevate *assert* lausete kirjutamiseks. [39] Ühik- ja kitsendatud skoobiga integratsioonitestide kirjutamiseks kasutatakse Mockito raamistikku, mis võimaldab lihtsalt siduda lahti testitav komponent teistest ja kontrollida, mis meetodeid kui mitu korda ja milliste parameetritena välja kutsuti. [40] Manuaalset testimist tehti Postmaniga. [41]

Andmebaasi disainimisel kasutati esialgu DbSchema rakendust, kus on väga mugav defineerida seoseid ja andmetüüpe ning mis suudab skeemist luua SQL laused ja vastupidi. [42] Kuid kui selle prooviversioon läbi sai, otsustas autor minna üle laialdaselt kasutatud ja kõrgelt hinnatud App Diagrams rakendusele, mis on väga universaalne ja toetab väga erinevate diagrammide loomist (valmiselemendid, näidismallid, pluginad, põhjalik stiliseerimine), integreerub paljude rakendustega (näiteks Google Drivega) ja pakub automaatset salvestamist ning erinevaid eksportimisviise. Seda kasutati antud projektis ka näiteks voo-, arhitektuuri- ja teiste skeemide jaoks. [43]

3.2 Andmebaasi arhitektuur

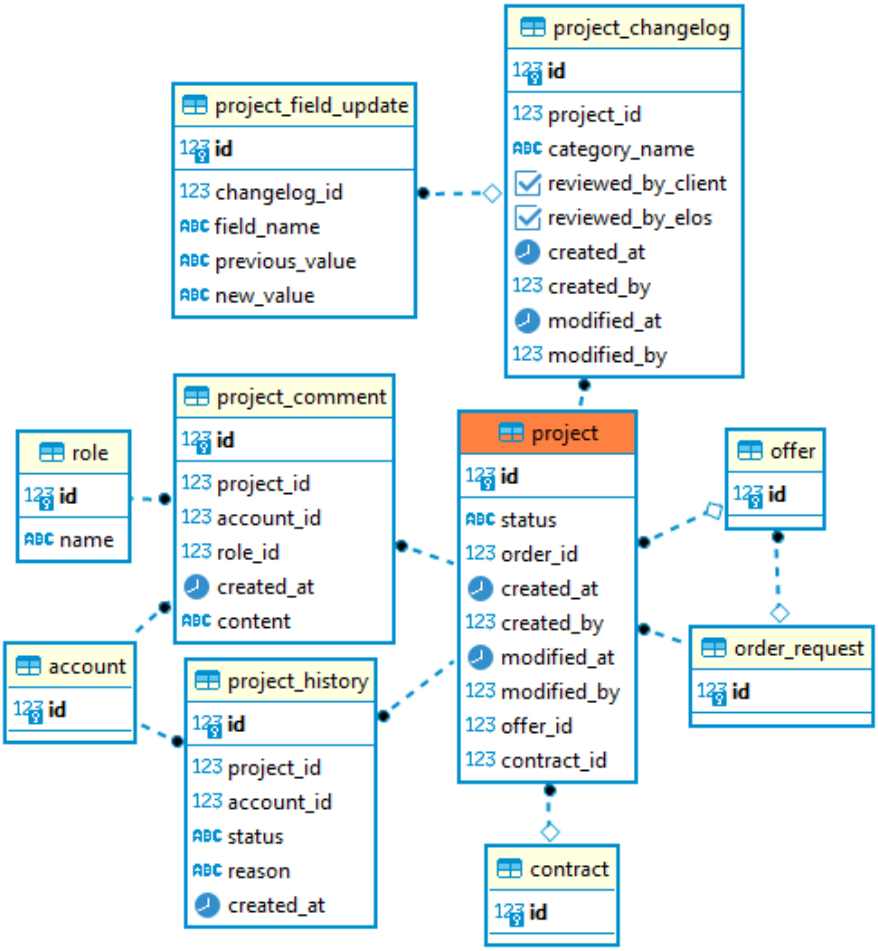
Järgnevalt edastatakse ülevaade andmebaasi skeemist, mis kirjeldab tagarakendusega seotud andmebaasi disaini. MVP skoobiga seoses on info talletatud 35 tabelisse, millest 31 on põhitabelid ning 4 mitu-mitmele relatsioonideks mõeldud vahetabelid. Täies mahus tabel on kujutatud Lisas 6.

Joonisel 3 on kujutatud toodetega seotud andmebaasi skeemi osa. Projekti põhitoodete info asub *panel*, *inverter_model* ja *product_extra* (akud, laadimisjaamad ja lisagarantiid) tabelites, igale tootele on võimalik dünaamiliselt lisada neid kirjeldavaid karakteristikuid (*characteristic_type* ja *characteristic*), näiteks tootepõhine garantii. Toodetest on kombineeritud paketid - paneel on konkreetselt määratud, inverterite mudelit ja soovitud lisatooteid on võimalik valida vastavalt soovile valitud paketi piires. Igale inverteri mudelile vastab üks või mitu inverter toodet, mille võimsustaluvused on erinevad.



Joonis 3. *Toodete andmebaasi skeemi osa*

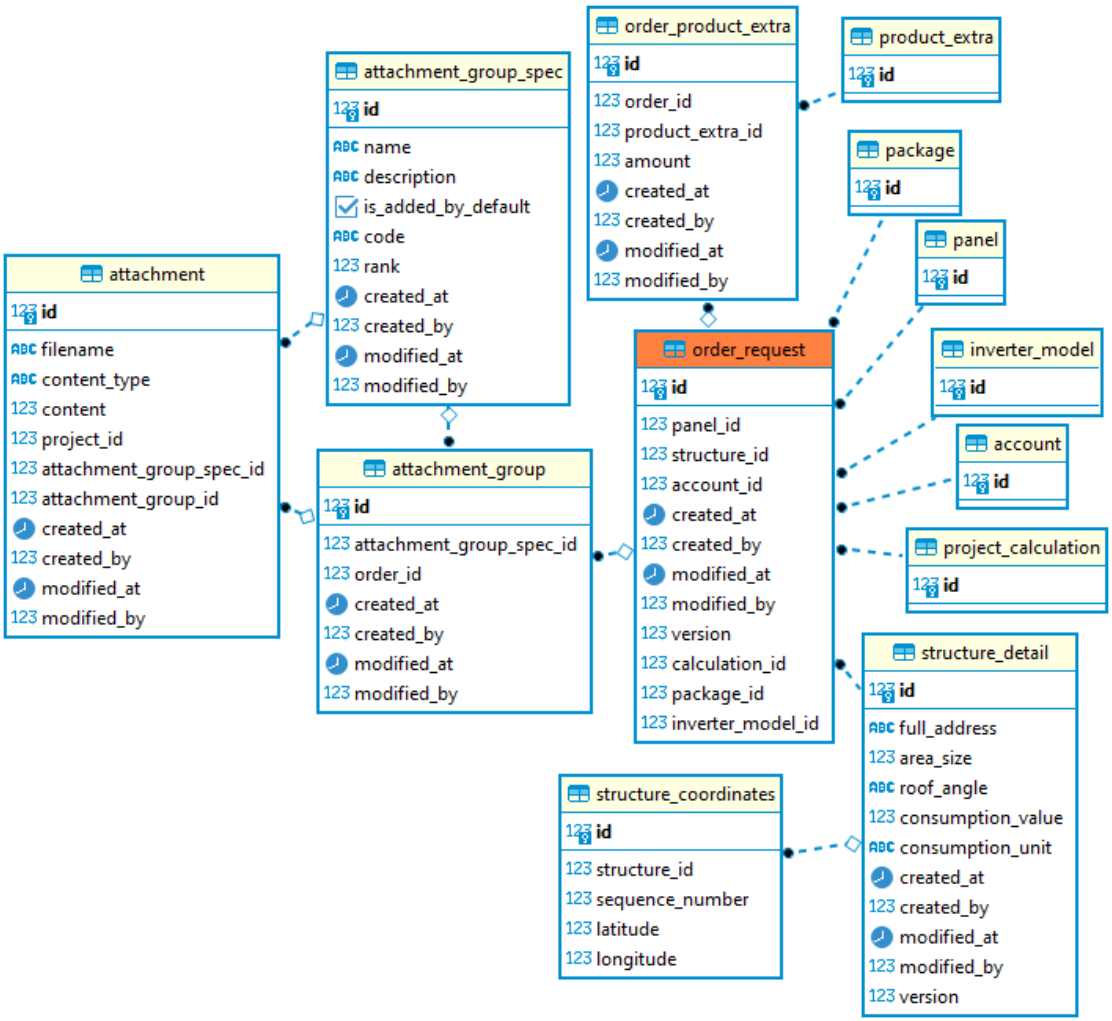
Joonisel 4 on kujutatud projekti tabel ja selle otsesed relatsioonid. Projekt on keskne objekt, millega seotakse tellimuse info, pakkumine, leping ja tulevikus ka paigaldusega seotud info. *Project_comment* tabelis hoitakse infot projekti külge postitatud kommentaaride kohta (selle sisu, postitaja ja postitusaja kohta). *Project_changelog* tabelis tähistab iga rida mingit muudetud objekti (näiteks muudeti hoone andmeid või kalkulatsioonitulemusi) ning kas see muudatus on kinnitatud nõutud osapoolte poolt. *Project_field_update* tabelis hoitakse mingi objekti nende väljade muudatusi, mille väärtus erines. *Project_history* tabelis hoitakse projekti eri faaside vahel liikumisi, liikumise toimumise aega, põhjustajat ning põhjust.



Joonis 4. *Projekti andmebaasi skeemi osa*

Joonisel 5 on kujutatud tellimusega ehk kliendilt küsitud sisendandmetega seotud andmebaasi skeemi osa. Tellimuse külge seotakse valitud pakett ning muutmiste korral konfliktide vältimiseks eraldi ka paneel ja inverteri mudel ning lisatooted. Hoone andmed asuvad *structure_detail* tabelis, kus juures Google Mapsi rakendusega joonistatud hoone polügoni punktid salvestatakse maha *structure_coordinates* tabelisse, et see joonis oleks taastatav. *Attachment_group* ja *attachment_group_spec* on seotud lisainfo sektsiooni andmete tal-

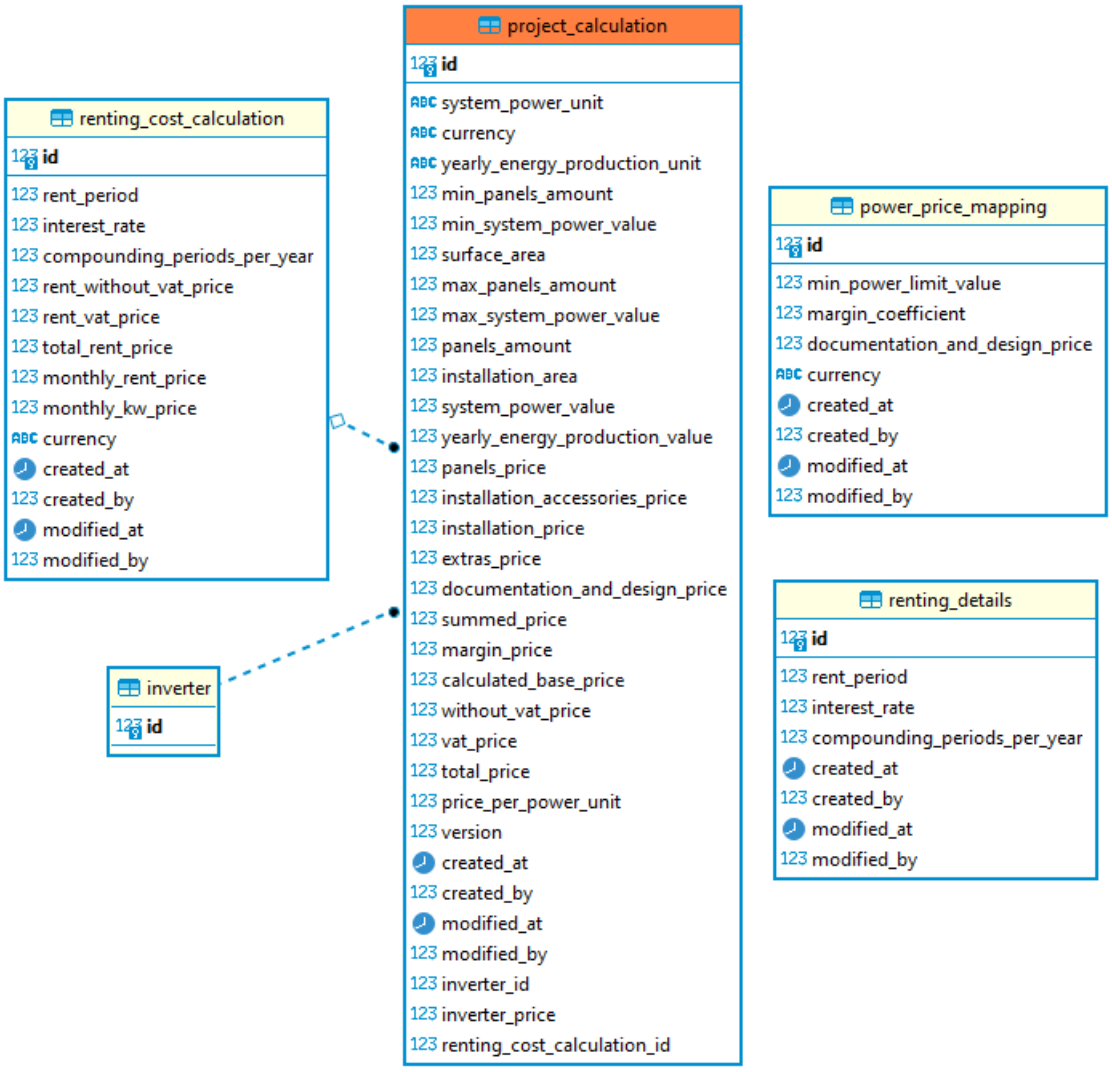
letamisega. Viimases on valik kategooriatest, mille kohta saab lisainfot küsida (näiteks pildid katusest või info elektrivõrgu kohta), näidisfailid asuvad *attachment* tabelis. *AttachmentGroupSpec* põhjal luuakse *AttachmentGroup* instants, kui see lisatakse konkreetse projekti külge kas projektijuhi või rakenduse poolt ning nende puhul kliendi poolt lisatud failid talletatakse samuti *attachment* tabelis. Sisendandmete põhjal tehtud projekti suuruse ja hinna arvutused asuvad *project_calculation* tabelis.



