TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

IC40LT

Urmas Öövel 175322IADB

Salvestuselektrijaama tarkvara arendamine

Bakalaureusetöö

Juhendaja: Kaido Kikkas
Kaasprofessor
Kaasjuhendaja: Heigo Mõlder
Insener

Tallinn 2022

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Urmas Öövel

09.05.2022

Annotatsioon

Käesolevas töös tegeletekse elektrijaama tarkvara arendamisega. Töö eesmärgiks on tarkvara loomine, mis toetaks salvestuselektrijaama tegevust: elektriturul osalemise tegevused, andmete salvestus hilisema esitusvõimalusega ning jälgimine ja juhtimine reaalajas. Arenduskeskkonnana kasutatakse Java Eclipse'i. Tarkvara toetab elektrijaama osalemist elektriturul. Teostatakse müüdud energiakoguste arvutamist andmebaasi kogutud arvestite näitude põhjal. Kliendi kasutajaliides toimib veebibrauseris. Andmebaasina kasutatakse MySQL'i. Veebiserverina kasutatakse Apache TomCat'i. Töö ühe osana valminud tarkvara põhifunktsionaalsus on veebipõhine lepingute sõlmimine elektrijaama klientidega ja tegelike tarbitud energiakoguste arvutamine. Kliendi jaoks on tehtud kasutajaliidesed tellimuse vormistamiseks ja temale teostatud arvutuste vaatamiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 35 leheküljel, 7 peatükki, 17 joonist.

Abstract

Software Development of Storage Power Plant

This thesis deals with software development for a power plant. The purpose of this software is creation of software, supporting storage power plant: activities for participation in electricity market, data storage with visualization possibilities, real time supervision and control. The development environment is Java Eclipse. The software supports participation in the electricity market. The energy consumption by customers is calculated, based on the energy metering data. The data is stored in MySQL database. The web container is Apache TomCat. The main functionalities of the software are: contract management with customers and calculation of consumed electrical energy. For the customer is made user interfaces for filling applications and watching energy calculations.

The thesis is in Estonian language and contains 35 pages of text, 7 chapters, 17 figures.

Lühendite ja mõistete sõnastik

API	*Application Program Interface*, rakenduse kasutajaliides
DAO	*Data Access Object*, andmete ligipääsu objekt
EJB	*Enterprise JavaBeans,* ettevõtte JavaBean-id.
GB	*GigaByte,* gigabait
HTML	*Hyper Text Markup Language,* kirjeldav päringukeel
ICT	*Information and Communication Technology*, info -ja kommunikatsiooni tehnoloogia
ID	*Identificator*, võti
IDEF0	*Integrated computer-aided manufacturing definition 0*, integreeritud arvutipõhise tootmise definitsioon 0
IT	*Information Technology*, infotehnoloogia
JDBC	*Java Database Connectivity,* Java andmebaasi liides
JEE	*Java Enterprise Edition*, Java versioon ettevõtetele
JSP	*JavaServer Pages,* Java Serveri lehed
LDAP	*Lightweight Directory Access Protocol,* kataloogipöörduse kergprotokoll
MVC	*Model View Controller*, mudel vaade kontroller
OS	*Operating system,* operatsioonisüsteem
RAM	*Random Access Memory,* operatiivmälu
REST	*Representational state transfer*, kirjeldav seisundi edastus
SCADA	*Supervisiory Control And Data Acquisition*, juhtimine ja monitooring
UML	*Unified Markup Language*, unifitseeritud märgendigeel
XPath	*XML Path Language*, XML Path keel
$	*Dollar of the United States,* Ameerika Ühendriikide dollar

Sisukord

Jooniste loetelu

1 Sissejuhatus

Käesolevas lõputöös käsitletakse salvestuselektrijaama tarkvara arendamist Java Eclipse keskkonnas. Praegust Eesti ühiskonda jälgides tekkis idee, et Eestis võiks olla salvestuselektrijaam. Idee on ajendatud sellest, et elektri hind muutub tihti ja elektri ebaühtlast genereerimist on palju. Salvestuselektrijaam täiendaks elektriga varustatust ja aitaks stabiliseerida elektrisüsteemi. Ebastabiilsus tuleneb tuule -ja päikeseelektrijaamadest. Salvestuselektrijaamadest kirjutatakse meedias ja mõnda neist juba rajatakse.

Töö eesmärgiks on tarkvara loomine, mis toetaks salvestuselektrijaama tegevust: elektriturul osalemise tegevused, andmete salvestus hilisema esitusvõimalusega ning jälgimine ja juhtimine reaalajas. Tarkvara loomine hõlmab ka andmebaasi struktuuri disainimist. Andmebaas on vajalik, et säilitada elektrijaama tegevuse ajalugu ja saada ülevaatlikke talitlusandmeid. Andmete ajalugu kasutatakse ka elektriturul osalemise äriloogikas. Rikete korral pakub statistika võimalust viga analüüsida. Käesolevas töös on peamine rõhuasetus elektriturul osalemise funktsionaalsuse arendamisel. Elektriturul osalemise funktsionaalsus hõlmab klientide tellimuste käitlemist ja klientide tarbitud elektrienergia koguste arvutusi.

Viiendas peatükis kirjeldatakse valminud tarkvara, mis on täielikult autori looming. Tarkvaraarenduse sisupoolelt väärivad esiletõstmist viiendas peatükis asuvad funktsionaalsuse UML skeemid koos äriloogika kirjeldustega. Joonistena esitatakse olulisemad kasutajaliidese vaated. Lisaks kirjeldatakse veel mitmeid vaateid teksti kujul. Samuti on tähtsateks autori väljatöötatud andmebaasiskeem, mille järgi arendati töötav andmebaas.

Töö neljandas peatükis kirjeldatakse arendusvahendeid ja meetodeid. Arenduskeskkonnana kasutatakse Eclipse'i. Turbe osa teostamiseks valiti Spring Security. Arendusmustrina kasutatakse Spring MVC'd ja andmebaasina MySQL'i. Autori panuseks oli erinevate osade kokku sobitamine, kõikehõlmavat näidet eeskujuks polnud. Töö praktiline osa tugineb ühe olemiga näitele, kus polnud olemite süsteemi. Samuti polnud sinna lisatud turbeosa.

2 Lähteülesanne

Käesolev lähteülesanne on spetsiifiline seetõttu, et seostab teadmisi elektri ja infotehnoloogia valdkondadest. Tegemist on tarkvara loomisega salvestuselektrijaamale, mille rajamise idee on käesoleva töö autori isiklik idee. Idee kohaselt planeeritakse ehitada uus salvestuselektrijaam. Jaama tüüp on liitium-ioonjaam. Võimsuse alusel on tegemist mikrojaamaga võimsusega 100 kW. Jaama talitluse käigus genereeritakse mitmete parameetrite mõõtmisi, mida on vaja nii jooksvalt kuvada kui ka andmebaasi koguda. Andmebaasi põhjal küsitakse perioodide andmeid, mis esitatakse visualiseeritult. Serveri rakendustarkvara arendusel kasutatakse Java Eclipse'i arenduskeskkonda.

2.1 Probleemi kirjeldus

Kaasaegse elektrijaama käitamiseks kasutatakse vastavat eriotstarbelist tarkvara. Kõnealuse elektrijaama jaoks pole veel vajalikku tarkvara – selle loomisega tegeletaksegi kavandatavas bakalaureusetöös.

Elektrijaama talitlusel on elektrotehniline ja äriline aspekt. Elektrotehnilisel poolel tegeletakse jaama igapäevase monitooringu ja tehnilise juhtimisega. Oluline on, et saavutatakse jaama reaalaja mõõteandmete jälgimine asukohast sõltumatult, mis vähendaks mehitatud valve tööjõukulu. Senistes kirjandusallikates käsitletakse enam tehnilist poolt, ärilise poole (elektriturul osalemine) kohta on tehtud vähem töid. Käesolevas töös asetatakse põhirõhk elektriturul osalemist toetava tarkvara loomisele. Vastavalt võimalustele käsitletakse ka tehnilist poolt.

Käesoleva bakalaureusetöö tegemisega luuakse valmidust elektrijaama projekti infotehnoloogilise osa teostamiseks käesoleva ja ka teiste salvestuselektrijaamade rajamiseks.

2.2 Ülesande püstitus ja eesmärk

Luua tarkvara järgmise funktsionaalsusega:

- Disainitakse andmebaasi struktuur kõigi mõõdikute andmete salvestamiseks. Personali pöördumisel tarkvara kasutajaliidese poole küsitakse defineeritud ajahetke või ajaperioodi kohta väljavõte andmebaasist ja andmed kuvatakse kliendile.
- Tarkvaraga automatiseeritakse elektriturul osalemiseks vajalikud tegevused: elektrijaama klientidel võimaldatakse esitada tellimusi; prognoositakse elektriturule pakutavate võimsuste tabelid; arveldatakse klientidele tegelikult müüdud energiakogused.
- Reaalaja talitluse andmed esitatakse mõõdikute paneelil. Mõõdikute paneeli vaadatakse interneti kaudu, valitud isikute poolt. Mõõdikute paneel teostatakse ainult virtuaalsena. Jaama personal peab saama mõõdikuid vaadata asukohast sõltumatult ja tavaarvuti abil.

3 Ülesande taust

Ülesande taust hõlmab mitmesuguseid tarkvarade näited, mida elektripaigaldiste jaoks maailmas on tehtud – siin mõeldakse mistahes elektripaigaldistes kasutatavaid tarkvaralahendusi. Näiteks võivad olla alajaamad, elektrijaamad, targad kodud. Näiteid käsitletakse pinnapealselt ja ei selgitata mõne lahenduse kasutatavust Eestis või käesolevas töös arendatava tarkvara asendusena. Käesolevas töös luuakse originaalset lahendust, mis vastab salvestuselektrijaama elektriturul osalemise autoripoolsele kuvandile. Pole teada, et täpselt niisugust tarkvara oleks keegi loonud, nišš erineb.

Elektriosa metoodika: Esimeses lahendusversioonis vaadeldakse võimalikult lihtsat skeemi, edasi liigutakse keerukamatele. Analüüsimisel kasutatakse skemaatilisi esitusviise: plokkskeemid koos seostega, plokid sisendite ja väljunditega. Salvestuselektrijaama eripära väljendub selles, et tema talitlusrežiimi hulka kuuluvad ka laadimistsüklid. Selle tõttu ei saa ta pidevalt nimivõimsust välja anda.

3.1 Elektriturg

Käesolevas töös kasutatav äriloogika sõltub hetkel olemasolevatest elektrituru reeglitest, kuid ei kopeeri olemasolevaid elektrituru reegleid täielikult.

Olemasolevad elektrituru reeglid: tavatarbija tellib elektrienergiat bilansihalduri vahendusel, mitte otse elektrijaamast. Bilansihaldur omakorda võtab energiat elektrijaamast. Kui ühest elektrijaamast ei saa või pole see piisav, siis ostetakse energia kusagilt mujalt. Üheks ostmise võimaluseks on ka elektribörs. Kui kokkuvõttes ikkagi jäi energiat puudu, siis bilansienergia ostetakse süsteemihaldurilt. [1]

Elektrituru reeglite komplektid käesolevas töös: Võimaldatakse kasutada erinevaid tururegleid. Põhiversioonis lähtutakse nendest tururegleitest, mis oleksid sellele elektrijaamale sobivad.

Elektrituru reeglite komplekt 1: Elektrijaam on ka oma klientide bilansihalduriks. Eeldatakse, et elektrijaam võib kliendiga lepingu sõlmimisest keelduda, kui tal endal

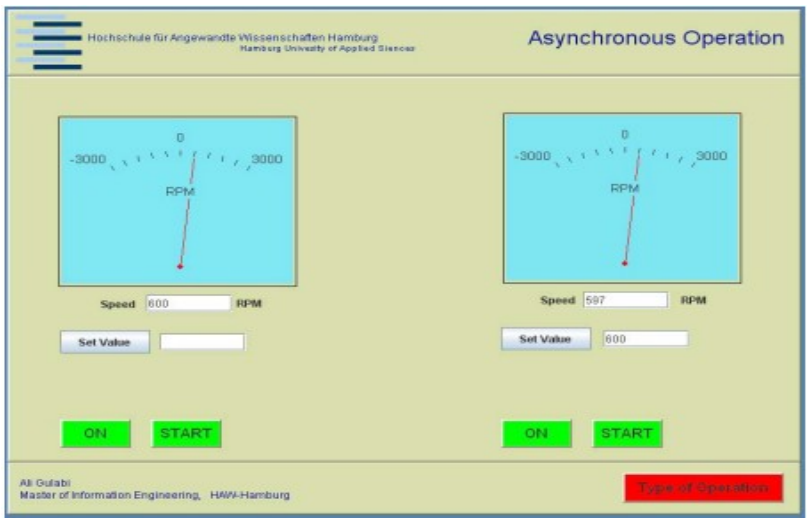
pole piisavalt võimsust. Bilansihalduriks on elektrijaam klientidele selle võimsuse ulatuses, mis elektrijaamal endal on. Uutest tellimustest keeldutakse, kui elektrijaamal endal energiat ei jätku. Uute tellimuste soovijad suunatakse selle kliendiga eelnevalt kokkulepitud bilansihalduri kliendiks.

Elektrituru reeglite komplekt 2: Elektrijaam on ka oma klientide bilansihalduriks. Eeldatakse, et elektrijaam ei või kliendiga lepingu sõlmimisest keelduda, kui tal endal pole piisavalt võimsust. Bilansihalduriks on elektrijaam klientidele selle võimsuse ulatuses, mis elektrijaamal endal on. Uutest tellimustest ei keelduta, kui elektrijaamal endal energiat ei jätku. Niisugusel juhul vahendab elektrijaam oma kliendile börsilt elektrit puudujäägi ulatuses.

Tüüpkoormusgraafikud: Esmaseks elektrijaama võimsuste graafikute arvutamiseks kasutatakse tüüpkoormusgraafikuid, mis pärinevad allikast. [2]

3.2 Elektripaigaldiste tarkvarade näiteid

Mootorite SCADA tarkvara Java Eclipse'i keskkonnas: Allikas [3] kirjeldatakse kasutajaliidese arendamist Java Eclipse'i keskkonnas. Järgneval joonisel (joonis 1) esitatakse peamise kasutajaliidese vaade.



Joonis 1. Mootorite juhtimise kasutajaliides [3] .

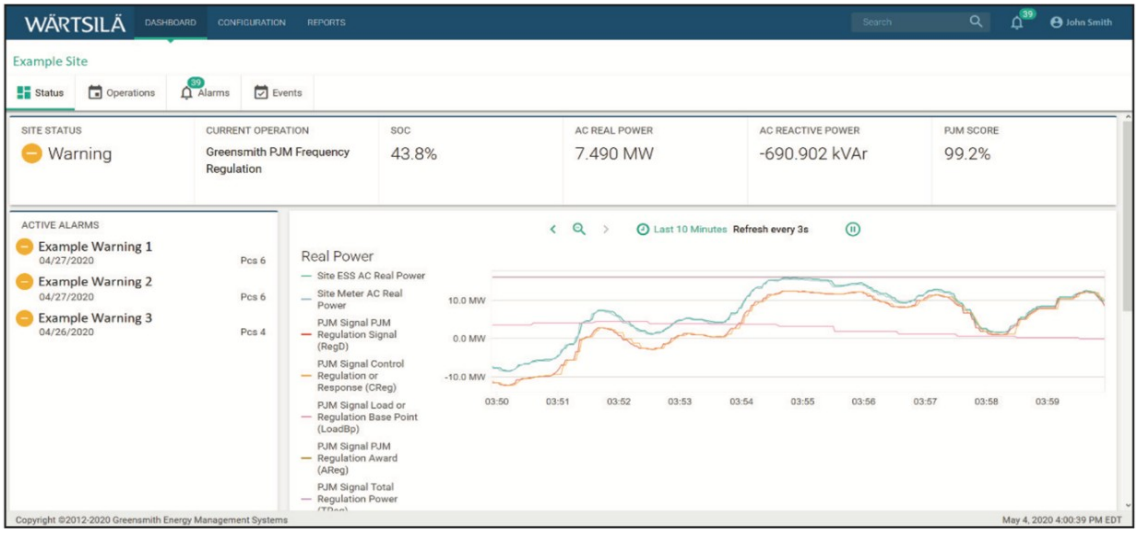
Süsteemiks on mootorite juhtimine, mille komponentideks on programmeeritav kontroller, mootorid ja veebibrauseris toimiv mootorite juhtimise kasutajaliides. Üks põhjus, miks mootoreid juhtida distantsilt, on inimeste ohutuse tagamine. Teiseks et

oleks võimalik juhtimisele ligi pääseda, olenemata personali hetke asukohast. Java tarkvara on üheks komponendiks SCADA süsteemis, mis monitoorib mõõteandmeid. Süsteemiga võimaldatakse ka mootoreid juhtida.

Wärtsilä salvestuselektrijaam Ameerika Ühendriikides [4] : See on üks esimesi näiteid, kus salvestussüsteemides kasutatakse tehisintellekti. Elektrienergiaga kaubeldakse elektriturul. Kõnealuses Wärtsilä hübriidjaamas kasutatakse pilve-põhist tarkvara „IntelliBidder". „IntelliBidder" tarkvara võimaldab elektri turuhinna ja taastuvenergia tootmise ennustamist, elektrijaama töögraafiku koostamist ja realiseerimist, automaatset pakkumiste genereerimist, pakkumiste sisestamist käsitsi, pakkumiste kinnitamist, tähtsamate indikaatorite jälgimist [5] . Wärtsilä ilmselt kasutab tarkvara arendamiseks kõige enam Java't [6] .

Origami elektrikaubanduse tarkvara [7] : Origami võimaldab seadmete mõõteandmete kogumist ja seadmete juhimist, täielikult automatiseeritud elektrijaama võimsuse planeerimist, seadmete jälgimist. Origami toetab osalemist kõikidel tähtsatel elektriturgudel nagu elektri hulgikaubandus, elektrisüsteemi tasakaalustusmehhanismid, sageduse ja talitluskindluse teenused.

Salvestussüsteemi energiavoogude juhtimise tarkvara [8] : Järgneval joonisel (joonis 2) esitatakse Wärtsilä arendatud salvestussüsteemide talitluse juhtimise paneeli näide.

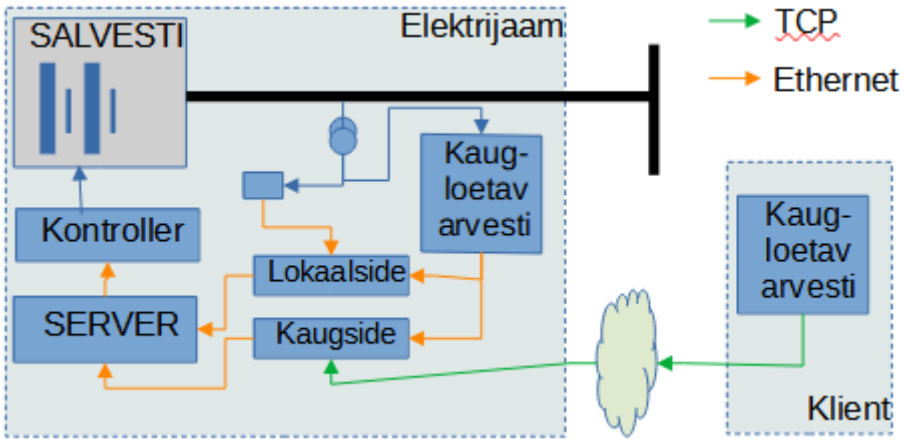


Joonis 2. Salvestussüsteemi mõõdikute paneel [8] .

Tarkvara võimaldab koordineerida taastuvenergia genereerimist ja salvestamist, erinevate seadmete juhtimist, kasumi maksimeerimist, elektrienergia genereerimise prognoosimist.