Joonis 5. *Tellimuse andmebaasi skeemi osa*

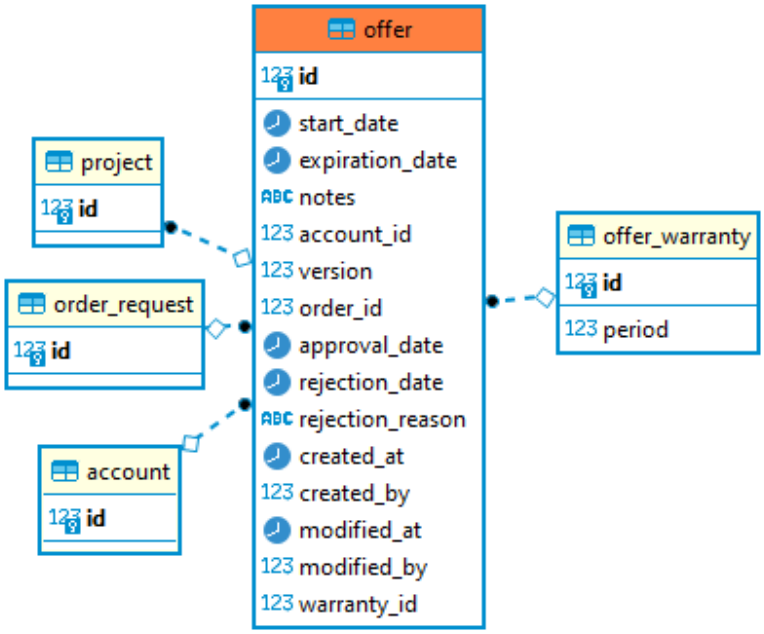
Joonisel 6 on kujutatud kalkulatsioonidega seotud andmeid. *Power_price _mapping* tabelis on projekti koguvõimsuse ja vastava kasumikoefitsiendi hinng dokumenteerimise hinna vasted, mida võetakse sisendina hinna kalkuleerimisel. *Renting_details* tabelis asuvad võimalikud rentimise perioodid ning nendele vastavad intressimäärad ja maksegraafik, mis võetakse sisendina rendihinna kalkuleerimiseks, kui klient on selle valiku teinud. *Project_calculation* tabelis asub kalkulatsioonidega seotud info: süsteemi võimsus, aastane tootlikkuse hulk, paneelide arv, paigalduse alla kuluv ala ning vastavalt süsteemi

võimsusele ja valitud inverteri mudelile ka konkreetne sobiv inverter. Selle info põhjal arvutatakse ka eri komponentide hinnad ning vastavalt ostu või rendi valikule ka lõplik hind ja ühiku hind koos käibemaksu arvutustega. Lisaks arvutatakse projekti minimaalsed ja maksimaalsed piirväärtused, mille raames saab projekti soovitud suurust mõjutada.



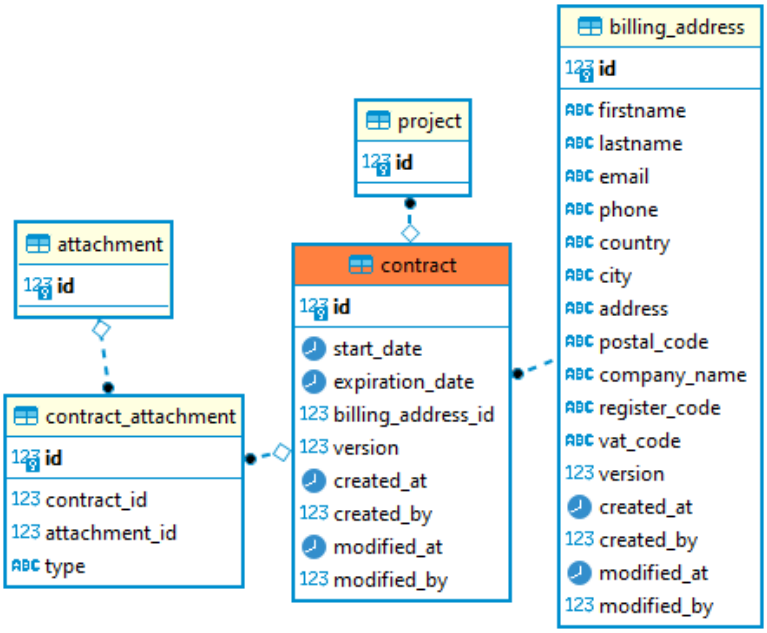
Joonis 6. *Kalkulatsioonide andmebaasi skeemi osa*

Joonisel 7 kujutatakse pakkumise informatsiooni. *Offer_warranty* tabelis hoitakse valikuid, mis garantiid saab pakkumisele määrata, tulbas notes saab projektijuht lahti kirjutada lisatingimused või teadaanded. Pakkumisel on kindel kehtivusaeg, pärast mida klient ei saa enam iseseisvalt otsust vastu võtta ning ühtlasi pakkumisest keeldumise puhul talletatakse ka keeldumise põhjus.



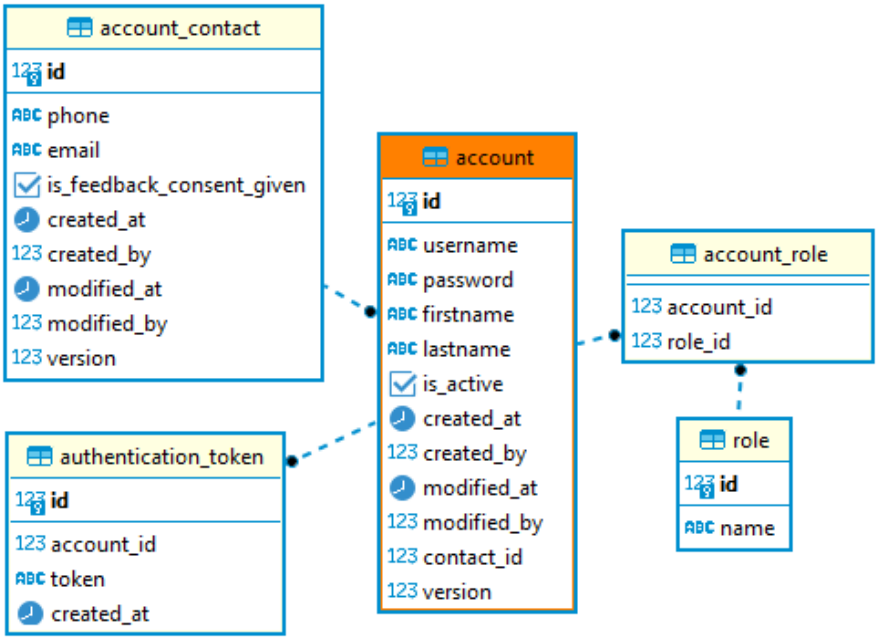
Joonis 7. *Pakkumise andmebaasi skeemi osa*

Joonisel 8 kujutatakse lepingutega seotud informatsiooni. *Billing_address* tabelis hoitakse kliendi arve andmeid, arve saab taotleda nii ettevõttena kui eraisikuna. *Contract_attachment* tabelisse salvestatakse lepinguga seotud põhilepingu PDF, võimalikud lisade PDFid ning vastavalt projektijuhi ja kliendi poolt digiallkirjastatud ASICE failid.



Joonis 8. *Lepingu andmebaasi skeemi osa*

Joonisel 9 kujutatakse kasutajaga seotud andmeid. *Authentication_token* tabelisse salvestatakse ära autentimiseks vajaminev *token* ehk pikk sõne, mis lisatakse kliendi päringutesse küpsisena, et edasistel päringutel teaks tagarakendus, kellega tegu ja mis ressurssidele teda ligi lasta. *Account_contact* tabelis salvestatakse hetkel ainult kliendi telefoninumber ja kontaktemaili aadress ning kas isik on märkinud, et temaga võib emaili teel ühendust võtta tagasiside ja paranduste eesmärgil, hiljem saab sinna külge kinnitada profiilipõhist informatsiooni (näiteks partnerite puhul nende andmed).



Joonis 9. *Kasutajate andmebaasi skeemi osa*

Tabelites, mille info on muutliku iseloomuga, on veerg *version* optimistliku luku implementeerimiseks. See võimaldab vältida konfliktide tekkimist, kui näiteks projektis toimub mitmeid muudatusi samal ajal või mingi osapool üritab otsust langetada aegunud info põhjal.

3.3 Rakenduse implementatsioon

Järgmisena antakse ülevaade rakenduse implementatsioonist. Esimesena käsitletakse rakenduse arhitektuuri ning selle komponente, seejärel esitatakse andmebaasi disain ning REST-liideste implementatsioon ja viimasena käsitletakse veidi põhjalikumalt ka rakendusega autentimise ja autoriseerimise protsessi.

3.3.1 Rakenduse arhitektuur ja komponendid

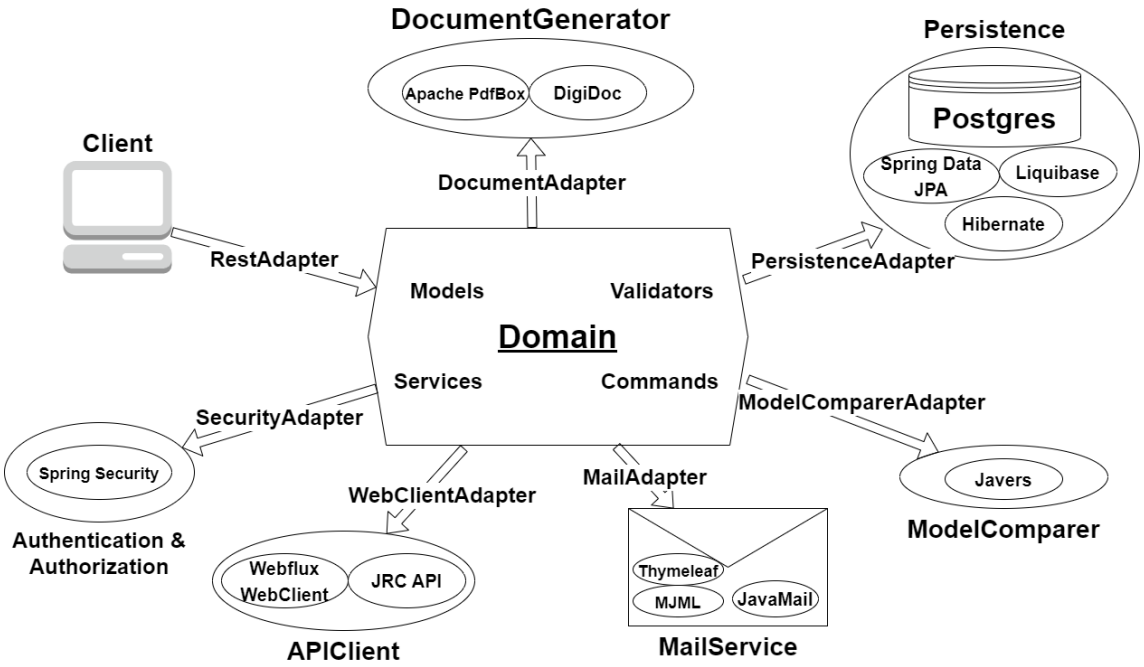
Rakenduse arhitektuuri disainimisel võeti malli heksagonaalse arhitektuuri põhimõtetest. Heksagonaalse arhitektuuri põhiselt on rakenduse tuumaks domeen, kus paikneb äriloogika. Selle külge kinnituvad portide abil rakenduse eri komponendid adapteritena, mis on mingid konkreetsed implementatsioonid kasutades spetsiifilisi tehnoloogiaid ja raamistikke. Õpiku järgi kuulub heksagonaalse arhitektuuri põhimõtete juurde veel palju mõisteid ja mustriosasid nagu kasutusjuhtude mudel, kuid antud projekti implementatsioonis on muudetud arhitektuuri abstraktsemaks, et vähendada keerukust ja kiirendada arendusprotsessi. Heksagonaalse arhitektuuri lähenemisel on mitmeid eeliseid. Esiteks muudab see rakenduse paindlikumaks ja tehnoloogiatest sõltumatuks. Kui muudetakse mõne komponendi implementatsiooni või vahetatakse tehnoloogiat, ei mõjuta see ärifunktsionaalsust. Komponendid on teineteisest eraldatud ning aitavad fookust hoida ühe komponendi ülesandel, mis parandab loetatavust, hallatavust, skaleeritavust ja testitavust. [44]

Konkreetse projekti implementatsiooni puhul, mida on kujutatud joonisel 10, on domeenis defineeritud Model klassid, mis kirjeldavad ärilisi andmeid ja seoseid. Rakenduse ärilised nõuded ja loogika jagunevad Service klassidesse, mis omakorda sisaldab Validator klasse. Viimased vastutavad ärireeglite ja andmete valideerimise ning vajadusel vastavate erindite viskamise eest informatiivsete veateadetega (kus juures erindite loomisel pidi arvestama, et peavad toetama mitmekeelseid veateateid). Domeeni külge kinnituvad rakenduse erinevad komponendid adapteritena. Hetkeses rakenduses on järgmised komponendid (visualiseeritud joonisel 10):

1. Andmebaasi komponent - vastutab andmete talletamise eest. Sisaldab Entity ja Repository klasse, kus on andmebaasi efektiivsemate päringute tegemiseks mängitud nende omavaheliste seoste ning projektsioonidega ja vastavalt konkreetsele juhule loetakse vaid vajaminevaid andmeid. Domeeniobjektideks ehitatakse sealsamas Adapter klassides.
2. REST komponent - vastutab esirakenduse päringute valideerimise ja vastuvõtmise ning vastuste vormistamise eest
3. Turvalisuskomponent – vastutab autentimise ja autoriseerimise eest kasutades Spring Security raamistiku funktsionaalsust
4. Dokumentide komponent – genereerib PDF faile ja loob ASICE failikonteinereid.
5. Veebiklient – vastutab väliste APIdega suhtlemise eest kasutades Spring WebClientiga. Hetkel on integratsioon Join Research Center APIga kalkulatsioonide jaoks vajalike andmete pärimiseks.
6. Emaili komponent – vastutab meilide saatmise eest kasutades JavaMaili ning genereerides sisu Thymeleaf mallimootori ja ekraani- ja brauseritundliku kujunduse

MJML abil.

7. Muudatuslogide generaator – kasutab Javersi, et genereerida inimloetavaid muudatuslogisid.



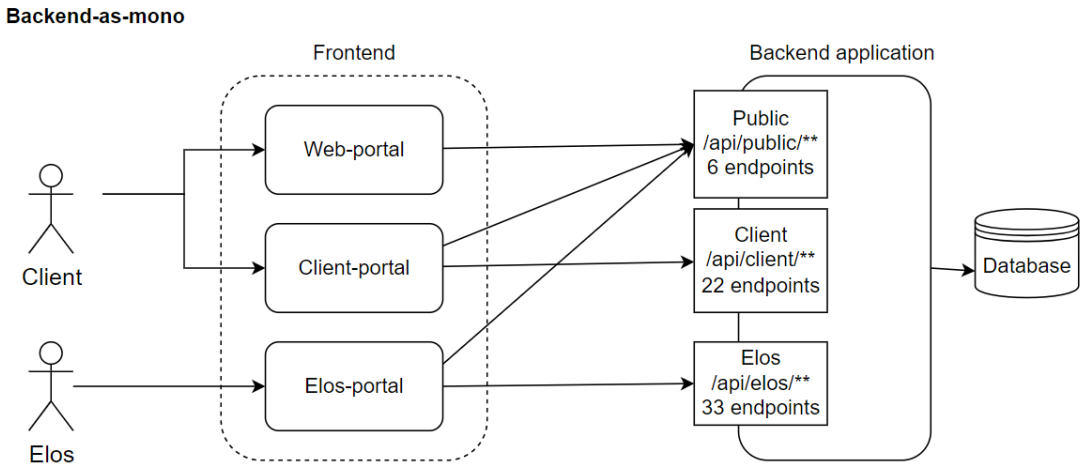
Joonis 10. *Rakenduse arhitektuur*

Tulles tagasi eeliste juurde, kui näiteks andmebaasi *Entity* objektid ning andmemuudatusoperatsioonid sisalduksid otse domeenis, toimuks vajalike andmete kokkupanemine üheskoos äriloogikaga ning samuti toimuks äriloogikaga koos andmebaasi-spetsiifilised toimingud nagu näiteks transaktsioonide haldamine ja tagasivõtmise strateegiad, mis muudaksid koodi hulga keerulisemaks ja segasemaks. Sellise implementatsiooni puhul võib ka domeen jääda samaks, isegi kui andmebaasiskeemis toimuvad mingid muudatused näiteks efektiivsuse parandamiseks. Samuti implementeeritud arhitektuuri tõttu ei pea näiteks domeen midagi teadma PDFi disainimise implementatsioonist ning vastupidi ei pea PDF generaator ise tegelema andmebaasist vajalike andmete pärimisega, vaid eeldab, et domeenikihist on need talle juba serveeritud. Selline abstraktsus ja sõltumatus võimaldab hiljem jaotada rakenduse ka hõlpsalt mikroteenusteks vastavalt komponent-teenustele.

3.3.2 REST-liidesed

Rakenduse REST-kihis asuvad lõpp-punktid, millest igale ühele on määratud aadress ja mille poole rakenduse kasutajad pöörduvad, et pääseda ligi rakenduse ressurssidele. Seal kihis teisendatakse ärimudeli põhised objektid vastavalt päringule andmeedastusobjektideks ning vastupidi sissetulevad ressursid ärimudeli objektideks. Ühtlasi vastutab REST-kiht

sissetulevate andmete esmaste valideerimiste eest. Antud rakenduses on eraldatud avaliku, kliendi ja Elose REST-liidesed, mis on kujutatud joonisel 11.

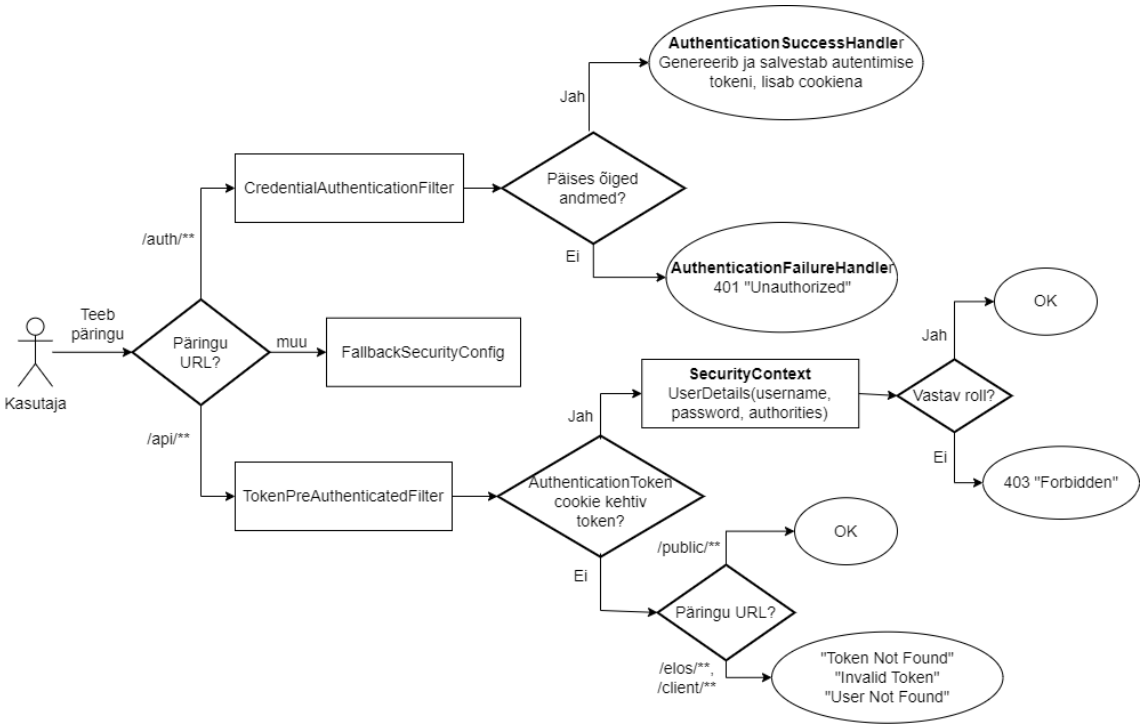


Joonis 11. *REST adapterid monoarhitektuurses rakenduses*

Antud liidesed on kirjutatud nii, et nad oleks teineteisest sõltumatud. Nii on uute rollipõhiste funktsionaalsuste arendamisel need konkreetselt ja arusaadavalt eraldatud ning iga lõpp-punkti puhul saab täpselt määrata, mis infot iga kasutaja mingi päringu korral kätte või sisestada saab – näiteks võib projektijuht kalkulatsioonitulemusi nähes näha põhjalikumat informatsiooni kui klient või vastupidi sissetuleva päringu näitel saab projektijuht parandada kõikide kasutajate kontaktandmeid, klient ainult enda oma. Tulevikus mikroteenusteks jagamisel on ühtlasi siis eri portaalide liidesed juba eraldatud ning uute admin ja partneri liideste implementeerimine samuti lihtne ning mida arendades ei puudutata varasemaid.

3.3.3 Spring Security implementatsioon

Rakenduse autentimise ja autoriseerimise jaoks on kasutatud Spring Security raamistikku, mis on Springi põhistes rakendustes *de facto* standard. Joonis 12 kujutab autentimise ja autoriseerimisega seotud voogu.



Joonis 12. *Autentimise ja autoriseerimise voog*

Oma ressurssidele ligi saamiseks peavad kasutajad autentima tehes vastava päringu, mille päisesse on paigaldatud kliendi kasutajanimi ja parool. Neid võrreldakse neid andmebaasis asuvate andmete vastu ning sobivuse korral genereeritakse *token* ehk pikk sõne, mida klient saab edaspidi kasutada, et ennast rakendusele tuvastada. See *token* koos säilivusajaga salvestatakse andmebaasi ning ühtlasi lisatakse vastusele *AuthenticationToken* nimega küpsis, mille väärtuseks see määratakse. [45] Küpsisel on seatud tõeseks ka turvalisuse lipp, mis tähendab, et vastasolev klient saadab seda ainult krüpteeritud ühenduse puhul, et vältida selle lugemist kolmandate osapoolte poolt [46] ning *HttpOnly* lipp, mis ei luba ligipääsu kliendipoolsetel skriptidel, kaitstes murdskriptimise ehk XSS (*cross-site scripting*) rünnakute vastu. [47]

Rakenduse ressursid asuvad kõik aadressi "/api/**" taga ning kui mingi päring tehakse, käivitatakse autentimise ja autoriseerimise protsess. Kui tegemist on avalike lõpp-punktidega, lubatakse kasutajal ressurssidele ligipääs sõltumata *tokeni* olemasolust või valiidsusest. Autoriseerimist vajavate lõpppunktide ajal kontrollitakse, kas küpsis on olemas, selles leiduv *token* on õige ja kehtiv ning kas selle põhjal tuvastatud isiku roll vastab nõutele.

Turvalisuselemendina kasutatakse veel päringuvõltsingu ehk CSRF (*Cross-Site Request Forgery*) *tokenit*. CSRF on rünnaku tüüp, mille puhul ründaja meelitab autentitud isikut sooritama päringuid, mida nad ise ei plaaninud. Selle kaitseks tagarakendus genereerib ja

lisab päisesse eraldi *tokeni*, mida siis kasutatakse andmete muutvate päringute (enamlevinud POST ja PUT) puhul valeidentideedi alt tehtud päringute vastu. [48]

4. Rakenduse avalikustamine

Autori ülesanne oli lisaks tagarakenduse arendamisele luua kogu rakenduse (nii tagarakendus koos andmebaasiga kui ka esirakendus) jaoks pilves asuvad test- ja avalikud keskkonnad ning defineerima nendega seotud pideva integreerimise ja pideva tarnimise protsessid.

Ülesande tegi keerulisemaks asjaolu, et esirakendus on tegelikult jagatud kolmeks eraldiseisvaks portaaliks, nii et iga keskkonna pilvelahendus peab siduma ja haldama 5 erinevat komponenti. Samuti ei ole autoril eriti varasemat kokku puudet pilvekeskkondade rajamisega.

Ei kliendil ega ka ettevõttel ei olnud pilveteenuse pakkujate või teenuste valikul rangeid nõudmisi. Sõltuvalt rakenduse olemusest, tulevikuplaanidest ning spetsialiseeritud isiku puudumisest otsis autor lahendust, mis võimalikult palju lihtsustaks autori jaoks konfigureerimise, ressursihalduse ja turvalisuse manageerimist, oleks lihtsasti laiendatav ja uuendatav, kui soovitakse arendustega jätkata, lisada uusi portaale või minna üle mikroteenuse arhitektuurile ning samal ajal oleks võimalikult veakindel ja mõistliku hinnaga ning võimaldaks limiteerida maasolekuaega.

Sellise laiendatava mikroteenuste arhitektuuri loomiseks on kõige hõlpsam iga teenus konteineriseerida ning need pilve paigutada ja koos tööle panna kasutades konteinerite orkestreerimiste teenust. [49] Üks tuntumaid selliseid platvorme on Kubernetes, kuid selle ülesseadmist ning haldamist peetakse suure õppimiskõverusega ja keeruliseks ning kulukaks ja vajab pigem spetsialiseeritud isikuid, keda ei olnud võimalik kaasata. [50]

4.1 Pilvelahendus Amazon Web Servicega

Autor uuris põgusalt teenusepakkujate erinevaid lahendusi ning lõpuks otsustati võtta põhiteenuseks kasutusele Amazon Web Service (AWS) Fargate. Amazon Web Service (AWS) on pilveteenuse pakkujate seas kindel turuliider. Ta pakub tohutu hulk teenuseid erinevateks olukordadeks ja nõueteks, nii et kõik vajalik on võimalik leida ühelt teenusepakkujalt, on ulatusliku dokumentatsiooniga ning hallatav nii käsu- kui ka kasutajaliidese põhiselt, hea jõudlusega ning turvaline., samuti pakub ta laialdaselt tasuta/soodustusega hüvesid esimese aasta kasutajatele. [51]

4.1.1 Amazon Fargate

Fargate on (kasutaja jaoks) *server-less* keskkond konteinerite jooksutamiseks, mis võtab arendajalt vajaduse omada, jooksutada ja hallata konteinerite infrastruktuuri elutsüklit ja turvalisust ning kalkuleerib ise konteineri jooksutamiseks vajaminevad ressursid. Tasu arvutamine käib kasutuse põhjal – see tähendab, et ei pea maksma ressursside eest, mida on liiast. [52] Fargate põhikomponendid on järgmised:

- Klaster - loogiline grupeering tegumitest ja teenustest, mis isoleerib rakendust ning mille ressursse manageeritakse, et juhtida tegumeid.
- Tegumi (*task*) definitsioon - juhend, mille põhjal tegumeid hakatakse looma. Defineerida saab näiteks alloleva operatsioonisüsteemi parameetrid, avatud pordid, alloleva Docker kujutise (mis võetakse näiteks Amazoni konteinerregistrist), andmevolüümid ja keskkonnamuutujad.
- Tegum - klastris jooksev tegumi definitsiooni implementatsioon. Ühe ja sama definitsiooni põhjal võib korraga hoida töös mitmeid tegumeid.
- Teenus – jooksutab ja haldab soovitud arv tegumeid, luues ja peatades neid tegumeid definitsiooni põhjal vastavalt vajadusele. Kui näiteks üks tegum peatub, käivitakse ning asendatakse see uuega.

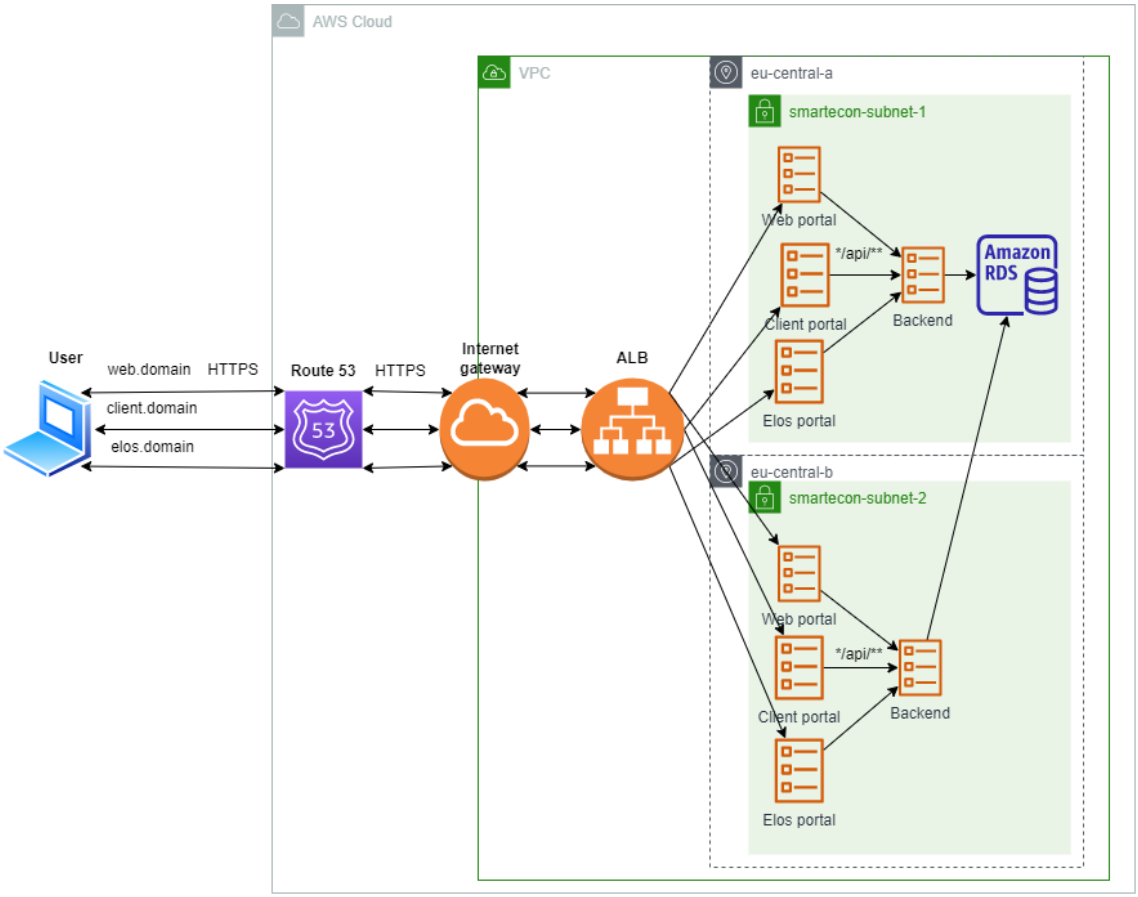
4.1.2 Võrgu implementatsioon Application Load Balanceriga

Implementeeritud võrku on kujutatud joonisel 13. Klastri jaoks defineeritakse virtuaalne privaatne pilv (*Virtual Private Cloud* ehk VPC), mis võimaldab kontrollida ressursside asetust, ühenduvust ja turvalisust saavadustsoonide, alamvõrkude, marsuutimistabelitega. Selleks, et VPC komponendid oleks välismaailmale kättesaadavad, tuleb defineerida lüüs (*internet gateway*).