3.3 Planeeritav mõõtesüsteem

Järgneval joonisel (joonis 3) esitatakse mõõtestruktuuri üldskeem plokkidena. Kliendi arvestid on kaugloetavad. Eeldatakse, et arvestinäitude andmebaasi jõudmisel mõned lugemid lähevad kaduma ja andmebaasis on mõnede tundide näitude asemel lüngad. Arvestinäitude lugemine ei ole oma olemuselt sada protsenti töökindel. Tekivad lüngad – infosüsteem ei too neid andmebaasi. Ebatäiuslikke mõõteandmeid on vajalik töödelda kas visuaalse kontrolliga inimese poolt või estimeerimise kaudu poolautomaatselt. Poolautomaatselt selle tõttu, et mingites etappides on vajalik ka inimese hindamine isegi kui andmete parandamine programmiliselt lahendatakse.



Joonis 3. Mõõtestruktuuri üldskeem (autori joonis).

Käesolevas töös ei ole mõõtevigade tarkvaralist korrigeerimist kavas teostada. Inimese jaoks teostatakse käsitsi kontrollimise ja parandamise kasutajaliides veebibrauseris.

Kaugloetavate arvestite näite loetakse iga tunni tagant. Mingi kuu või päeva nulltund on samal ajal ka talle eelnenud perioodi viimase tunni lõpp. Seega on iga kuu kõige esimene näit samaaegselt ka talle eelnenud kuu viimane näit. Sama sõlmpunkti näit kaasatakse sõlmpunktile järgneva ja sõlmpunktile eelneva perioodi arvutusse. Näiteks 1.veebruari kell 00:00:00 on arvestinäit 100 kWh, 1.märtsi kell 00:00:00 arvestinäit 250kWh, 1.aprilli kell 00:00:00 300 kWh. Keskmist näitu 250 kasutatakse nii veebruari kui ka märtsi jooksul tarbitud energia koguse väljaarvutamiseks: veeruaris 250-100=150kWh; märtsis 300-250=50kWh. 1. märtsi näitu kell 00:00:00 kasutatakse seega nii märtsi esimesel tunnil kui ka veebruari viimasel tunnil tarbitud energiakoguse väljaarvutamiseks. Keskkoha (perioodi algus kell 00:00:00) mõõtetulemus on selle keskkoha arvesti näit, aga ei ole selle keskkoha tarbimine, sest keskkoht on lõpmatult väike ja arvesti näit asetub täpselt keskkohale.

4 Arendusvahendid ja metoodika

Käesolev peatükk sisaldab arendusvahendite kohta nii üldteadmisi kui ka väga käesoleva veebirakenduse arendamise spetsiifilist teavet. Info läbitöötamine ja esitamine siinkohal aitas edasi ka töö praktilise osana valminud tarkvara arendusele.

Käesolevas töös kasutatakse Linuxi operatsioonisüsteemiga arvutit. Linux sobib arendamiseks paremini eelkõige suurema kiiruse tõttu kui Windows [9] . Linux on ka turvalisem kui Windows [9] . Linuxi suurema kiiruse väide tugineb nii kirjandus-allikatele kui ka autori isiklikule kogemusele. Windowsil on rohkem lisapluginaid, mida töö tegijal vaja pole ning mis aeglustavad arvutit ja hajutavad tähelepanu. Arendamisel on tähtis kiirus, sest arendatavat tarkvara on vaja sageli käivitada ja taaskäivitada. Iga käivituse ajalise kestuse suurusjärk võib ulatuda minutitesse. Nii on oluline erinevus, kas taaskäivitus kestab sekundeid või kümneid sekundeid. Kokkuvõttes saab Linux hakkama väiksemate arvuti ressurssidega.

Käesoleva tarkvara arendamisel kasutatakse järgmisi töövahendeid:

- Acer Aspire 3 A315-33-P29Y (sülearvuti); Linux Ubuntu 20.04 (op. süsteem)
- MySQL Community Server; Qsee SuperLite
- Eclipse Java EE 2021-12; Java development kit 1.8; JavaSE-11
- Java Spring MVC; Spring Security
- Apache Tomcat 9.0.58 (server)

Arenduskeskkonna seadistamine esitatakse töö lisas (Lisa 2).

Lisaks kasutatakse veel toetava funktsiooniga tarkvara: LibreOffice (Writer, Calc, Draw); GNOME Screenshot; Ubuntu terminal; Chrome; Shotwell Viewer.

Kasutatud veebiteenused: YouTube; Google'i otsingumootor.

MySQL'i kasutamine põhineb ühel varasemal lahendusel. Kuna näites turbeosa üldse puudus, siis pole andmebaasi väljavahetamise otstarbekust analüüsitud. Baasnäitele on turbe ja olemite süsteemiga juba palju juurde ehitatud.

4.1 Arendusvahendite võrdlus

Peamised võimalikud tarkvara arendamise keeled suuremate veebirakenduste jaoks on Java ja C#. Arenduskeskkondadeks Java jaoks on Eclipse või IntelliJ. Arenduskeskkondadeks C# jaoks on Rider või Visual Studio. Arendamise raamistikuks Java'ga on Spring MVC ja C#'ga ASP.NET. Operatsioonisüsteemideks võivad olla Windows või Linux. Programmeerimiskeeltest, keskkondadest ja raamistikest on võimalik moodustada erinevate omadustega arendamise komplekte. Arendusvahendite komplekti omadusteks võivad olla hind, kasutusmugavus, paindlikkus, arendamise koguprotsessi kiirus, lõpliku rakenduse turvalisus, nõudlikkus riistvara võimsusele, testkäivituse kiirus arendusprotsessi käigus. Järgnev joonis (joonis 4) esitab kokkuvõtva võrdluse arenduskomplektidest, mis tugineb autori ülikoolis ja praktikal saadud kogemustele. Plussiga märgistatakse omadus, mille poolest üks komplekt on eelistatum.

Omadus Vahendite komplekt (operatsioonisüsteem; keel; keskkond; raamistik)	Hind	Mugavus	Paindlikkus	Riistvara võimsus	Arendamise kiirus	Testkäivituse kiirus	Turbe tundma õppimine	Olemite süsteemi õppimine	Laiendatavus seadmetele
Linux; Java; Eclipse; Spring MVC	+	+	+	+		+	+		
Linux; Java; IntelliJ; Spring MVC			+				+		
Linux; C#; Rider; ASP.NET					+			+	
Linux; C#; Rider; REST									+

Joonis 4. Arenduskomplektide võrdlus (autori joonis)

Allikale tuginedes on IntelliJ litsentsitasu 49$ ja Rider'i litsentsitasu 34$ kuus. Eclipse'il litsentsitasu pole [10] . Minimaalne RAM Eclipse'i korral 0,5 GB, IntelliJ jaoks 2GB [11] . Minimaalne RAM Rider'i jaoks on 4 GB [12] . Minimaalne protsessori kiirus Rider'i jaoks on 2 GHz [12] , Eclipse'i jaoks 0,8 GHz [11] . IntelliJ kasutamiseks on minimaalne protsessori kiirus autori hinnangu kohaselt 1,6 GHz.

Käesoleva töö tegemiseks kasutatakse sülearvutit protsessori kiirusega 1,6 GHz ja RAM'iga 8 GB.

4.2 Veebirakendus Eclipse'i arenduskeskkonnas

Eclipse'i üldteave: Eclipse on arenduskeskkond, kus arendatakse käesoleva töö osana valminud tarkvara. Käesoleval juhul kasutatakse Eclipse'i veebirakenduse kirjutamise keskkonnana. Eclipse keskkonnas kasutatakse programmeerimiskeelt Java. Järgnev Eclipse'i üldteave pärineb Ram Kulkarni raamatust „Java EE8 Development with Eclipse" [13] . Eclipse on platvorm ettevõtete rakendustarkvara arendamiseks. Eclipse'i tüüpe on mitmeid. Käesolevas töös kasutatakse „Eclipse Java Enterprise Editioni" ehk Eclipse JEEd. Java Enterprise Edition, lühendina JEE, on kollektsioon paljudest Java programmidest. JEE komponentideks on **esitluskiht, ärikiht** ja **ettevõtte integratsioonikiht**.

Esitluskiht: Esitluskihis on tehnoloogiad, mis võimaldavad võtta vastu päringuid veebiserverilt ja saata tagasi vastust, tüüpiliselt HTML formaadis. Võimalik on saada andmeid ka JSONi või XMLi formaadis. Esitluskihi klassid käivitatakse enamasti veebikonteineris. Veebikonteiner on osa rakendusserverist, mis käsitleb veebipäringuid. Apache Tomcat on näide populaarsest veebikonteinerist.

Ärikiht: Ärikihis kirjutatakse tavaliselt koodi, mis tegeleb rakenduse äriloogikaga. Pöördumised sellesse kihti peaksid tulema esitluskihist, otseselt kliendirakendusest, või veebiteenuste vahekihist. Selle kihi klassid käivitatakse rakenduskonteineris, mis on JEE serveri osa [13] . Lihtsaimaks äriloogika näiteks võib tuua hotelli broneeringud: ärikihis selgitatakse välja vabade tubade olemasolu. Teiseks näiteks võib tuua materjalide laoseisude arvutused kaubanduses ja logistikafirmades, näiteks ehituspoes kauba ostmisel laoseisu uuendamise äriloogika. Samuti mitmesugustel perioodidel müüdud kaubakoguste arvutamine; käivete arvutused; maksuametile aruandluse esitamise loogika. Äriloogika on võimalik kirjutada mitmesuguste valdkondade protsesside tõhustamiseks.

Ettevõtte integratsioonikiht: Integratsioonikihti kasutatakse nende ettevõtte süsteemidega suhtlemiseks, mis asuvad väljaspool Eclipse'i rakendust. Näiteks andmebaas või kasutajaliides.