VPC siseselt suunab rakenduse koormusjaotur (*Application Load Balancer*) kliendilt tehtud päringuid erinevate komponentide vahel URLi põhiselt kasutades reegleid (prioriteetsus on ülevalt alla):

- Kui URLis leidub muster /portal/**, siis suunatakse päring tagarakenduse sihtgrupile
- Kui URL vastab elos.domain/**, siis suunatakse päring Elose töötaja portaali sihtgrupile
- Kui URL vastab client.domain/**, siis suunatakse päring kliendiportaali sihtgrupile
- Kui URL vastab web.domain/**, siis suunatakse päring avalikule portaali sihtgrupile

Sihtgrupile suunamine tähendab seda, et ühes sihtgrupis võib joosta mitu tegumit näiteks tagarakendusest, mille vahel koormusjaotur saab koormust jaotada. Samuti saab koormusjaotur suunata vastavalt saadavusele päringu teisele saadavustsoonile. Lubatud liiklusvoogu saab dikteerida turvalisusgruppide põhiste reeglitega (näiteks on meil seatud reegel, et test-keskkonnale pääseb ligi vaid Net Groupi ja Smarteconi IP-aadressitelt). [53]



Joonis 13. *Fargate võrguarhitektuur*

4.1.3 Amazon Relational Database Service

Pilveandmebaasi jaoks kasutab autor Amazon Relational Database Service't. See on teenus, mis võimaldab paljusid andmebaasi instantse pilvekeskkonda üles seada erinevate andmebaasihaldusüsteemidega (sealhulgas PostgreSQL) ning pakub põhjalikke võimalusi andmebaasi automaatseks manageerimiseks ja hooldamiseks, sealhulgas erinevatesse tsoonidesse paigaldamine, migratsioonid, monitoorimine, perioodiline varundamine, vajadusel andmebaasi ajahetke põhine taastamine ning ligipääsureeglid pakkudes niimoodi turvalisust ja veakindlust ning vähendades arendaja poolset ajakulu infrastruktuuri ja süsteemi haldamiseks. Samuti on tegu üsna kuluefektiivse lahendusega, kuna tasu küsitakse ressursside pealt, mida reaalselt kasutatakse. [54]

Loodud andmebaasi instants on reeglitega privaatseks muudetud, mis võtab vastu ja vastab ainult päringutele, mis on tehtud VPC siseselt tagarakenduse poolt või Net Groupi IP-aadressilt (jälgimiste põhjustel) pordilt 5432.

4.1.4 Route 53 ja Certificate Manager

Route 53 on Amazoni pakutav domeeninimede süsteemi ehk DNS (*Domain Name System*) teenus, mis võimaldab nimeserverite abil tõlkida domeeninime vastavale IP-aadressile või teisele domeenile (antud lahenduse puhul koormusjaoturile määratud keerulisele domeeninimele). [55]

Lisaks on võimalik rakendada palju muud funktsionaalsust nagu domeeni registreerimine, tulemüüride seadistamine ja liiklusvoo haldus, kuid need on kaetud teiste teenustega ja seetõttu neid vaja ei olnud.

Selleks, et ühenduvus toimuks üle HTTPS protokolli, mitte krüpteerimata HTTP protokolli, oli olemasolevate domeenide jaoks vaja registreerida transpordikihi turbe ehk TLS (*Transport Layer Security*) sertifikaadid. [56] Amazon Certificate Manager pakub selleks lihtsat viisi nii põhidomeeni kui ka ühe taseme alamdomeenide jaoks ning sertfikaadi lihtsat ühildamist Route53 majutustsooniga. [57] Samuti on koormusjaoturile loodud suunamisreegel, et kui päring tuleb pordilt 80 (HTTP), suunatakse see ümber pordile 443 (HTTPS), kus suunatakse see edasi vastavale komponendile.

4.1.5 Monitoorimine Amazon CloudWatch, Lambda ja Teams Webhooki abil

Kasutatud Fargate ja Relational Database Service teenustega ühildub hästi Amazon CloudWatch, mis logib süsteemi sõnumeid, mida on võimalik näiteks rakenduse ebakorrektse toimimise korral sirvida, et avastada vigasid. Samuti monitoorib see nii koormusjaoturi kui ka rakenduse komponente paljude mõõdikute põhjal. Koormusjaoturi puhul on informatiivseks näiteks päringute hulgad staatuskoodide põhjal ja tema poolt tehtavad tervisekontrolli päringud, rakenduse komponentide ja andmebaasi puhul näiteks koormus ja mälukasutus. [58]

Ühtlasi on autor seadistanud mõõdikute põhjal teavitustesüsteemi. CloudWatch võimaldab defineerida mõõdikute põhjal alarmid (vaata joonis 14), mis tõstetakse, kui mõni mõõdik ületab lubatud väärtuse (näiteks hakkab andmebaasi maht täis saama). Samuti on konfigureeritud kuulaja logifailidele, mis veateatega logi korral alarmi tõstab. Nende juhtumite

korral kasutatakse Simple Notification Service (SNS) ja Lambda (võimaldab jooksutada erinevates keeltes kirjutatud skripte vastavalt defineeritud sündmustele) teenuseid ning Teams Webhooki, et saata automaatselt informeeriv sõnum autorile Teamsi vestlusesse (vt Lisa 7), et ta ilma viiviseta teaks, kui süsteemiga on probleemid võimingi komponent ei tööta nagu peab. [59, 60, 61]

☐	TEST elos-web-unhealthy-tasks-eu-central-1b	OK	2023-03-24 14:27:33	UnHealthyHostCount > 0 for 1 datapoints within 1 minute	Actions enabled
☐	TEST elos-elos-unhealthy-tasks-eu-central-1b	Insufficient data	2023-03-24 14:27:15	UnHealthyHostCount > 0 for 1 datapoints within 1 minute	Actions enabled
☐	TEST elos-client-unhealthy-tasks-eu-central-1b	Insufficient data	2023-03-24 14:23:30	UnHealthyHostCount > 0 for 1 datapoints within 1 minute	Actions enabled
☐	TEST elos-backend-unhealthy-tasks-eu-central-1b	Insufficient data	2023-03-24 14:21:31	UnHealthyHostCount > 0 for 1 datapoints within 1 minute	Actions enabled
☐	TEST elos-elos-unhealthy-tasks-eu-central-1a	OK	2023-03-24 14:19:35	UnHealthyHostCount > 0 for 1 datapoints within 1 minute	Actions enabled
☐	TEST elos-web-unhealthy-tasks-eu-central-1a	Insufficient data	2023-03-24 14:19:35	UnHealthyHostCount > 0 for 1 datapoints within 1 minute	Actions enabled
☐	TEST elos-client-unhealthy-tasks-eu-central-1a	OK	2023-03-24 14:16:54	UnHealthyHostCount > 0 for 1 datapoints within 1 minute	Actions enabled
☐	TEST elos-backend-unhealthy-tasks-eu-central-1a	OK	2023-03-24 14:08:06	UnHealthyHostCount > 0 for 1 datapoints within 1 minute	Actions enabled
☐	TEST elos-elos-cpu	OK	2023-03-24 13:54:25	CPUUtilization > 85 for 1 datapoints within 5 minutes	Actions enabled
☐	TEST elos-elos-ram	OK	2023-03-24 13:51:36	MemoryUtilization > 85 for 1 datapoints within 5 minutes	Actions enabled
☐	TEST elos-client-ram	OK	2023-03-24 13:49:19	MemoryUtilization > 85 for 1 datapoints within 5 minutes	Actions enabled
☐	TEST elos-client-cpu	OK	2023-03-24 13:47:42	CPUUtilization > 85 for 1 datapoints within 5 minutes	Actions enabled
☐	TEST elos-web-cpu	OK	2023-03-24 13:37:35	CPUUtilization > 85 for 1 datapoints within 5 minutes	Actions enabled
☐	TEST elos-web-ram	OK	2023-03-24 13:36:39	MemoryUtilization > 85 for 1 datapoints within 5 minutes	Actions enabled
☐	TEST elos-backend-ram	OK	2023-03-24 13:34:02	MemoryUtilization > 85 for 1 datapoints within 5 minutes	Actions enabled
☐	TEST elos-backend-cpu	OK	2023-03-24 13:17:39	CPUUtilization > 80 for 1 datapoints within 5 minutes	Actions enabled
☐	TEST elos-db-cpu	OK	2023-03-24 13:14:10	CPUUtilization > 80 for 1 datapoints within 5 minutes	Actions enabled
☐	TEST elos-db-ram	OK	2023-03-24 11:34:18	FreeableMemory < 30000000 for 1 datapoints within 1 minute	Actions enabled
☐	TEST elos-db-storage	OK	2023-03-24 11:08:49	FreeStorageSpace <= 5000000000 for 1 datapoints within 5 minutes	Actions enabled

Joonis 14. *Pilvekeskkonna alarmid*

4.1.6 Elastic Container Registry ja Docker

Nii tagarakenduse kui ka esirakenduste kujutiste ehitamiseks Fargate jaoks kui ka lokaalseks testimiseks konteineritena kasutab autor Dockerit. Docker on laialdaselt tuntud avaliku lähtekoodiga tarkvara, mis võimaldab rakenduse lähtekoodi jooksutada isoleeritud keskkonnas (konteineris). [62]

Tagarakenduse puhul kasutab autor veel abistavat pluginat Gradle Jibi, mis optimiseerib Dockeri kujutiste ehitamist, jagades rakenduse kihtideks ning eraldades rakenduse enda sõltuvused, mis tähendab, et kui arendaja ei lisa ega muuda sõltuvusi, ei peata neid uuesti ehitama. Samuti ei ole Jibi kasutades vaja Docker deemonit ning Dockerfile'i. [63]

Neid ehitatud kujutisi talletatakse Amazon Elastic Container Registry repositooriumites, kus Fargate pääseb hõlpsalt ligi ning saab tegumi definitsioonis defineeritud kujutise versiooni põhiselt käivitada Fargate tegumi. [64]

4.1.7 Amazon Parameter Store

Jooksvad tegumid vajavad oma tööks erinevaid parameetreid, millest osa on muutlikud ja/või salajased (paroolid, krüpteerimsvõtmed, andmebaasi ühendusparameetrid), mille puhul ei ole hea tava neid otse lähtekoodi panna või avaliku keskkonna muutujana hoida. Selliseid andmeid hoitakse (krüpteerituna) eraldi Parameter Store'is, kus neid on võimalik tegumi käivitamise hetkel pärida. Samuti hoitakse seal eelnimetatud Teamsi WebHooki ühendusparameetreid. [65]

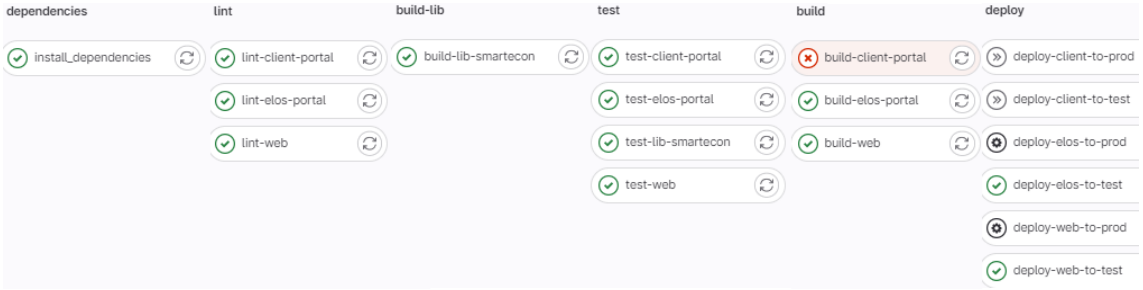
4.2 Rakenduse pidev integreerimine ja pidev tarnimine

Selleks, et veenduda, et muudatused koodibaasis ei ajaks rakenduse funktsionaalsust katki ning need muudatused jõuaksid ka pilves asuvatesse keskkondadesse ilma arendaja pidevate manuaalsete käskude sisestamistega, oli vaja implementeerida nii taga- kui ka esirakenduste jaoks pideva integreerimise ja pideva tarnimise protsessid. [66]

Autor kasutab selleks GitLabi CI tööriista, mis võimaldab .gitlab-ci.yml konfiguratsioonifailis defineerida ära konveieri. Konveier jaguneb etappideks (*stages*), millesse kuuluvad tööd (*jobs*), milles on omakorda kirjeldatud ära vastavad käsud. Seda konveierit täidab Gitlab Runneri rakendus, mis on paigaldatud Net Groupi enda serverisse. [67]

4.2.1 Esirakenduse konveier

Hetkel koosneb esirakenduse täiemahuline konveier kuuest etapist (vt joonis 15).



Joonis 15. *Esirakenduse konveier*

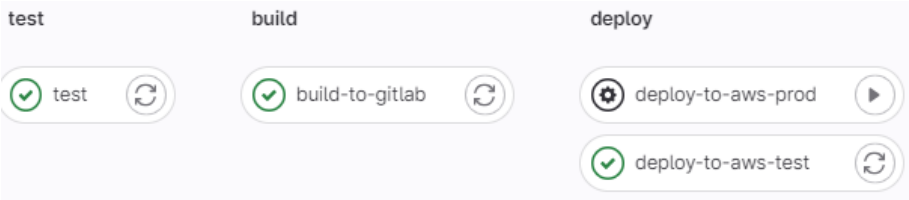
1. *dependencies* – installeerib vajalikud sõltuvused ning salvestab need artifaktina järgmiste tööde tarbeks.
2. *lint* – teostab koodistiili kontrolli kõigile kolmele portaalile. Kui tuvastab vea, siis töö ebaõnnestub.
3. *build-lib* – ehitab valmis korduvkasutatavate komponentide teegi, mida kõik portaalid kasutavad.
4. *test* – jooksutab eraldi nii teegi kui kõigi kolme portaali automaatteste. Töö ebaõnnestub, kui mõni test ebaõnnestub.
5. *build* – ehitab kõikide portaalide artifaktifailid (jooksutatav rakenduse kood)
6. *deploy* – jõustab rakenduse muudatused pilvekeskkondadesse
    - Laeb alla ja käivitab AWS, Dockeri ja jq käsuprogrammid, mis võimaldavad vajalikke käske jooksutada.
    - Autendib AWS portaaliga kasutades GitLabi CI muutujatesse salvestatud logimisparameetreid, et need ei oleks avalikult näha
    - Ehitab eelneva töö tulemusel saadud artifaktide põhjal Dockeri kujutised ning laeb need üles vastavatesse Elastic Container Registry repositooriumitesse.
    - Annab vastavale test-keskkonna Fargate teenusele teada, et too käivitaks uue tegumi uue versiooni põhjal ning selle õnnestumisel peatab aegunud lähtekoodil põhineva tegumi.
    - Pärast edukat uuenduste testimist test-keskkonnas on võimalik käsitsi algatada töö, mis sarnaselt käivitab uue versiooni rakendusest avalikus keskkonnas.

Kõige selle juures on üritatud antud konveierit optimeerida nii, et ehitatakse ja tarnitakse ainult neid portaale, mida viimati tehtud muudatused mõjutavad. See tähendab, et kui muudetud on ainult kliendiportaali mingit osa, mis ei mõjuta Elose portaali, siis Elose portaali ehitamise ja tarnimisega seotud töid ei käivitata. Samuti käivitatakse ehitamise ja tarnimise osa vaid siis, kui muudetakse põhiharu. Kui muudetakse mingit muud haru, siis

vaikimisi käivitatakse etapid ainult kuni testimiseni. Konveierist on näha ka, et portaalid on teineteisest sõltumatult tarnitavad, s.t antud olukorras kliendiportaali ehitamine ebaõnnestus ning seda pilvekeskkonda ei tarnitud, samal ajal kui teiste portaalide muudatused edukalt test-keskkonda nähtavale jõudsid.

4.2.2 Tagarakenduse konveier

Tagarakenduse konveier on tunduvalt lühem, koosnedes kolmest etapist, mida on näha joonisel 16.



Joonis 16. *Tagarakenduse konveier*

1. *test* – kasutab Gradle enda käsku, et kontrollida stiili, jooksutada teste ja ehitada rakendus üles.
2. *build* – kasutab Gradle jib pluginat, et ehitada Dockeri kujutis ja laadida see Gitlabi konteineri registrisse.
3. *deploy* - jõustab rakenduse muudatused pilvekeskkondadesse
   - Laeb alla AWS, Dockeri ja jq käsuprogrammid, mis võimaldavad vajalikke käske jooksutada.
   - Autendib AWS portaaliga kasutades GitLabi CI muutujatesse salvestatud logimisparameetreid, et need ei oleks avalikult näha
   - Laeb alla Gitlabi registrisse ehitatud Dockeri kujutise ning laeb selle üles Elastic Container Registry repositooriumisse.
   - Annab vastavale test-keskkonna Fargate teenusele teada, et too käivitaks uue tegumi uue versiooni põhjal ning selle õnnestumisel peatab aegunud lähtekoodil põhineva tegumi.
   - Pärast edukat uuenduste testimist test-keskkonnas on võimalik käsitsi algatada töö, mis sarnaselt käivitab uue versiooni rakendusest avalikus keskkonnas.

Samamoodi käivitatakse *build* ja *deploy* etapid vaid siis, kui muudatus toimub põhiharus, vastasel juhul käivitatakse ainult test etapp.

5. Tulemuste analüüs

Lõputöö skoobiks oli luua veebiplatvormi Elose minimaalse töötava toote tagarakendus, mis pidi pakkuma reaalset väärtust projekti elutsüklis, hankides potentsiaalseid kliente ning langetades projektijuhtide koormust. Smarteconiga arutades pidi see olema rakendus, mis suudaks:

- Tekitada eraklientides huvi, kogudes neilt arusaadavalt ja ajaefektiivselt kokku esmased vajalikud andmed ja talletades need edasisteks sammudeks
- Esitada neile automaatselt esmane hinnapakkumine potentsiaalsete klientide filtreerimiseks
- Genereerida automaatselt pakkumine, mille klient saab vastu võtta või ära öelda
- Genereerida allkirjastatud leping koos lisadega, mida klient saab digiallkirjastada ning üles laadida.
- Anda projektijuhtidele hea ülevaade projekti hetkeseisust ning võimaldades vajadusel sekkuda rakenduse voogu (muuta sisendandmeid, lisada täpsustusi, mõjutada kalkulatsioonitulemusi või lepingutingimusi, teha pakkumiste ebaõnnestumisel uusi pakkumisi)

Kuna päikesepaneelide projektid on kulukad ja keerulised ning ka rakenduse tulevikuvisioon on suur, siis pidi lisanõuetena mõistagi olema rakendus turvaline, veakindel ning hõlpsasti edasiarendatav. Samuti, kuna minimaalne töötav toode pidi minema reaalsele Leedu turule testimisele, oli autori ülesanne see ka koos esirakendusega avalikustada. Täpsemad nõuded rakenduse funktsionaalsusest, väljanägemisest, objektide spetsifikatsioonidest ja voogudest olid esialgu suhteliselt hägused kiiresti implenteeritud Figma prototüübi ja esmase lähteülesande dokumendi kujul.

5.1 Lõpptulemused

Antud lõputöö tulemuseks arendas autor iseseisvalt tagarakenduse, mis haldab vastavalt ülaltoodud põhieesmärkidele tellimuse esitamise ja kooskõlastamise, pakkumise genereerimise ja lepingute haldamise etappe. Projekti arenduse ja testimiste käigus kadusid osad algses lähtedokumendis ning prototüübis esitatud nõudmised ning kerkisid esile uued, täiendavad nõudmised. Lisaks, kuna esmaseks planeeritavaks sihtturuks sai Leedu, implementeeris autor ka tõlketoe veateadetele, teavitusemailidele, muudatuslogidele ja pakkumise PDF failile.

Alguses alustati vastavalt visioonile rakenduse voo implementeerimisega, mis võimaldaks kliendil läbida neid etappe valdavalt iseseisvalt, kuid arenduste, testimiste ja arutelude käigus leiti, et kohe alguses kulukate ja keeruliste projektide puhul sellist lähenemist kasutada võib olla liiga riskantne või veniks arenduste ja valideerimiste periood liiga pikaks. Selleks otsustati kiiremaks ja ohutumaks turule sisenemiseks esialgu ikkagi kaasata igas etapis projektijuhti ning hiljem pärast esimesi kogemusi ja testimisi arendada rakenduse funktsionaalsus selliseks, et see suudaks enamustel juhtudel piisavalt täpselt informatsiooni küsida, kalkuleerida, arvutada ja suunata. See aga tähendas osa voo ümber implementeerimist ning validatsioonireeglite muutmist.

Kuna selliselt implementeeritud voog nõuab rohkem kordamööda tegevusi, siis selleks, et kiirendada protsessi, nähti vajalikuks implementeerida e-maili teavituste süsteem, mis annaks pidevalt märku vastavale osapoolele, kes peaks järgmisena midagi tegema. Selleks, et tagada arvukate meilide mitmekeelsuse tugi ning esinduslikum välimus igas seadmes, nägi autor vajalikuks kasutada mallimootorit ja emailidele disanimisele spetsialiseeritud raamistikku.

Testimiste käigus kerkis esile probleem, et mitme inimese poolt projektiga seotud andmete muutmisel võib osa muudatusi jääda teisele osapoolele märkamatuks ning seetõttu oli vaja luua inimloetav muudatuslogide generaator, mis annab märku, kui sellel osapoolel on ülevaatamata muudatusi. Selleks, et see logide genereerimine oleks hallatav ka siis, kui objektid pidevalt arenduse käigus muutuvad, otsustas autor kasutusele võtta raamistiku, mis nende võrdlemist mugavdab. Samuti tekkis idee luua täpsustuste küsimiseks ja lisaselgituste jaoks kommentaarium, kus projekti kõik osapooled saavad jätta märkmeid.

Automaatsest lepingute genereerimisest ja digiallkirjastamisest esialgu loobuti, sest tegemist on tähtsate dokumentidega, mis võivad vajada ka mõnedel juhtudel lisaklausleid või muudatusi. Samuti kaotati voo muudatuste implementeerimisel nii ärilistel põhjustel kui ka segaduse vältimiseks kliendi võimekus autenditud portaalis omaalgatuslikult muuta esialgselt valitud tooteid ja sisendandmeid.

Veebruaris sündis idee pakkuda projekti väljaostu asemel ka rentimise võimalust, et suurendada potentsiaalsete klientide arvu nende arvelt, kellel ei ole võimalik koheselt suuri väljaminekuid teha. Tagarakenduses tähendas see lisakeerukust vastavalt maksmisviisile andmete kuvamise, talletamise, rendihindade kalkuleerimise ning rendiparameetrite, nagu rendiperiood ja intressimäär, haldamise näol.

Uuel aastal võeti vastu otsus teha ka rebränding - alguses oli loodud esialgne lihtsakoelisem prototüüp Smarteconi brändi all, kuid siis otsustati esimesena rakendus turule lasta hoopis

Leedus ja sellest tulenevalt siduda Elos täielikult Smarteconist lahti ning selle raames tehti ka algne prototüüp värskelt kaasatud analüütiku poolt drastiliselt ümber. Suuri muudatusi tähendas see esirakenduse arendaja ja maandumislehe kujundaja jaoks, kuid ka tagarakenduses oli vaja vastavalt uuele brändile ning prototüübile defineerida ringi osa funktsionaalsust (mis kujul ja millal mingeid päringuid tehakse), muuta pakkumiste ja emailide sisu ja kujundust ning hakata tõlkima tooteid, emaile ja veateateid leedu keelde. Selle tulemusel jäi esirakendus ka tagarakendusest voo ja funktsionaalsuste implementeerimisel maha.

Kuigi algselt oli autori vastutusaladeks tagarakenduse arendamine, analüüside teostamine ja dokumenteerimine, paluti tal novembris luua ka avalik test-keskkond, kus saaks läbi viia valideerimisi ja testimisi. Kuigi autoril on sellega olnud varem vähe kogemusi, tundus see põneva ja arendava väljakutsena ning tingimusel, et talle antakse piisavalt aega uurida ja analüüsida, ta nõustus. Alguses esines küll mõningaid komplikatsioone - näiteks otsis autor mõnda aega viga, miks tagarakendus peatatakse ning põhjuseks osutus, et see ei jõudnud nii kiiresti käivituda, kui oli seatud tervisekontrolli ebaõnnestumiste maksimaalne ajaline väärtus. Samuti ei leidnud alguses nii esi- kui ka tagarakendus pilvekeskkonnas üles enda ressursifaile ning blokeeriti ka emailide saatmine tagarakendusest. Hoolimata ebamugavustest oli detsembriks rakendus kogu senise funktsionaalsusega pilvekeskkonnas töötamas ning mugavuse ja veaohutuse eesmärgil otsustas ta implementeerida ka pideva tarnimise protsessid, et ei peaks iga muudatuse korral käsitsi uut versiooni test-keskkonda transportima. Kuna test-keskkond on töötanud sujuvalt, usaldati autorile ka avaliku keskkonna ülesseadmine, mida kasutavad reaalsed kliendid ja kus liiguvad pärisandmed ning mille ülesseadmine ja haldamine on ka seetõttu tunduvalt vastutusrikkam.

Tagarakendusest, pilvelahendusest ja arendusprotsessist on lõputöö kirjutamise abiga valminud ka ulatuslik ja põhjalik eesti keelne dokumentatsioon, mis tiimijuhi hinnangul sobib esitamiseks näiteks lisatööjõu kaasamisel, et anda ülevaade ärilisest taustast, rakenduse visioonist, kasutatud tehnoloogiatest ning implementatsioonist. Lõputöö dokumendi väliselt leidub juhendeid veelgi Postman päringute, pilvelahenduse konfiguratsioonide ja juhiste kujul.

5.2 Tulemuste valideerimine

Rakenduse funktsionaalsuse oodatud toimimist valideeris autor pidevalt automaat- ja manuaaltestimisega. Autor on kirjutanud juba praeguseks üle 600 automaattesti kombineerituna kitsa skoobiga üksustestidest, komponentide omavahelist koostööd testivatest osalistest integratsioonitestidest ning terveid kasutusjuhte testivatest suuremahulistest integratsioonitestidest. Nende abil sai autor enesekindlamalt ka uusi muudatusi sisse viia,

kartmata lõhkumata ära rakenduse tööd. Oluline koht oli muidugi ka manuaalsel testimisel, mille jaoks oli autor defineerinud üle 60 Postmani päringu, mis võimaldas läbida erinevaid keerulisemaid voogusid. Ulatuslikumaid manuaalse testimise sessioone tegi autor suuremate kasutajaloode valmimisel ning demode eel.

Koodibaasi arusaadavuse ja mõistlikkuse valideerimiseks tegi autorile koodiülevaatuseid esirakenduse arendaja, mille puhul sai ta vajadusel selgitusi küsida ja soovitusi jagada ning ülevaate ka enda jaoks esirakenduse implementeerimisel.

Samuti aitas varem üks arendaja Net Groupist teha rakendusele JMeteriga [68] koormusteste, tehes põhilisemaid päringuid 1000 korda. Tema kogemuse põhjal oli üldpilt hea ja erilisi anomaaliaid ei tuvastatud. Nõrgeim lüli massiliste päringute korral on hetkel kalkulatsioonide puhul kasutatav JRC API (enamus juhtudel suhteliselt märkamatu, halvematel juhtudel umbes 3 sekundit), kuid klient pidas seda talutavaks, kuna rakenduse olemuse poolest ei kalkuleerita neid päringuid väga tihti ümber ning ajalist hilinemist esines ka konkurentidel. Arutati alternatiivi luua andmebaasi tugi regioonide keskmistega, mille puhul oleksid päringud tunduvalt kiiremad, kuid klient eelistas pigem täpsust, mida nimetatud API pakub.

Üle kahe nädala toimusid juhtkonna koosolekud, kus olid Net Groupi ja Smarteconi esindajad nii Eestist kui Leedust ning kus tegeleti rakenduse üle vaatamise ja järgmiste nõudmiste ja ajaraamide seadmistega. Enamus täpsemaid ärilisi otsuseid sündis seal nende koosolekute käigus, näiteks vajadus põhjalike pakettide loogika järele, idee luua rentimisvõimalus, muudatuslogide genereerimine, kommentaarium, emaili teavitused, skänneerimise etapist loobumine ning rebränding. Samuti sai nendel koosolekutel küsida implementeeritud lahenduse kohta koheselt tagasisidet, näiteks Leedu esindajalt pakkumisfaili sisu ja disani kohta või JRC API pakutud tulemuste valideerimine Smarteconi enda andmestikuga.

Seatud töömetoodika põhiselt osales igapäevastel kohtumistel Elose tooteomanik, kes on varasemalt pikaaegselt olnud Smarteconi müügiisik ja väljastanud suurel hulgal projekte ning seega oskas pakkuda palju sisendit ja valideerimist funktsionaalsuste ja voogude, toodete spetsifikatsioonide, kalkulatsioonide, pakkumise koostamise, sisu ja teiste teemade kohta ning prioritiseeris arendusülesandeid. Analüütikuga koos testisid nad ka täpsemalt vooge läbi, mõeldes, mida saaks paremaks teha. Nende käigus tuli väga palju tagasisidet esirakenduse arendajal visuaalse väljanägemise kohta, millest mõned vajasid ka tagarakenduse tuge andmete serveerimise näol.