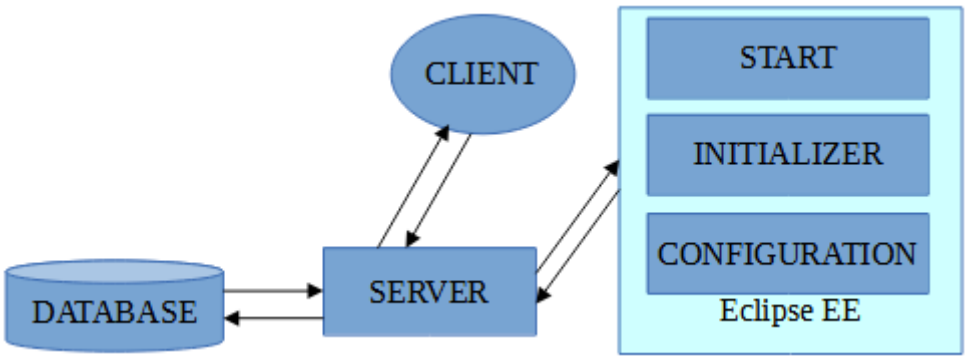
Java Database Connectivity: JDBC on spetsifikatsioon juurdepääsuks relatsioonilisele andmebaasile. JDBC-d kasutades saab käivitada SQL päringuid ja saada tulemusi erinevatest andmebaasidest kasutades harjumuspärast rakendusliidest.

Päringute haldamine: Päringute haldamisega võivad tegeleda nii **„Java Servletid"** kui ka **„Java Server Pages"**. „Java Servletid" on serveripoolsed moodulid, tüüpiliselt tegelevad päringute vastuvõtmisega ja nendele vastamisega. Servletid on kasulikud päringutega tegelemiseks, kui ei genereerita palju HTMLi vastuseks. Neid kasutatakse tüüpiliselt MVC raamistikes kui kontrollereid.

„Java Server Pages" (JSP-d) sobivad hästi päringute jaoks, millele vastamiseks genereeritakse palju HTML-i. JSP failides kombineeritakse Java koodi ja HTMLi koodi. Vajadusel saab lisada JavaScripti ja CSS-i. JSP lehtede kasutamine lihtsustab HTMLis vastuse koostamist.

Enterprise JavaBeans: EJB-d on klassid, kuhu kirjutatakse äriloogika. EJB-d võimaldavad mitmeid teenuseid, mis on olulised ettevõtete rakenduste juures. Niisugusteks teenusteks on turve, transaktsioonide haldus, komponentide otsing, objektide puulimine. EJB-d võivad olla kas Session bean- või message-driven bean -tüüpi.

Klient-server süsteem: Järgnevalt kirjeldatakse klient-server süsteemi põhimõtet, kui kasutatakse Eclipse arenduskeskkonda. Kirjeldus esitatakse joonis 5 selgitamiseks.



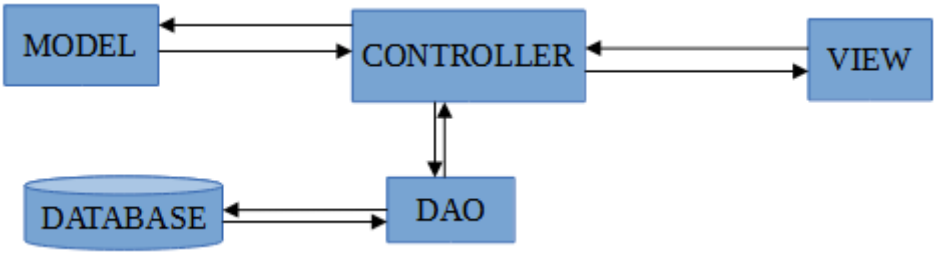
Joonis 5. Klient-server süsteem [14] .

Töötava tarkvara vaates on server keskne abstraktsioon. Tarkvara arendaja vaates on keskseks arenduskeskkond Eclipse. Kui Eclipse'i keskkonnas on valminud kood, siis see antakse käiku serveris. Kogu liiklus serveri ja teiste abstraktsioonide vahel on kahesuunaline. Klient füüsilises tähenduses on arvuti ekraan, infotehnoloogiliselt

veebibrauser. Klient koosneb nii sellest, mida inimene arvutiekraanilt näeb kui ka veebibrauseris varjatult asuvast infost. Klient saab esitada päringuid, server koostab vastuse. Andmebaasis asub näiteks arvestinäitude info. Kui kliendi päringule vastamiseks on vajalik, siis server pöördub omakorda päringutega andmebaasi poole.

Eclipse saab paigaldada rakenduse serverisse, saab anda „start" ja „stop" korraldusi serverile ja saata serverile programmiuuendusi. Server saab saata jooksvaid raporteid korralduste täitmise kulgemise kohta. Joonisel on nähtav ka Eclipse'i arenduskeskkond. Arenduskeskkonnaga tegeleb tarkvara väljatöötaja. Kui süsteem on valmis ja talitleb, siis arenduskeskkond otseselt ei osale tegevuses. Lisaks arendusele on Eclipse kasutatav testkeskkonnana. Kui Eclipse'is on tarkvarauuendused valmis saanud, siis laaditakse uuendused serverisse, mis alustab talitlust uuendatud versiooniga.

Spring MVC Eclipse'i keskkonnas: Spring MVC veebirakenduse arendamise raamistikus koostatavate skriptide struktuur esitatakse joonisel 6, mida kirjeldatakse järgnevalt. Äriloogika asub kontrolleris. Vaade on see, mida inimene arvuti ekraanilt nägema hakkab. Kontroller pidevalt „kuulab" „vaadet". Kliendil on võimalus esitada



Joonis 6. Spring MVC Eclipse keskkonnas [14] .

päringuid. Serveris on paigaldatud ja seadistatud süsteem „Kontroller-Mudel-DAO", mis teostab päringutele vastamise. „DAO" on serveri poolne abstraktsioon päringute teostamiseks. Mudelis asuvad etalonid. Mudelil on keeruline tarkvaraarenduslik sisu. Iga mudel koosneb atribuutidest. Atribuut on sisuliselt mudeli muutuja. Kui vaadelda näiteks olemi „ISIK" mudelit, siis selle atribuutideks võivad olla: peakate, nimi, telefoninumber, silmade värv, isikukood, sünnikuupäev. Atribuudi „peakate" väärtusteks erinevate isendite puhul võivad olla: kaabu, müts, soni. Mudelis asuvad olemite kirjeldused atribuutide kaudu, aga mudelis ei asu ühtegi isendit. Kui keegi vaate poolt küsib, et näiteks anna mulle konkreetse isiku „X", andmed, siis küsitakse

andmebaasist tema andmeid. Kui küsitakse isiku peakatet, siis andmebaasist otsitakse isiku „X" peakate. Võidakse tahta isikut ka kõigi tema omadustega, siis selle jaoks kasutatakse mudelit kui plaani. Süsteemi osade koostööna päritakse huvipakkuv info andmebaasist. Info andmebaasist vormistatakse mudeli põhjal kontrollerisse ja „isend" saadetakse vaatesse. Iga isendi eristamiseks andmebaasis kasutatakse unikaalset tunnust. Lihtsaimaks tunnuseks on näiteks erinev number.

Koodinäiteid vaate, vaatemudeli, DAO, kontrolleri ja mudeli atribuutide saatmise kohta esitatakse Lisas 4. Sealhulgas on näiteks kliendi tellimuse vaate näide esitatud joonisel 19, samale vaatele vastav vaatemudel joonisel 18. Kliendi tellimuse vaatega seotud kontrolleri meetod esitatakse joonisel 22. Näitena saab vaadata veel teist kontrolleri meetodit tellimuse salvestamise kohta joonisel 23. DAO näide esitatakse joonisel 24.

4.3 Turbe arendamise raamistik Spring Security

Iga veebirakenduse lahutamatuks osaks on ka turbe arendamine. Ilma turbeta on võimalik ründetundlikes süsteemides palju negatiivset teha: info varastamine, võltsimised, rakenduse aeglustamine ja halvamine, andmete muutmised. Näiteks avalike infosüsteemide tööseisakud. Käesolevas alapeatükis käsitletakse turvanõrkuste kõrvaldamiseks vajalikku teavet. Spring Security on Spring-raamistikus arendatud veebirakenduste turvanõrkuste kõrvaldamise töövahend. Järgnevalt loetletakse turvanõrkused ja kirjeldatakse valikuliselt.

Turvanõrkused veebirakendustes [15] [16]

- Vigane autentimine (broken authentication)
- Seansipete (session fixations)
- Skriptisüstimisrünne (cross-site scripting – XSS)
- Päringuvõltsing (cross-site request forgery – CSRF)
- Süstimisrünne (injections)
- Tundlike andmete paljang (sensitive data exposure)
- Puudulik pääsumeetodi reguleerimine (lack of method access control)
- Tuntud nõrkustega sõltuvuste kasutamine (using dependencies with known vulnerabilities)

Märkus: Eestikeelsed terminid on võetud allikast [16]

Veebirakenduste levinumaid ründetüüpe on kaheksa. Järgnevalt kirjeldatakse neid valikuliselt. Rünnete definitsioonid on võetud allikast „Spring Security in Action" **[15]** .

Süstimisründeid võib olla mitut tüüpi: skriptisüstimine, SQL süstimine, XPath süstimine, OS käsusüstimine, LDAP süstimine. Süstimisründe tüüpe on rohkem, kui siinkohal loetletud. Levinuim on SQL süstimine. SQL süstimisel ründaja proovib käivitada erinevaid SQL päringuid, et muuta, kustutada või varastada andmeid süsteemist. Süstimisründe kindlateks on vaja arendada eeskätt pangasüsteemid ja hääletussüsteemid. Käesoleva veebirakenduse korral tähendaks see seda, et isik saaks talle mitte kuuluva arvesti näite vaadata, sisestades päringu mõnda sisestuslahtrisse. Samuti saaks saata ühe inimese elektriarvet kellelegi teisele maksmiseks. XPath süstimisrünnet selgitatakse allikas [17] .

Seansipete võimaldab genereeritud sessioonivõtit (ID) korduvalt kasutada. See tähendab, et seanssi ei lõpetata ära, kui kasutaja on näiteks vaated sulgenud korralikult välja logimata. Seansil peaks olema ajaline kestvus, mille jooksul seanssi värskendamata seanss lõpetatakse. Seanss tuleks lõpetada ka siis, kui kasutaja pole määratud aja jooksul aktiivne olnud. Seansipete on esinenud käesoleva töö koostamise ajal 11. aprillil 2022 Gmail postkastil ja Facebookil.

Päringuvõltsingu korral saab serveri ja kliendi vahelist päringu täitmise protsessi mõjutada mitme rakendusliidese kaudu. Näiteks on võimalik e-poes osta teise inimese raha eest esimesele inimesele kaupu.

Autentimisel selgitatakse isiku juurdepääsuõigust. Veebirakendustele juurdepääs on tavaliselt rollipõhine. Turbes eristatakse mõisteid autentimine ja autoriseerimine. Autoriseerimise üldpõhimõte arendaja seisukohast: igal veebirakendusel on tavaliselt kõigile nähtavad vahelehed ja piiratud juurdepääsuga vahelehed. Juurdepääsu piiramiseks kasutatakse filtreerimist. Kõige tavalisemalt kasutatakse filtreerimist kasutajatunnuse ja salasõna alusel. Autentimisel tuvastatakse isiku ligipääsuõigus mingile infole, autoriseerimisel tuvastatakse lisaks ka see, kas isik on info loojaks olnud.

Konfiguratsioonifail: Turbe arendamise esimene samm on XML konfiguratsioonifaili loomine. Fail kirjeldab kõik vajalikud Spring Security komponendid, mis on vajalikud tüüpiliste veebipäringute katmiseks [18] . Joonisel 7 esitatakse näitlikustamiseks fragment ühest võimalikust konfiguratsioonist.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/security"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:beans="http://www.springframework.org/schema/beans"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
```

Joonis 7. Spring Security konfiguratsioonifail [18] .

Konfiguratsioonifaili alternatiiviks on turbe seadistamine Java koodiga, mida kirjeldatakse Nam Ha Minh allikas [19] . Algajale arusaadavam on turbe teostamine Java koodiga, mida kasutatakse ka käesolevas töös.

4.4 Andmebaasitarkvara MySQL

MySQL on andmebaasitarkvara, mida käesolevas töös kasutatakse. MySQLi seadistus esitatakse lisas. Käesolevas töös kasutatakse palju ajatöötlust ja ajaga seotud operatsioonid on siin ühed keerukamad. Järgnevalt esitatakse arendamise jaoks vajalikku infot.

JAVA arenduskeskkonnas kasutatavad ja MySQL'is kasutatavad andmetüübid peavad ühilduma. JAVA keskkonnas on aja tüübi nimetus „Timestamp", millele vastab MySQL'i andmebaasi ajatüüp „DateTime". Teisiti sõnastades MySQL'i „DateTime" andmetüübile vastab Java klassides kasutatav andmetüüp „TimeStamp". Käesolevas rakenduses kasutatakse „DateTime", sest energia arvutused tehakse tunniajalise täpsusega. Arvestite näidud loetakse samuti tunnise täpsusega. Timestamp andmetüübi kohta on kasulikke näiteid „Dariawan" allikas[20] . Näide DateTime kasutamise kohta MySQL'i andmebaasitabelis: olgu andmebaasis tabel „periood". Näiteks periood „2022 Jaanuar" kestab esimesest jaanuarist kell 00:00 kuni esimese veebruarini 2022 kell 00:00.

Näitepäring tabelist **„periood”** vastusega esitatakse joonisel 8:

```
mysql> select * from periood;
+-----------+---------------------+---------------------+---------------+
| perioodid | alates              | kuni                | pernimetus    |
+-----------+---------------------+---------------------+---------------+
|         1 | 2022-01-01 00:00:00 | 2022-02-01 00:00:00 | 2022 Jaanuar  |
|         2 | 2022-02-01 00:00:00 | 2022-03-01 00:00:00 | 2022 Veebruar |
|         3 | 2022-03-01 00:00:00 | 2022-04-01 00:00:00 | 2022 Märts    |
|         4 | 2022-04-01 00:00:00 | 2022-05-01 00:00:00 | 2022 Aprill   |
|         5 | 2022-05-01 00:00:00 | 2022-06-01 00:00:00 | 2022 Mai      |
|         6 | 2022-06-01 00:00:00 | 2022-07-01 00:00:00 | 2022 Juuni    |
|         7 | 2022-07-01 00:00:00 | 2022-08-31 00:00:00 | 2022 Juuli    |
+-----------+---------------------+---------------------+---------------+
```

Joonis 8. DateTime andmebaasis (autori joonis)

Päring võtmesõnaga „BETWEEN” tagastab väärtused defineeritud vahemikus koos otspunktidega [21] . Järgnevalt esitatakse näitepäring tabelist **„arvestinait”** vastusega (joonis 9).