Suuremate arendustükkide valmimisel tehti ka suuremaid kasutuskogemuse testimisi. Es-

mase versiooniga rakendusest käidi Smarteconi kontoris kohapeal testimas, lastes mitmetel projektijuhtidel ja müügiisikutel proovida, tutvustades tulevast tööriista ning küsides tagasisidet. Tegemist oli algelisema versiooniga, kuid juba siis oldi entusiastlikud ning põnevil ja tunti, et täiustamisel teeks selline tööriist nende tööd tunduvalt efektiivsemaks. Sealt tuli ka esmane sisend defineerimast ümber rakenduse voogu, et millal midagi teha saab ja millal midagi kujutatud on. Samuti tuli ettepanek loobuda kliendil omaalgatuslikult sisendandmete muutmisest pärast esmast registreerimist ning valideerimiseks ja testimiseks implementeerida esimene versioon voost siiski suurema projektijuhtide osalusega ning hiljem hakata liikuma üle iseteenindusplatvormi voogudele.

Klientide rahulolu valideerimiseks küsib Net Group kõikidelt klientidelt korra kvartalis *Net Promoter Score* hinnangut, mille raames klient annab hinnangu 10-palli süsteemis, kui rahul nad pakutava teenusega on ning kui tõenäoliselt nad soovitaksid ettevõtet teistele. Smartecon andis arendustiimile tulemuseks 9 palli 10-st, mis viitab tõlgenduse järgi kõrgele rahulolule. [69]

AWS pilvekeskkonnas olev test-keskkond on olnud üleval detsembrist saati ning selle aja jooksul on rakendust pidevalt kasutatud ja monitooritud, mille puhul pole ilmnenud ühegi rakenduse komponendi puhul iseenesliku peatumist või häireid, ühendusprobleeme ega rakenduse kasutamise ja testimise süsteemitõrkeid, et ressursid ei oleks saadaval, rakendus oleks liigselt aeglane, ei läheks emailid läbi või muud. Samuti hoiti silma peal kuludel, sest kasutusele võetud tehnoloogiate puhul sõltub see ressursside kasutusest. Test-keskkonna täislahenduse keskmine kuu maksumus kogu perioodi lõikes on olnud 48,27 eurot. Samuti on uute arenduste käigus Gitlabi konveierite abil kandunud kõik muudatused õigesti test-keskkonda. Käsitsi on meelega kutsutud esile häireid süsteemi töös, et kontrollida, kas tõrgete korral annab implementeeritud monitooring autorile teada tõrget esinemisest, mis samuti töötas edukalt.

21. aprillil toimus Tartus Maamess [70], kus oli väljas Smarteconi tiim ning kus otsustasime lasta põhjalikuma tagasiside saamiseks reaalsete projektidega testida. Selleks sõitsid tiimijuht ja esirakenduse arendaja Tartusse ning autor jäi kontorisse valvesse, et ta saaks keskkonda ja andmebaasi monitoorida ning probleemide korral kiireid parandusi teha. Senine tagarakendus ja pilvelahendus alt ei vedanud - kõik info serveeriti ja talletati korrektselt, loodi kasutajad ning vajalikud emailid jõudsid pärale mõlematele osapooltele, ühtlasi kuvati projektide infot kenasti portaalis, kus seda ka muuta sai. Ainuke probleem oli veidi paigast nihkunud disain ühes email-klientrakenduses, kuid see sai kiiresti likvideeritud. Ühtlasi on vaja veel kalibreerida kalkulatsioonimootorit, kuid see on hetkel Smarteconi laual ning autor on ärilise sisendi ootel, et vastavalt vajadusele kas arvutuskäikude, koefitsientide või hindade muudatused sisse kanda.

Kasutajaliidese kohapealt anti esirakenduse arendajale siiski päris palju tagasisidet. Kõige läbitungivam probleem oli Google Maps API implementatsiooni kasutamine paigaldusala polügoni joonistamiseks, mis valmistas paljudele probleeme, samuti pakuti välja palju ideid ja tagasisidet disanimuudatuste asjus.

Ühest küljest oli rebrändingu tõttu esirakenduse arendaja autenditud portaali kohapealt juba eelnevalt tempost maas ja tagasiside põhjal tuleb tal ka esimeses vormis osa funktsionaalsust, vaateid ja Google Mapsi lahendust parandada. Teisalt on Smarteconil endal veel vaja veidi aega viimase sisendi andmiseks, domeeni kättesaamiseks ja ärilise poole üles seadmiseks. Seetõttu nihkub portaali avalikustamine ka edasi juuni teise poolde, enne mida toimub ilmselt veel üks põhjalikum kasutuskogemuse testimine paranduste kontrollimiseks.

5.3 Tiimi hinnang autorile

Autori tiimijuht, kes veab eest Smarteconi projekti, usub, et noore arendajana on ta saanud enda tööülesannetega väga hästi hakkama, andes ka arenguvestluse käigus teostatava tulemuslikkuse ülevaates autorile 4-palli skaalas 4 ehk "üle ootuste". Autori tugevusteks toob ta välja kiire kohanemisvõime, avatuse uutele väljakutsetele ning püüdlikkuse, mille tõttu ei pelganud ta arendajale usaldada ka päris laia vastutusala: ambitsioonika projekti iseseisev tagarakenduse arendamine ning testimine, skaleeritav pilvelahendus, kohati analüütiku rolli täitmine kliendinõuete kirjeldamisel ning äriliste lahenduste väljamõtlemise toetamisel, arendustööde planeerimine ja ülesannete kirjeldamine, projekti dokumenteerimine ning aegajalt väiksemate ülesannetega esirakenduse arenduse toetamine. Ta lisab, et autor on hoolimata muutlikest nõuetest ka ilusti tähtaegades püsinud.

Vanemanalüütik, kes aitas kriitilisemaid nõudeid ja voogusid defineerida ning rakenduse prototüüpi luua, kiidab, kuidas autor läheneb ka ise probleemidele analüütiliselt ning mõtleb läbi erinevaid lahendusi ja potentsiaalseid probleeme ka tuleviku vaatest, andes seejuures ka äriliselt kasulikku sisendit. Tema teostas ka esmaseid kvaliteedikontrolle ning tunnustab autori põhjalikkust ja järjekindlust, märkides, et temani ei ole jõudnud kriitilisi vigasid, ainult mõned väiksema kaaluga vead nagu ülearuse lehekülje printimine PDF failides.

Autori tiimikaaslane, kes arendas esirakendust ning kellega käis tihe koostöö arendusplaanide ja funktsionaalsuste koordineerimisel ja kes teostas ka koodiülevaatuseid, kiidab autori hoolikust ja läbimõtlemise oskust. Koodiülevaatuseid teostades valmistas talle muljet arvukad testid ning tõi välja, et autor nägi uute funktsionaalsuste arendamisel palju vaeva ka olemasoleva koodi ja andmebaasi skeemi refaktoreerimisel.

5.4 Autori hinnang projektile ja arendustöödele

Projekti vaatest tunneb autor, et tegu on olnud väga mitmekülgselt arendava projektiga. Võimalus seada üles projekt täiesti nullist, disainida arhitektuuri ja andmebaasi, uurida ja töötada paljudeerinevate modernsete tehnoloogiatega, proovida erinevaid lahendusviise ning suhelda pärisklientidega on pakkunud väga laiapõhjalist kogemust, aidanud mõista suurt pilti ning rajanud hea vundamendi teiste projektide jaoks.

Peamine asi, millest autor projekti puhul puudust tunneb, on suuremas tiimis töötamine. Autor on pidanud koostööd tegema tiimijuhi, esirakenduse arendaja ja perioodiliti analüütikuga ning pidevalt suhtlema ja valideerima kliendiga, kuid pole selle projekti raames eriti tõelist tiimiga arendamise kogemust saanud, mis seisneb teiste koodi analüüsimises ning sellega töötamises, ranges tööprotokollis ja koodiülevaatustes. Samuti oleks mõne kogenud seeniorarendajaga koos arendades võibolla saanud paremini arutada mingisuguseid lahendusi, saada põhjalikumaid koodiülevaatuseid, näinud teistsuguseid arenduspraktikaid ning säästnud aega mõnede teemade puhul, näiteks konfiguratsioonivead (mõneks näiteks keskkonnaprofiilid, emailide saatmine, konveieris autentimine), mille tuvastamiste ja lahendamistega läks autoril teinekord oodatust kauem aega.

Lõpptulemuse puhul usub autor, et loodud lahendus on läbimõeldult, paindlikult ja arusaadavalt kirjutatud ka edasiarendusteks. Ta tunneb, et loodud arhitektuur ning ka valitud tehnoloogiad täidavad oma eesmärki. Üks asi, mis siiski vajab tema silmis ringi tegemist, on failide kandmine tagarakenduse andmebaasist eraldi faililattu ning salvestada rakenduse enda andmebaasis vaid viiteid failide asukoha kohta. Siiani on kõiki faile koos sisuga hoitud *attachment* tabelis baidijadana, mis ei ole üldiselt soovitatav, kuna nii suuremahulised andmed koormavad andmebaasi jakulutab väärtuslikku ressurssi, aeglustab päringuid ning muudab andmebaasi hooldamise ja varundamise kulukaks operatsiooniks. [71] Selle jaoks on autor uurinud Amazoni S3 faililao kohta, mis on odav, turvaline ja paljude lisafunktsionaalsustega ning kus failide organiseerimine toimub mugavalt ämbrite põhiselt, millele turvalist programmaatilist ligipääsu Javaga on üsna lihtne implementeerida ning kus neid saab pärida faili nime põhjal. Tegelikult on autor selle juba kasutusele võtnud, lisades sinna toodete pildid, kuid pole veel jõudnud ülejäänud failide loogikat sellega siduda. [72]

Arendusprotsessi puhul tunneb autor, et oleks võibolla sellise *start-up* lähenemisega projekti puhul saanud toimida ajaliselt efektiivsemalt, kui kasutada algusfaasis inverteeritud testimise püramiidi põhimõtteid, mille puhul vähendatakse rõhku arvukate üksustestide kirjutamise arvelt ning eelistatakse pigem mahukaid integratsiooniteste, mis testivad rakenduse funktsionaalsust üleüldiselt ning sõltuvad vähem sellest, milline on sisemine implementatsioon. [73] Niimoodi jõutakse kiiremini esimeste voogude valideerimisteni

ning koodi ja funktsionaalsust on vajadusel ka kergem ümber kirjutada. Seejärel, kui visioon ja nõuded hakkavad kinnistuma, saab üle liikuda üksustestidele, mis on kiiremad ja täpsemad ning millega saab kriitilistes kohtades rohkem erinevaid olukordi testida.

5.5 Jätkuarendused

Prioriteetseim ülesanne on rakenduse hetkesele voole juurde implementeerida paigaldusfaas, et rakendus kataks ja toetaks projekti kliendi esmasest päringust kuni projekti üleandmiseni. Selleks on vaja täiendada nii kliendi kui ka projektijuhi voogu paigaldusetappidega seotud funktsionaalsusega kui ka luua eraldiseisev paigalduspartnerite portaal.

Järgmisena oleks vaja Elose poolelt luua lisaks projektijuhtidele ka administraatori portaal, kes oleks nii öelda superkasutaja. Tema ülesanne on hallata kliente, projektijuhte, partnereid, tooteid ja portaali andmestikku (lisasektsiooni kategooriaid). Samuti on temale ettenähtud funktsionaalsus projektide käekäigu ja statistikate ülevaatamiseks, et koordineerida ja parandada rakenduse tööd.

Pärast seda, kui projekti täielik voog on valmis ning seda on turul testitud ja piisavalt sisendeid ja kogemust saadud, on vaja hakata olemasolevaid protsesse ja funktsionaalsuseid täiendama, et liikuda lõppvisiooni suunas, mis muudaks rakenduse piisavalt kindlaks ja võimekaks, et see suudaks suunata ja koordineerida kliente ja partnereid nii, et projektijuhid oleks pigem ülevaataja rollis ja sekkuksid juhendamise või täiendamiste näol vaid vajadusel.

Kui rakendus on piisavalt võimekas, et edukalt vastavalt seatud eesmärkidele teenindada Leedu kliente, on vaja hakata liikuma ülejäänud sihtturu poole. Senise funktsionaalsuse implementeerimisel on seda arvestatud, kuid tegemist on keerulise ülesandega ja vajab palju tööd: vaja on luua tõlketabeleid, rakendada regioonipõhist loogikat (erinevad tooted, kalkulatsiooniparameetrid, projektide tingimused, asukohapõhine projektijuhtide ja paigalduspartnerite määramine) ning arvestada ka pilvelahenduse juures serverite asukoha, andmemahtude ja süsteemile avalduvate koormuste näol.

6. Kokkuvõte

Antud töö eesmärgiks oli arendada valmis päikeseparkide teenindusplatvormi tagarakenduse minimaalse elujõulise toote versioon kliendile Smartecon ning seejärel avalikustada see koos esirakendustega Leedu turu jaoks pilveteenuste abil. Töös seletatakse lahti hetkeolukord ja tulevikuvisioon, uuritakse konkurentide olemasolevaid lahendusi, sõnastatakse rakenduse funktsionaalsed nõuded ning esitletakse implementeeritud tehnoloogilist- ja pilvelahendust. Lõpus analüüsitakse järelevaatavalt tehtud tööd ja protsessi ning seatakse plaan edasisteks arendusteks.

Valminud rakendus on alles esimene versioon ning selle edasiarendused jätkuvad, et luua võimas teenindusplatvorm, mis suudab erakliente suunata ajaefektiivselt ja poolautomaatselt läbi keerulise projekti esimesest tellimuspäringust kuni paigaldatud projekti üleandmiseni, vähendades seeläbi oluliselt projektijuhtide tööd.

Kasutatud kirjandus

[1] *Solar power market size share*. [Loetud: 10-04-2023]. URL: https://www.fortunebusinessinsights.com/industry-reports/solar-power-market-100764.

[2] *Metsolar*. [Loetud: 10-04-2023]. URL: https://metsolar.eu/.

[3] *Solitek*. [Loetud: 10-04-2023]. URL: https://www.solitek.lt/lt.

[4] *Svea Solar*. [Loetud: 10-04-2023]. URL: https://sveasolar.de/de-de.

[5] *Solarwatt*. [Loetud: 10-04-2023]. URL: https://www.solarwatt.de/.

[6] *Figma*. [Loetud: 17-04-2023]. URL: https://www.figma.com/ui-design-tool/.

[7] *What's an AsiCe file: How to Open and Sign AsiCe Documents?* [Loetud: 28-03-2023]. URL: https://marksign.eu/blog/how-to-open-and-sign-an-asice-file/.

[8] *9 best reasons to choose Java for web development*. [Loetud: 18-04-2023]. URL: https://www.xicom.biz/blog/9-best-reasons-to-choose-java-for-web-development/.

[9] *Why Spring?* [Loetud: 25-03-2023]. URL: https://spring.io/why-spring.