```
mysql> SELECT * FROM arvestinait WHERE arvestiid=1
       AND aeg BETWEEN '2022-01-01 00:00:00' AND '2022-02-01 00:00:00';
+---------------+-----------+---------------------+---------+
| arvestinaitid | arvestiid | aeg                 | energia |
+---------------+-----------+---------------------+---------+
|             1 |         1 | 2022-01-01 07:00:00 |      10 |
|             2 |         1 | 2022-01-01 08:00:00 |      20 |
|             3 |         1 | 2022-01-01 09:00:00 |      30 |
+---------------+-----------+---------------------+---------+
```

Joonis 9. Ajavahemiku järgi filtreerimine (autori joonis)

4.5 Valitud meetodid

Arenduskeskkonnana kasutatakse Java Eclipse’i, arendusmustrina Spring MVCd. Tegemist on klient-server tüüpi veebirakendusega. Andmebaasina kasutatakse MySQL relatsioonilist andmebaasi. Serverina kasutatakse Apache TomCat serverit. Tarkvaraarenduse üldmeetodina kasutatakse Kanbanit. Programmeerimise tüübina kasutatakse objektorienteeritud programmeerimist, andmebaasi tüübina relatsioonilist andmebaasi. Andmebaasi disainimisel kasutatakse olemi-suhte diagrammi. Kanbani metoodikale tuginedes teostatakse kõigepealt lihtsaim võimalik tarkvarastruktuur, mida laiendatakse astmeliselt tagasiside põhjal. Äriprotsesside arendamiseks kasutatakse tarkvara analüüsimiseks mõeldud visualiseerimise tehnikaid: UML diagramm, IDEF0 protsessivoo diagramm. IDEF0 on UML’iga sarnane protsessiloogika visuaalse arendamise meetod, kuid veidi suuremat detailsust võimaldav.

Käesoleva bakalaureusetöö tegemise etapid:

- Esmase lühikese teoreetilise osa tegemine
- Arenduskeskkonna seadistamine ja testimine mistahes ühe lihtsa olemiga
- Esimese vajaliku olemiga rakenduse tegemine
- Rakenduse laiendamine kuni viiele olemile
- Esmase turbe arendamine
- 10...15 olemiga tuumikfunktsionaalsuse toorversiooni tegemine
- Tuumikfunktsionaalsuse täiustamine, kasutajaliidese silumine
- Turbe täiustamine

Vaheetappide tähistamine:

Järgneval joonisel (joonis 10) on detailselt esitatud kõik eesmärgiks olnud funktsioonid.

Funktsioon	Esmane	Lõppversioon	Täiustatud
Andmebaasi disain ja programmeerimine	+	+	
Andmete päring perioodi järgi ja kuvamine kliendile	+	+	
Andmete päring ajahetke järgi ja kuvamine kliendile	+	+	
Reaalaja mõõdikute paneel	+	+	
Kliendi tellimuse esitamine	+	+	
Pakutavate võimsuste tabelid	+	+	
Müüdud energiakoguste arvutamine	+	+	

Joonis 10. Eesmärgistatud funktsionaalsus

Bakalaureusetöö raames lubati teostada seitse funktsionaalsust. Tabel näitab ära, millised staadiumid on töö juba läbinud ja milliseid staadiume on võimalik veel läbida. Samuti näitab ka hetkeseisu. Näiteks töö kolmandasse staadiumisse jõudnud funktsionaalsus on eriti hästi lihvitud. Niisugust tabelit on hea kasutada töö käigus ja see annab kiire ülevaate ka sellest, kuhu tööga on jõutud. Töö käigus abivahendina kasutades on niisuguse tabeli abil hea seada vahe-eesmärke, mida mingil hetkel hakata süvendatult arendama. Töö vaheetappides aitab see hetkeseisu ülevaatlikustada. Kui mingil põhjusel mõnda funktsionaalsust ei jõuta piisavalt heaks lihvida, siis niisugune tabel näitab ära ka selles mõttes töö nõrkused.

Lõppversiooni ja täiustatud tulemuse vahel on erinevus selles, et näiteks kui on lõppversioonini jõutud ja siis tekkis mingi mõte, et saaks kuidagi veel paremaks teha, aga mida „täiustatud" etapis pole tehtud. Kaasnevad vajalikud tegevused, mida põhifunktsionaalsuse tabel ei sisalda: turbe täiustamine, testimine, koodi silumine, mõõteandmete kogumine ja salvestamine.

Analüüs ja disain [22] : Tööstuslike IT – süsteemide disain algab kahe seotud arhitektuuri arendamisest: funktsionaalsuse ehk tegevuste mudel ja informatsiooni- ehk andmemudel.

IDEF0 metoodika tegevuste mudeli arendamiseks [22] : IDEF0 metodoloogia järgi illustreeritakse tegevuste mudelit, kus komponentideks on: tegevusplokid, sisendid, väljundid, juhtsignaalid ja mehhanismid. Modelleerimise protsessis dekomponeeritakse funktsioonid hierarhiliselt „lapsevanem" ja „laps" tegevusteks. Igal dekompositsiooni tasandil saavutatakse terviklikkus „laps" tegevustega - „laps" tegevused pärivad sisendid ja väljundid „lapsevanem" tegevusdiagrammilt. Modelleerimisprotsess kestab niikaua, kuni saavutatakse kõigi elementide piisav detailsus: iga tegevuse esitus peab olema arusaadav ja kuidas nad on seotud teiste tegevustega infovoogude kaudu.

Töö arendamine vestlusringis: Käesoleva töö teoreetilise osa täiendamisel kasutatakse temaatilisi vestlusi juhendajatega. Lisaks vesteldakse ja esitletakse tööd juhuslikele inimestele, kellel erialaseid teadmisi pole. Võhikutega vestlemisega täiustatakse töö sidusust.

Alternatiivid valitud metoodikale: REST [23] - *Representational State Transfer* on arhitektuuriline stiil standardite pakkumiseks arvutisüsteemide vahel veebis, muutes süsteemide omavahelise suhtlemise lihtsamaks. REST arhitektuurilises stiilis saab serveri arendust teha sõltumatult veebis asuvast kliendi kasutajaliidesest. Eraldades kasutajaliidese, täiustatakse kasutajaliidese paindlikkust platvormide lõikes ja paraneb rakenduse laiendatavus, lihtsustades serveri komponente. REST aitab saavutada töökindlust, kiirust ja laiendatavust[23] . REST arhitektuuriga arendamine on keerulisem kui MVC'ga. Mõeldav on kõigepealt arendada töötav süsteem MVC'ga, mille analoog teostada hiljem REST'iga.

5 Loodav lahendus

Töö tulemuseks on rollipõhine veebirakendus. Eristatakse klienti ja elektrijaama personali. Klient saab vaadata oma elektri ostmise ajalugu ja sõlmida elektri ostu lepingut. Rakendus arvutab ostetud elektri kliendile kuuluva arvesti näitude põhjal kokku. Elektrijaama personal saab vaadata klientide elektritarbimist ja kinnitada ostu-müügi lepinguid. Samuti vaadata elektrijaama talitluse ajalugu. Samuti et saaks vaadata elektrijaama mõõdikuid reaalajas.

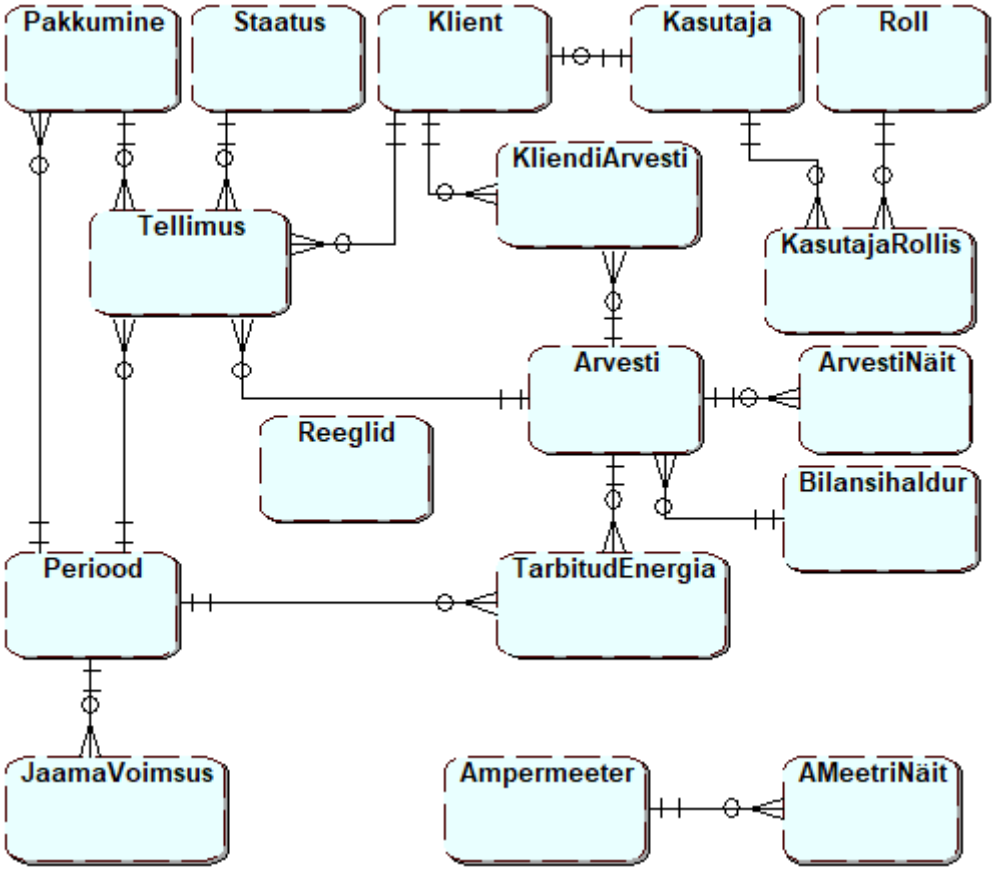
Nõuded arendatavale tarkvarale: Kliendi vaated peavad olema eestikeelsed; tellimusi ei tohiks saada esitada rohkem, kui elektrijaam suudab teenindada.

5.1 Andmebaasi struktuur

Andmebaasis olevate tabelite ja nende tabelite omavaheliste suhete disainimiseks kasutatakse olemi-suhte diagrammi. Diagrammi õige struktuur on aluseks, et edasisel arendamisel saaks funktsionaalsust korrektselt üles ehitada.

Olemi-suhte diagramm: Järgnevalt toodud diagramm (joonis 11) kirjeldab jaama andmebaasi struktuuri, mis on vajalik elektriturul osalemise funktsionaalsuse saavutamiseks. Protsess algab sellest, et jaamal on pakkuda elektrit müügiks. Klient saab pakkumist näha veebilehel ja saab vormistada tellimust elektri ostmiseks. Jaam saab tellimusi kinnitada. Jaam on sisuliselt ka bilansihaldur. Selleks kogub ta tunninäite oma klientide arvestitelt. Jaam määrab ära ka oma elektri müügihinna. Arvestinäitude voogude põhjal arvutab tarbitud energiakoguste vood ja summad tundide kohta, samuti summad perioodide kohta. Olem „Arvesti”: üldjuhul on siinkohal tegemist kliendi liitumispunktis asuva arvestiga.