[10] *Spring Projects*. [Loetud: 25-03-2023]. URL: https://spring.io/projects.

[11] *Gradle Vs Maven: What's The Difference?* [Loetud: 15-08-2022]. URL: https://www.interviewbit.com/blog/gradle-vs-maven/.

[12] *Gradle vs Maven: Performance Comparison*. [Loetud: 15-08-2022]. URL: https://gradle.org/gradle-vs-maven-performance/.

[13] Laxmi Sharma. *Why Choose PostgreSQL for your Next Application in 2023?* [Loetud: 15-08-2022]. URL: https://kanakinfosystems.com/blog/why-choose-postgresql?fbclid=IwAR1WKO2AUNn4uXzeR191tjFYWESyyqcwS1gjCOmmT9f-tPmcp_puQAanc.

[14] Amit Phaujdar. *PostgreSQL vs Oracle: 6 Critical Differences*. [Loetud: 15-08-2022]. URL: https://hevodata.com/learn/postgresql-vs-oracle/.

[15] Salman Ravoof. *PostgreSQL vs MySQL: Explore Their 12 Critical Differences*. [Loetud: 03-04-2023]. URL: https://kinsta.com/blog/postgresql-vs-mysql/.

[16] *H2 Database Engine*. [Loetud: 15-08-2022]. URL: https://www.h2database.com/html/main.html.

[17] Akshit Kumar. *10 Reasons Why Liquibase is the Best Tool for Database Migrations*. [Loetud: 16-08-2022. URL: https://blog.knoldus.com/10-reasons-why-liquibase-is-the-best-tool-for-database-migrations/.

[18] *Liquibase vs. Flyway*. [Loetud: 16-08-2022]. URL: https://www.liquibase.com/liquibase-vs-flyway.

[19] Raja Anbazhagan. *FreeMarker vs Groovy vs Mustache vs Thymeleaf*. [Loetud: 03-10-2022]. URL: https://springhow.com/spring-boot-template-engines-comparison/.

[20] *Thymeleaf*. [Loetud: 03-10-2022]. URL: https://www.thymeleaf.org/.

[21] *JavaMail*. [Loetud: 01-10-2022]. URL: https://javaee.github.io/javamail/.

[22] Douglas Karr. *Understanding The Challenges (And Frustrations) of HTML Email Design*. [Loetud: 09-04-2023]. URL: https://martech.zone/understanding-the-challenges-and-frustrations-of-html-email-design/?utm_content=cmp-true.

[23] *MJML - The responsive email framework*. [Loetud: 11-04-2023]. URL: https://mjml.io/.

[24] *iText*. [Loetud: 27-10-2022]. URL: https://itextpdf.com/.

[25] *Apache PDFBox*. [Loetud: 27-10-2022]. URL: https://pdfbox.apache.org/.

[26] *DigiDoc4j*. [Loetud: 28-03-2023]. URL: https://github.com/open-eid/digidoc4j.

[27] *Envers*. [Loetud: 21-01-2023]. URL: https://hibernate.org/orm/envers/.

[28] *Javers*. [Loetud: 21-01-2023]. URL: https://javers.org/.

[29] *Solar Irradiance Energy Prediction API*. [Loetud: 14-02-2023]. URL: https://openweathermap.org/api/solar-energy-prediction.

[30] *Photovoltaic Geographical Information System*. [Loetud: 15-02-2023]. URL: https://re.jrc.ec.europa.eu/pvg_tools/en/tools.html.

[31] Lowie Cuypers. *Switching from RestTemplate to WebClient: A reactive tale*. [Loetud: 16-02-2023]. URL: https://ordina-jworks.github.io/rest/2020/10/12/RestTemplate-vs-WebClient.html.

[32] *Jasypt - Java simplified encryption*. [Loetud: 19-09-2022]. URL: http://www.jasypt.org/features.html.

[33] *Passay - Password policy for Java*. [Loetud: 27-09-2022]. URL: https://www.passay.org/.

[34] Nicolas Fränkel. *A list of cache providers*. [Loetud: 02-11-2022]. URL: https://blog.frankel.ch/choose-cache/2/.

[35] Ben Manes. *Caffeine cache*. [Loetud: 02-11-2022]. URL: https://github.com/ben-manes/caffeine.

[36] *Project Lombok*. [Loetud: 15-08-2022]. URL: https://projectlombok.org/.

[37] *Spotless*. [Loetud: 16-08-2022]. URL: https://dev.to/ankityadav33/standardize-code-formatting-with-spotless-2bdh.

[38] *Errorprone*. [Loetud: 16-08-2022]. URL: https://errorprone.info/index.

[39] *AssertJ*. [Loetud: 18-08-2022]. URL: https://assertj.github.io/doc/.

[40] *Mockito*. [Loetud: 18-08-2022]. URL: https://site.mockito.org/.

[41] *Postman*. [Loetud: 22-08-2022]. URL: https://www.postman.com/.

[42] *DbSchema*. [Loetud: 13-08-2022]. URL: https://dbschema.com/.

[43] *App Diagrams*. [Loetud: 20-10-2022]. URL: https://www.diagrams.net/.

[44] Vullkan Halili. *Hexagonal architecture: What is it and why should you use it?* URL: https://cardoai.com/what-is-hexagonal-architecture-should-you-use-it/.

[45] Michelle Marcelline. *LocalStorage vs. Cookies: All You Need to Know About Storing JWT Tokens Securely in the Front-End*. URL: https://codeburst.io/localstorage-vs-cookies-all-you-need-to-know-about-storing-jwt-tokens-securely-in-the-front-end-70dc0a9b3ad3.

[46] Michael Coates. *Secure Cookie Attribute*. URL: https://owasp.org/www-community/controls/SecureCookieAttribute.

[47] *HttpOnly*. URL: https://owasp.org/www-community/HttpOnly.

[48] Sudip Sengupta. *What is CSRF Token*. URL: https://crashtest-security.com/csrf-token-meaning/.

[49] *What is Container Orchestration?* [Loetud: 19-11-2022]. URL: https://www.capitalone.com/tech/cloud/what-is-container-orchestration/.

[50] Kumar Harsh. *Kubernetes: Weighing Advantages and Disadvantages*. [Loetud: 19-11-2022]. URL: https://scoutapm.com/blog/kubernetes-advantages-disadvantages#h_704216136249716196497304 28.

[51] Mahesh Mogal. *What is AWS 15 reasons why should you choose it*. [Loetud: 20-11-2022]. URL: https://analyticshut.com/what-is-aws/.

[52] Naman Yash. *An Introduction Guide to AWS Fargate*. [Loetud: 20-11-2022]. URL: https://geekflare.com/aws-fargate-guide/.

[53] Naman Yash. *Application Load Balancer: Everything You Need to know*. [Loetud: 21-11-2022].

[54] Ben Lutkevich. *Amazon RDS (Relational Database Service)*. [Loetud: 02-12-2022]. URL: https://www.techtarget.com/searchaws/definition/Amazon-Relational-Database-Service-RDS.

[55] *Amazon Route 53*. [Loetud: 10-12-2022]. URL: https://aws.amazon.com/route53/.

[56] *What Is An SSL/TLS Certificate?* [Loetud: 11-12-2022]. URL: https://aws.amazon.com/what-is/ssl-certificate/.

[57] *AWS Certificate Manager*. [Loetud: 12-12-2022]. URL: https://aws.amazon.com/certificate-manager/.

[58] *Amazon CloudWatch*. [Loetud: 23-11-2022]. URL: https://aws.amazon.com/cloudwatch/.

[59] *Amazon SNS*. [Loetud: 24-03-2023]. URL: https://aws.amazon.com/sns/.

[60] *AWS Lambda*. [Loetud: 23-03-2023]. URL: https://aws.amazon.com/lambda/.

[61] *Create Incoming Webhooks*. [Loetud: 24-03-2023]. URL: https://learn.microsoft.com/en-us/microsoftteams/platform/webhooks-and-connectors/how-to/add-incoming-webhook?tabs=dotnet.

[62] *What is Docker?* [Loetud: 25-09-2022]. URL: https://www.ibm.com/topics/docker.

[63] Yolan Vloeberghs. *Jib: the next big thing to build your docker images*. [Loetud: 25-09-2022]. URL: https://ordina-jworks.github.io/docker/2020/05/07/jib.html.

[64] *Amazon Elastic Container Registry (Amazon ECR)*. [Loetud: 20-11-2022]. URL: https://aws.amazon.com/ecr/.

[65] *AWS Systems Manager Parameter Store*. [Loetud: 22-11-2022]. URL: https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html.

[66] *https://about.gitlab.com/topics/ci-cd/*. [Loetud: 04-12-2022]. URL: https://about.gitlab.com/topics/ci-cd/.

[67] *CI/CD pipelines*. [Loetud: 04-12-2022]. URL: https://docs.gitlab.com/ee/ci/pipelines/.

[68] *Apache JMeter*. [Loetud: 10-11-2022]. URL: https://jmeter.apache.org/.

[69] *What is Net Promoter Score?* [Loetud: 17-04-2023]. URL: https://www.hotjar.com/net-promoter-score/.

[70] *Maamess*. [Loetud:01-05-2023]. URL: https://maamess.ee/.

[71] Maxim Orlov. *Why storing files in the database is considered a bad practice*. [Loetud: 02-10-2022]. URL: https://maximorlov.com/why-storing-files-database-bad-practice/.

[72] *Amazon S3*. [Loetud: 13-04-2023]. URL: https://aws.amazon.com/s3/.

[73] Chris Rolls. *Rethinking the test automation pyramid*. [Loetud: 11-04-2023]. URL: https://ttcglobal.com/what-we-think/blog/rethinking-the-test-automation-pyramid.

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

Mina, Ain Kivimägi

1. annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Päikeseparkide projektide teenindusplatvormi tagarakendus", mille juhendajaks on Gert Kanter
   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.

21.05.2023

[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.

Lisa 2 – Registreerimise email arvutis ja telefonis

elos

Dear John,

Thank you for requesting your solar system design from ELOS.

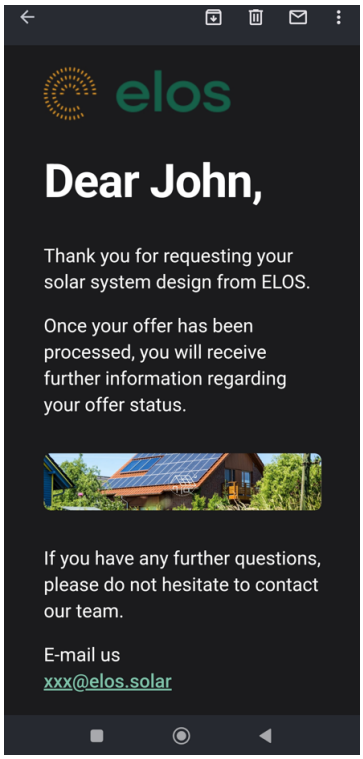
Once your offer has been processed, you will receive further information regarding your offer status.

If you have any further questions, please do not hesitate to contact our team.

E-mail us
xxx@elos.solar

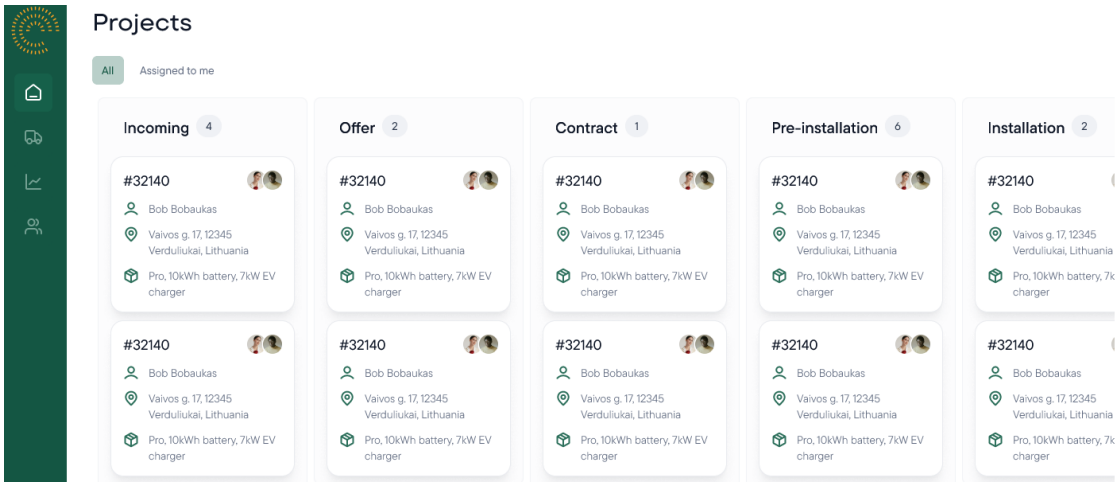
Call us
+370 1234 56787

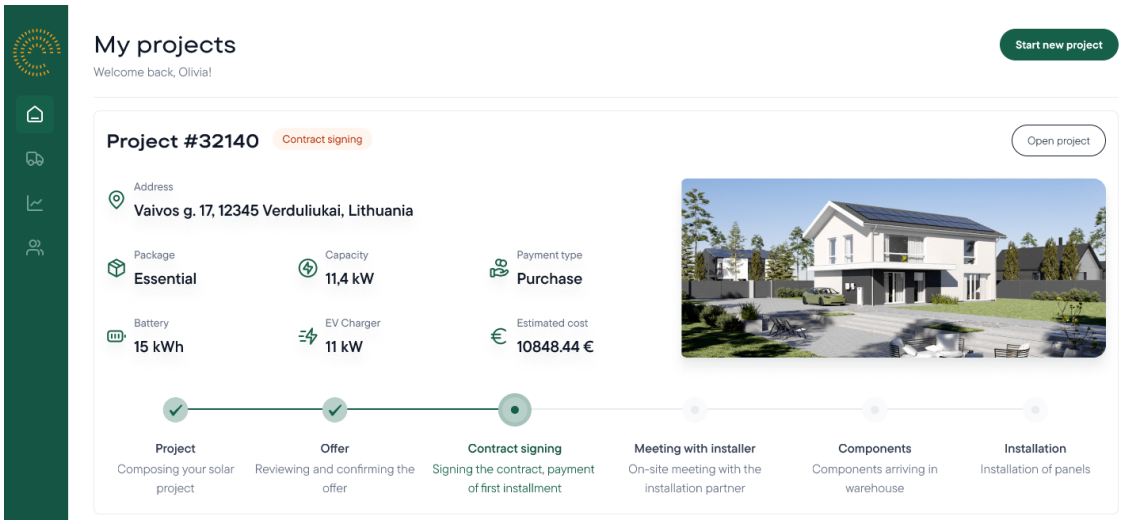
Have a wonderful day,
Your ELOS team



Lisa 3 - Projektijuhi ja kliendi projektide dashboard vaated

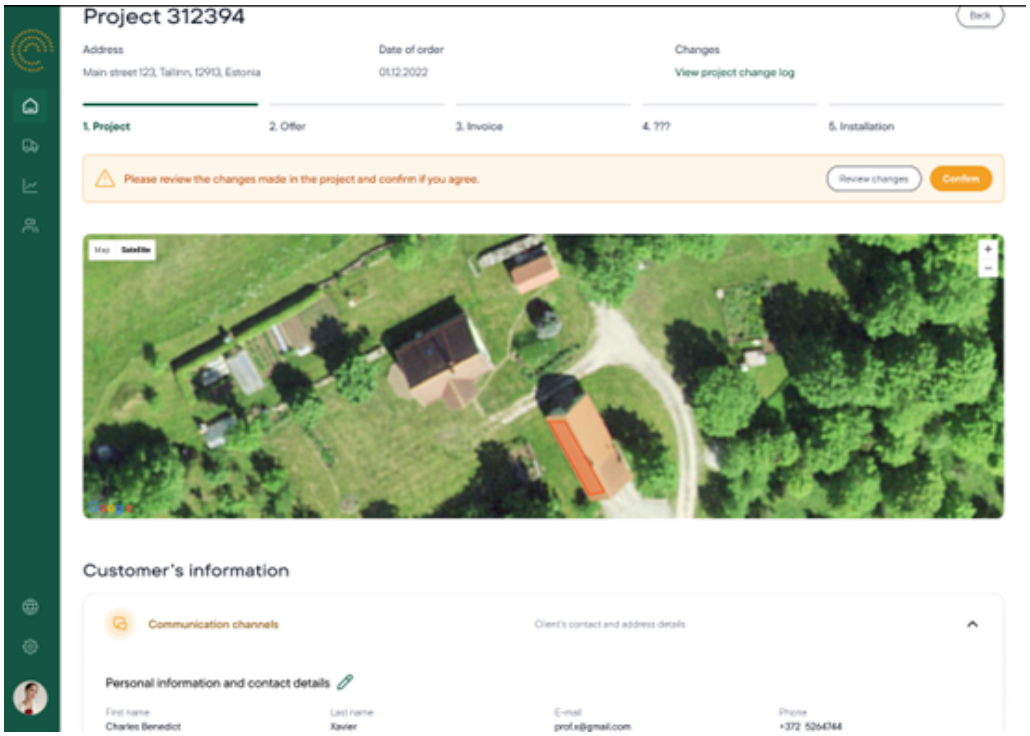
Vaade ei ole disainitud autori poolt.