Periood on vajalik selleks, et klient saaks perioodide arvutustulemusi küsida nii, et talle pakutakse perioodide loetelu valimiseks. Esmaselt perioodide tabel täidetakse mõne elektrijaama töötaja poolt.



Joonis 11. Olemi-suhte diagramm.

Autentimise funktsionaalsuse teostamiseks on ka olemid „Kasutaja”, „Roll”, ja „KasutajaRollis”. Need on üks-mitmene suhe: kasutajal võib olla üks roll ja rollil võib olla mitu kasutajat. Tabel „KasutajaRollis” näitab ära milline kasutaja on millises rollis. See, et rollil võib olla mitu kasutajat, tähendab sisuliselt seda, et kui kliendi rollis inimesi on rohkem kui üks (ja nii tavaliselt ongi). Kasutajal saab olla mitu rolli - näiteks siis, kui mõni klient peaks ühel päeval elektrijaama töötajaks astuma. Siis võimaldab tabel näidata, millistes rollides on kasutaja erinevatel ajaperioodidel olnud. Üldjuhul on siiski igal kasutajal konkreetne roll. Mitu rolli on ka elektrijaama töötajal, kes soovib samal ajal tarbida ka elektrijaama teenuseid. Näiteks tarkvara testival töötajal võib olla vajadus rolli vahetada. Kui tõepoolest mitmes rollis olev kasutaja on olemas ja siise logib, siis peaks ta saama valida, millises rollis ta hetkel olla tahab. Võimalik on ka nii teha, et mitmes rollis olija jaoks luuakse kaks kontot. Mitu-mitmese suhte kasutamine

võimaldab andmebaasi seisukohalt teha mistahes moodi: kas kasutajal on mitu rolli või mitme rolli asemel mitu kontot.

Jaama võimsuse olem on vajalik sellepärast, et elektrijaama välja antaval võimsusel on tehnilised piirid. Sellest sõltub, kui palju elektrijaam suudab kliente teenindada. Jaama võimsuse tabelis esitatakse tundide kaupa plaanilised välja antavad võimsused. Seega on jaama võimsus seotud jaama talitluse ette planeerimisega. Selle alusel hakatakse tellimuste koguvõimsust piirama. Jaama piirvõimsuste plaanid kaasatakse tellimuste esitamise protsessi, mis kontrollib, et kas vaba võimsust on.

Olemid „Ampermeeter" ja „Ampermeetri Näit": Ampermeeter on elektrijaama sisene tehniline mõõteriist. Ampermeetri näite esitatakse reaalaja mõõdikute paneelis.

MySQLi andmebaasi loomine: tabelite loomine on esitatud lisas (Lisa 3).

5.2 Funktsionaalsus

Algne ülesanne: teostada toetavat funktsionaalsusega tarkvara salvestuselektrijaamale. Arendussuundi on kaks: äriline ehk elektriturul osalemist toetava tarkvara arendamine ja elektrotehnilise tarkvara arendamine, mis hõlmab reaalaja toimepidevuse tagamist ja reaalajas jälgimist. Lahendus: elektriturul osalemise protsesside toetamiseks teostati tarkvara järgmiste võimalustega:

Elektrijaama kliendid saavad:

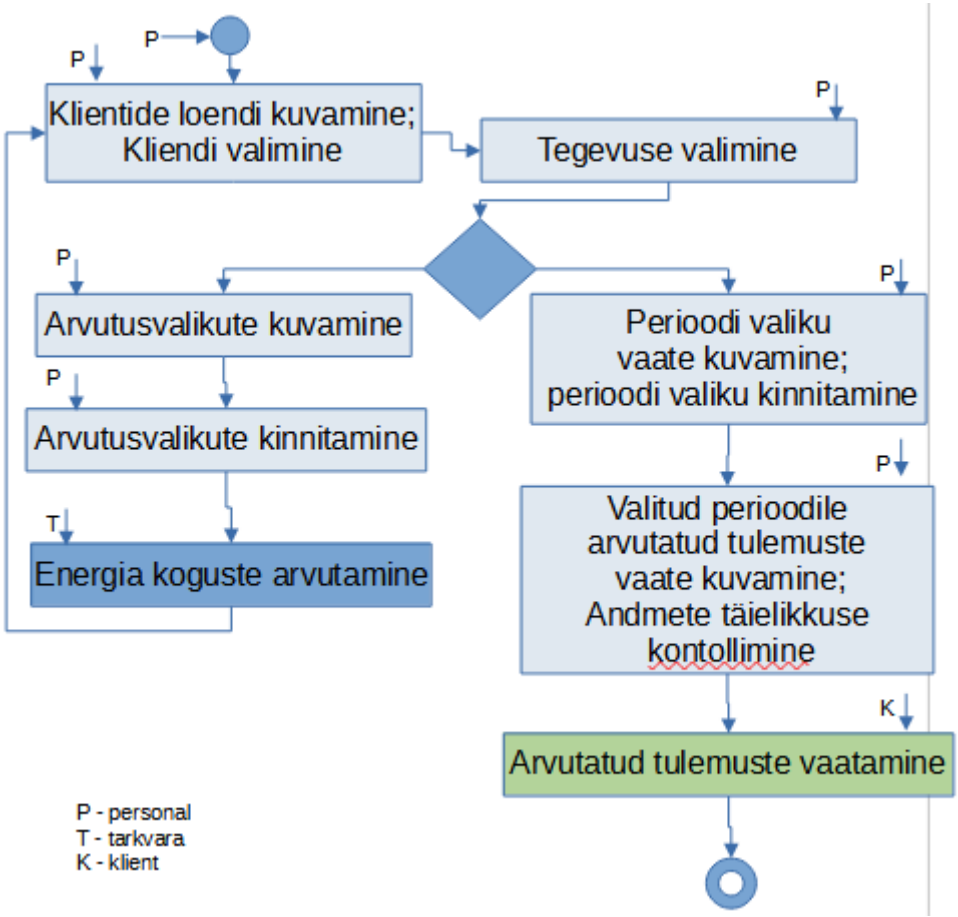
- Vormistada elektri ostmise leping, vaadata kehtivaid lepinguid
- Vaadata enda elektritarbimise arvutustulemusi

Elektrijaama personal saab:

- Defineerida müügiks oleva elektripaketid ja määrata hinna
- Vaadata elektri ostuavaldusi, ja kinnitada ostu-müügilepingud
- Arvutada, kontrollida ja korrigeerida kõikide klientide ostetud energiakoguseid
- Modifitseerida kliendile kasutamiseks olevaid vorme

Järgnevalt kirjeldatakse olulisemaid funktsioone põhjalikumalt

Funktsionaalsus: klientidele müüdud energiakoguste arveldamine. Järgneval joonisel (joonis 12) on näidatud energiakoguste arvutamise protsess tarkvaraarendaja vaatenurgast lähtudes.



Joonis 12. Protsessivoo diagramm energiakoguste arvutamiseks

Protsessivoo UML diagrammi kirjeldus:

Algseis: Arvestinäidud on andmebaasis iga tunni mõõtmistena. Personalitöötaja asub personalitöötajale mõeldud esilehel, kui ta on sisse loginud.

Personalitöötaja: Vajutab linki „Klient", et liikuda vaatesse, kus on kõigi klientide loend. Kõigi klientide jaoks saab personalitöötaja teha ka tegevusi.

Personalitöötaja: Valib selle kliendi, kelle jaoks ta soovib kas energiakoguseid arvutada või arvutustulemusi vaadata ja parandada.

Personalitöötaja: valib tegevuse. Tegevusteks võib olla kas kliendile mingi perioodi energia koguste arvutamine või juba välja arvutatud tulemuste vaatamine/parandamine. Energiakoguste arvutamiseks valitakse arvelduspериood loendist, näiteks „Mai 2022".

Tarkvara: Arvutab sellel perioodil iga tunni kohta müüdud elektri ja defineeritud perioodi kohta kokku. Periood saab endale nime, näiteks „Mai 2022". Andmed salvestatakse tabelisse.

Personalitöötaja: Liigub sellesse vaatesse, kus ta saab valida, millise isiku millise perioodi arvutustulemust kontrollida (Perioodi valiku vaate kuvamine). Perioodi valiku kuvamise vaatesse liigutakse klientide loendi vaatest. Ühest vaatest teise liikumise taustal teostab tarkvara ka vajaliku info hankimise, selle jaoks on eraldi meetod tarkvara kontrolleris.

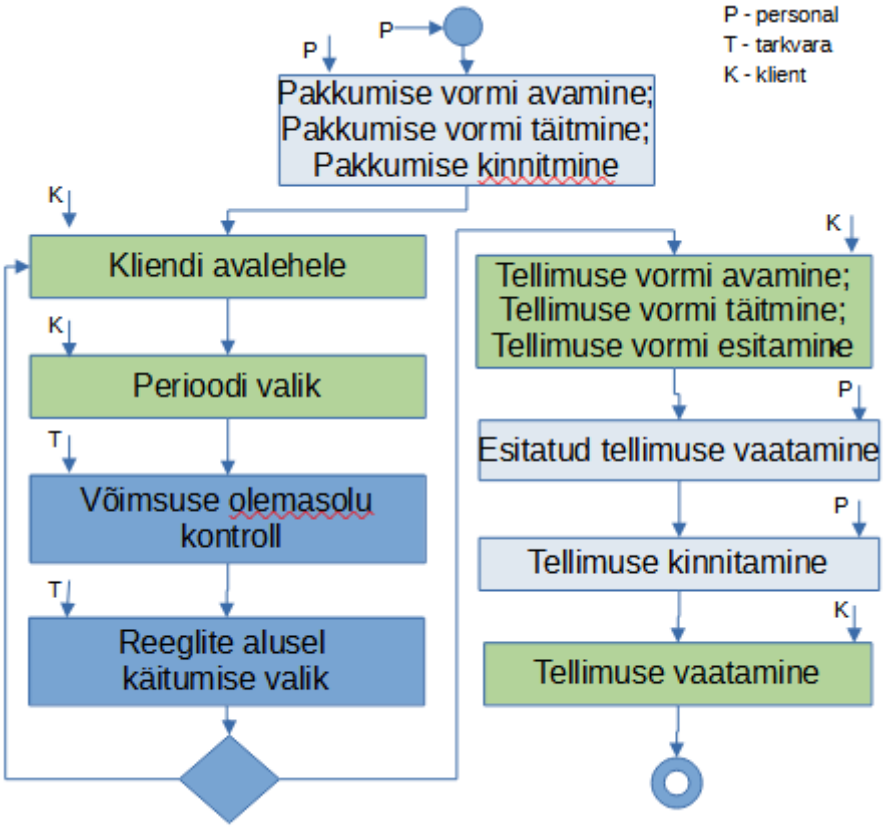
Personalitöötaja: kinnitab valiku, milleks on see periood, mille arvutust soovitakse kontrollida. Näiteks 2022 Jaanuar. Sellesse sammu on sisse arvestatud nii valiku tegemine, kui ka kinnitamise nupule vajutamine. Kui vajutatakse kinnitamise nupule, siis käivitatakse vastav meetod tarkvara kontrolleriosas, mis kogub vaatelehe valiku andmed ja vastavalt nendele andmetele teostatakse päringud andmebaasist ja kuvatakse järgmine vaade eeltäidetuna.

Personalitöötaja: On liikunud arvutatud tulemuste vaatesse vastavalt valkitud kliendile ja perioodile. Saab kontrollida ja korrigeerida igale perioodiühikule arvutatud tulemusi. Vaadetesse liikumise tegevused, vaadetes valikute tegemised ja kinnitamised on näidatud eraldi sammudena, sest sammud on seotud arendaja jaoks eraldi meetoditega. Meetodite tegevused ja vaadete esitamised on vajalik omavahel kokku sobitada. Nii on eraldi etappidena näidatud perioodi valiku vaate kuvamine ja perioodi valiku kinnitamine, sest esimese jaoks tegutseb üks meetod ja teise jaoks teine meetod. Samuti toimuvad kaks tegevust eraldi vaadetes. Perioodi valiku kinnitamine on eristatud arvutatud tulemuste vaatest, sest etapid toimuvad erinevates vaadetes, ja ühest vaatest teise liikumisel tegutseb ka meetod, mis esimese vaate valikuid arvestades valmistab andmed ette järgmise vaate jaoks. Arvutatud tulemuste vaate kuvamine on eraldatud arvutuste kontrollimisest, sest vaatesse liikumiseks kasutatakse meetodi abi. Arvutuste kontrollimine on eraldi etapina, sest kontrollimine eeldab inimese sekkumist.

Kontrollimise lehel avanevad ka tegevused, et mõnda viga parandada. Andmete täielikkuse kontrolli vaade aitab tuvastada peamiselt neid vigu, mis tuleneb arvestinäitude lünklikkusest andmebaasis. Lisaks aitab ta tuvastada vigased lõpptulemused ka mistahes muust põhjusest tulenevana. Näiteks vale hinna sisestus. Selles faasis näeb personalitöötaja viimast pilti enne selle kliendile näitamist ja mille alusel kliendile esitatakse ka arve. Põhiline viga võib tuleneda siiski sellest, et kaugloetavate arvestite kõik mõõteandmed ei saabu. Põhjus võib olla ka selles, et andmebaasiga ühendus katkes arvutamise ajal. Viimane vaade enne kliendile näitamist näitab need kõik ära.