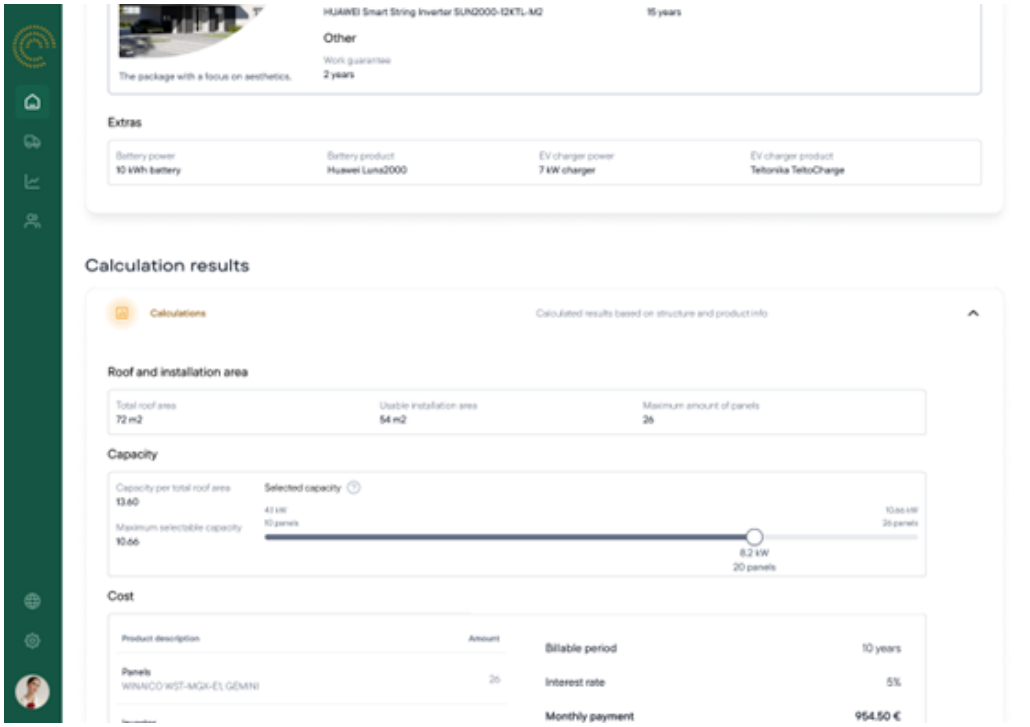


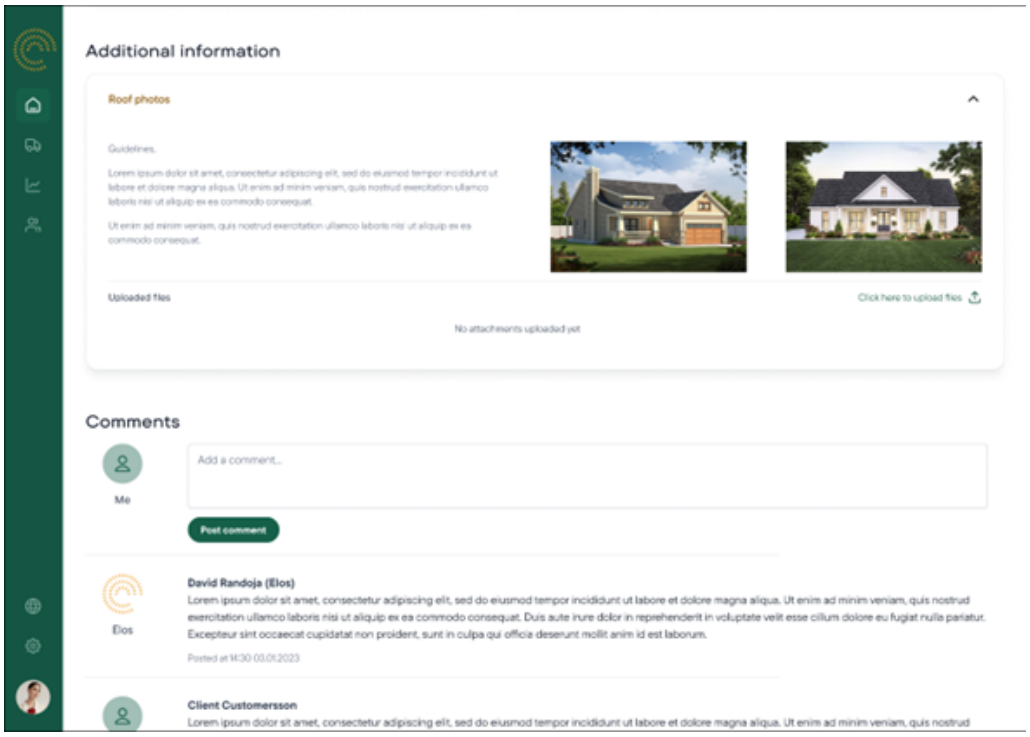


Lisa 4 – Tellimuse detailvaade

Vaade ei ole disainitud autori poolt.

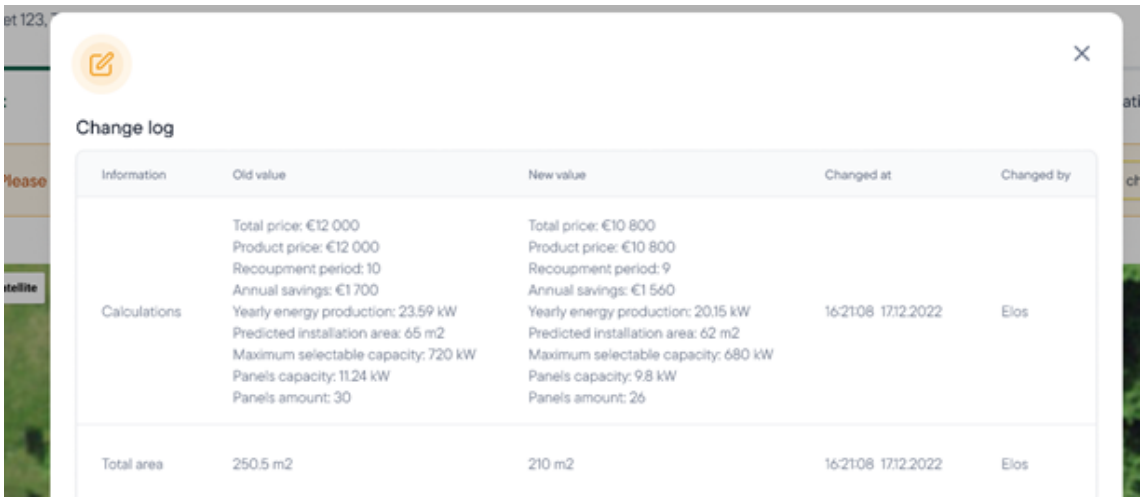




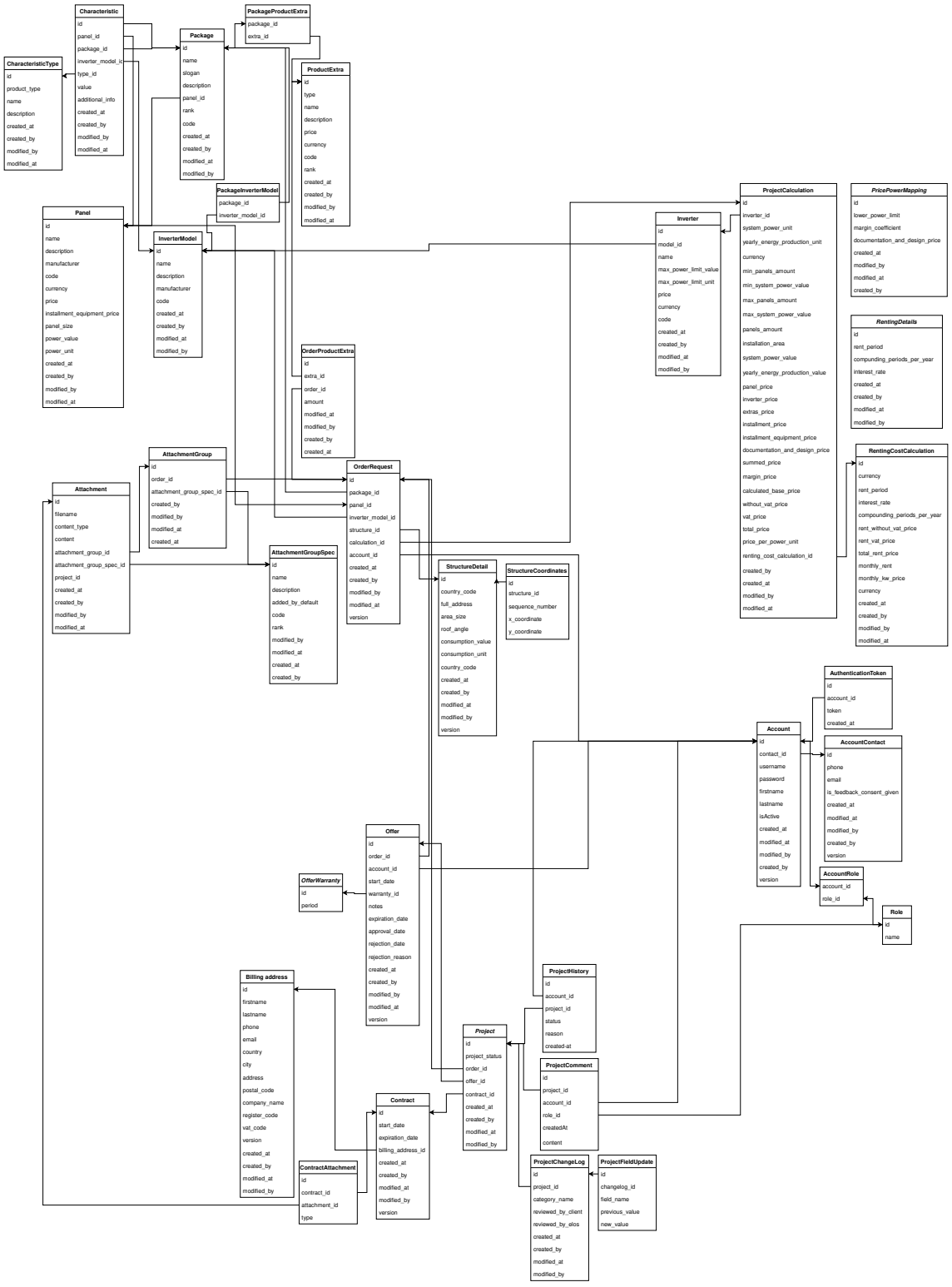

Additional information
Roof photos
Guidelines
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Uploaded files
Click here to upload files
No attachments uploaded yet
Comments
Me
Add a comment...
Post comment
Elos
David Randoja (Elos)
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Posted at 14:30 03.01.2023
Client Customersson
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud

Lisa 5 – Projekti muudatuslogi

Vaade ei ole disainitud autori poolt.



Lisa 6 – Andmebaasi skeem



Lisa 7 – AWS CloudWatchi veateade Teamsis

aws-test-monitoring 24.03 11:32

TEST elos-db-ram state is now ALARM: Threshold Crossed: 1 out of the last 1 datapoints [2.29687296E8 (24/03/23 09:31:00)] was greater than the threshold (3.0E7) (minimum 1 datapoint for OK -> ALARM transition).


```
{'AlarmName': 'TEST elos-db-ram', 'AlarmDescription': None, 'AWSAccountId': '926661572190', 'AlarmConfig
urationUpdatedTimestamp': '2023-03-24T09:31:03.957+0000', 'NewStateValue': 'ALARM', 'NewStateReason': 'T
hreshold Crossed: 1 out of the last 1 datapoints [2.29687296E8 (24/03/23 09:31:00)] was greater than the
threshold (3.0E7) (minimum 1 datapoint for OK -> ALARM transition).', 'StateChangeTime': '2023-03-24T09:
32:19.903+0000', 'Region': 'EU (Frankfurt)', 'AlarmArn': 'arn:aws:cloudwatch:eu-central-1:926661572190:a
larm:TEST elos-db-ram', 'OldStateValue': 'OK', 'OKActions': [], 'AlarmActions': ['arn:aws:sns:eu-central
-1:926661572190:elos-test-alarms'], 'InsufficientDataActions': [], 'Trigger': {'MetricName': 'FreeableMe
mory', 'Namespace': 'AWS/RDS', 'StatisticType': 'Statistic', 'Statistic': 'AVERAGE', 'Unit': None, 'Dime
nsions': [{'value': 'freetier-elos', 'name': 'DBInstanceIdentifier'}], 'Period': 60, 'EvaluationPeriod
s': 1, 'DatapointsToAlarm': 1, 'ComparisonOperator': 'GreaterThanThreshold', 'Threshold': 30000000.0, 'T
reatMissingData': 'missing', 'EvaluateLowSampleCountPercentile': ''}}```
```