Klient: Näeb talle välja arvutatud perioodide nimede loendit, ja perioodi nimele klikkides näidatakse talle perioodi arvutuse tulemust.

Funktsionaalsus: kliendi tellimuse esitamine. Järgneval joonisel (13) esitatakse protsessivoo diagramm kliendiga lepingu sõlmimise kohta.



Joonis 13. Protsessivoo diagramm tellimuse esitamiseks

Kui interneti kaudu klient tahab esitada tellimust, siis interneti kaudu tellimus esitatakse kliendi poolt ja kinnitatakse personalitöötaja poolt. Sisuliselt on see lepingu tegemine.

„Võimsuste olemasolu kontrolli" samm: enne tellimuse vormi avamist programm vaatab üle jaama vaba võimsuse olemasolu. Näiteks kui keegi soovib esitada tellimust perioodiks Jaanuar 2022, siis programm vaatab üle, kui palju on juba tellimuste võimsuste kogusumma, ja võrdleb seda sellega, mida elektrijaam suudab pakkuda. See on vajalik elektrijaama optimaalse tehnilise koormatuse tagamiseks.

Kui elektrijaamal uute tellimuste vastuvõtmiseks võimsust ei piisa, siis järgnevad tegevused sõltuvad hetkel kasutatavatest elektrituru reeglitest.

Vormi avamine ja vormi täitmine on eraldi etappides, sest konkreetne voodiagramm on esitatud tarkvara arendaja seisukohalt. Tarkvara arendajal on kergem teostada etapiliselt võimalikult tükeldatud protsessi.

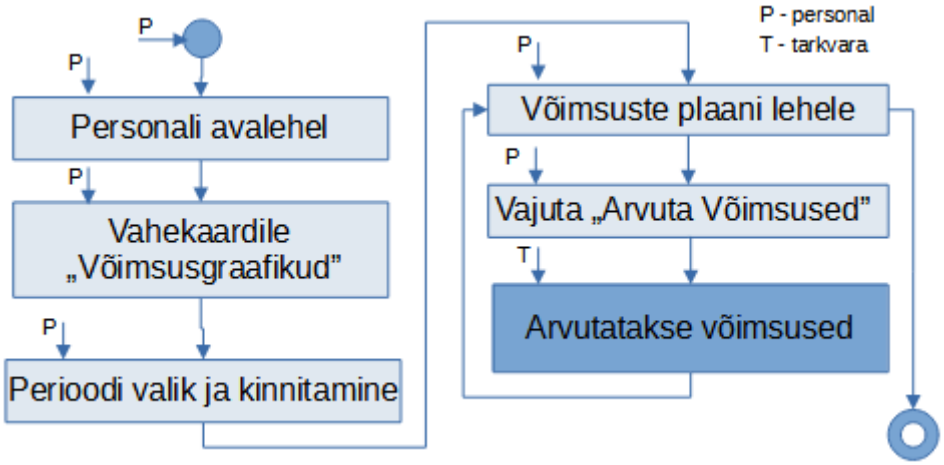
Tellimuse kinnitamine: Kinnitamisega muudetakse ära tellimuse staatus. Staatus võib olla „esitatud", „töötluses", „kinnitatud" või „tagasi lükatud".

Protsessi viimaseks sammuks on „Tellimuse vaatamine". Seda saab klient teha mistahes ajal. Tellimuse vaatamisel näeb ta ka oma tellimuse staatust. Tellimuse staatust klient ise muuta ei saa. Küll on võimalik aga kliendil tellimust muuta ja kustutada. Kui klient juba kinnitatud tellimust muudab, siis tarkvara muudab ära ka tellimuse staatuse algolekusse.

Funktsionaalsus: Pakutavate võimsuste tabelid. Elektrijaama töötajatest keegi peaks saama sisestada perioodide maksimumvõimsusi, mille põhjal arvutatakse välja pakutavate võimsuste tabelid. Pakutavate võimsuste arvutamise protsessi diagramm esitatakse joonisel 14.

Tellitud võimsused mingiks perioodiks arvutatakse kui kõigi selle perioodi lepinguliste klientide maksimumide summa, korrutades üheaegsusteguriga. Saadud võimsus on maksimaalne prognoositud töövõimsus sellel perioodil. Maksimumvõimsused näidatakse olemis „Arvesti". See on sisuliselt selle liitumispunkti pealüliti võimsus , kus arvesti asub.

Võimsused igaks üksiktunniks saadakse, korrutades plaanilist maksimumi iga tunni teguriga. Nii saadakse töövõimsused tundide lõikes.



Joonis 14. Protsessivoo diagramm võimsuste haldamiseks

5.3 Kliendi kasutajaliides

Järgnevalt kirjeldatakse vaateid, mida klient näeb.

Veebi avalik vaade: Kui soovitakse siseneda, siis avavaatest liigutakse sisenemise vaatesse, vajutades linki „Sisene”.

Sisenemise vaade: Sisenemiseks sisestatakse kasutajatunnused: kasutajanimi ja salasõna. Vajutades nuppu „Sisene”, käivitub autentimise protseduur, mille käigus tuvastatakse isiku õigus siseneda ja sisenenud isiku roll. Sisenenud kasutaja avavaade ja edasised võimalused sõltuvad sisenenud isiku rollist.

Sisenenud kliendi avavaade: Avavaates on sisenenud kliendil kaks põhilist edasi navigeerimise võimalust. Vajutades linki „Tellimused”, kuvatakse kliendile tema tellimuste loend. Tellimuste loendi vaates saab klient ka uusi tellimusi esitada.

Vajutades linki „Tarbimine”, saab klient vaadata temaga seotud arvesti mõõtetulemuste põhjal arvutatud tarbitud elektrienergia koguseid koos maksumusega. Elektroonses tellimuses kokku lepitud paketis sisalduva hinna alusel ja arvestite mõõtetulemuste põhjal arvutatakse arved kliendile esitamiseks.

Võimalus on ka rakendusest väljuda. Kliendi jaoks on ka teadete tahvel, kuhu edastatakse mitmesuguseid teateid veebirakenduse poolt. Näiteks et „Tellimuse esitamine oli edukas".

Tellimuste vaade: Tellimuste vaates saab klient vaadata, milliseid tellimusi ta on esitanud. Tellimuse vaates näeb klient, millisele pakkumisele ta on tellimuse esitanud. Pakkumine esitatakse kliendi jaoks nii, et seal oleks asjakohane info võimalikult arusaadavalt esitatud. Seetõttu pakkumise väli modifitseeritakse täiendava tööna. Pakkumise väljas sisaldub ka kokku lepitud hind. Tellimuse vaatest saab klient liikuda edasi uue tellimuse esitamise vaatesse. Perioodi andmetüübina kasutatakse andmebaasis „VARCHAR'i" ja Java keskkonnas „String'i". Töötlustes perioodi nime järgi ei järjestata. Kliendi tellimuste vaade esitatakse lisas skriptide peatükis (Lisa 4 joonis 19).

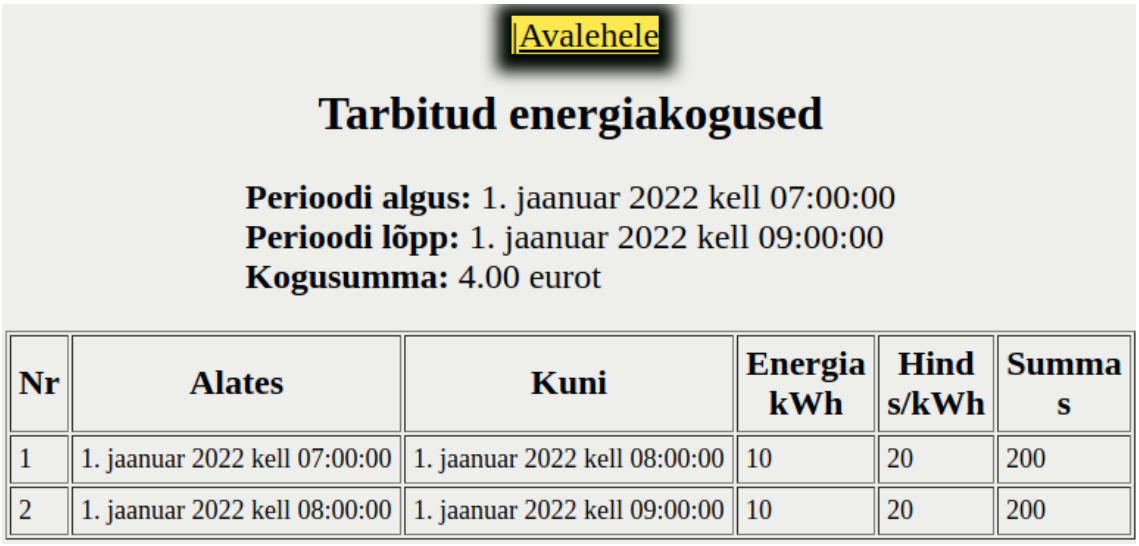
Perioodi valiku vaade tellimuse esitamiseks: Uue tellimuse esitamise vaatesse liigutakse läbi perioodi valiku vaate. Kõik tegevused on seotud mõne perioodiga. Perioodi valiku vaates valitakse ja kinnitatakse periood, mille jaoks soovitakse tellimust esitada. Kui pakkumist ei ole, siis tellimust esitada ei saa selle perioodi kohta ... siis suunatakse klient avavaatele ja edastatakse vastav teade, näiteks „Perioodiks „x" pakkumisi ei ole". Vaata joonist lisas skriptide peatükis (Lisa 4 joonis 21).

Uue tellimuse esitamise vaade: Tellimuse esitamiseks teostatakse valikud menüüdest ja vajutatakse nuppu „Esita tellimus". Tellimuse esitamise vaates on eeltäidetud periood, mis valiti eelnevas etapis. Lisaks on eeltäidetud selle kliendi nime lahter, kes on hetkel sisse loginud. Valikuvõimalused on sisestuslahtritel „Pakkumine" ka „Arvesti". Pakkumise loendis näidatakse hetkel aktiivseid pakkumisi vastava perioodi kohta. Näiteks maikuus on pakkumine nimega „stabiilne". Pakkumine on sisuliselt mingi pakett, mis valiku loendis esitatakse nimega. Ühe kliendiga võib olla seotud mitu arvestit. Näiteks korter ja suvila, siis ta saab sama rakenduse kaudu teha oma mõlema arvesti lepinguid. Kui tellimuse esitamiseks vajutatakse nuppu „Esita tellimus", siis tellimus registreeritakse süsteemis ja sellega hakkavad tegelema elektrijaama spetsialistid.

Tellimuse muutmise vaade: Vaata lisa skripide peatükis (Lisa 4 joonis 20)

Perioodi valiku vaade tarbimisandmete vaatamiseks: Kui klient soovib vaadata tarbitud energia koguseid, vajutatakse „Tarbitud energia". Enne energia koguste vaatamist suunatakse klient perioodi valimise vaatesse. Arvelduseriood on üldjuhul üks kuu. Perioodi valik tehakse rippmenüüst vastava kuu nime alusel, näiteks „Jaanuar 2022".

Tarbitud energiakoguste vaade: Tähtsaimaks kliendi vaateks on tema jaoks välja arvutatud energiakogused (järgnev joonis 15). Vaates on nähtavad nii detailandmed iga tunni kohta sellel perioodil kui ka perioodi koondtulemus vahetult enne tabelit. Koondtulemuse kõige olulisem number on summa, mis on kliendile maksmiseks mõeldud. Alltoodud joonisel on alamperioodi summa ja summa kokku antud sentides. Eurode esitamiseks kahe komakohaga peale koma kasutatakse vaate HTML mudelis JavaScript'i [24] . Elektrienergia hinna ühikuks on senti kilovatt-tunni kohta.



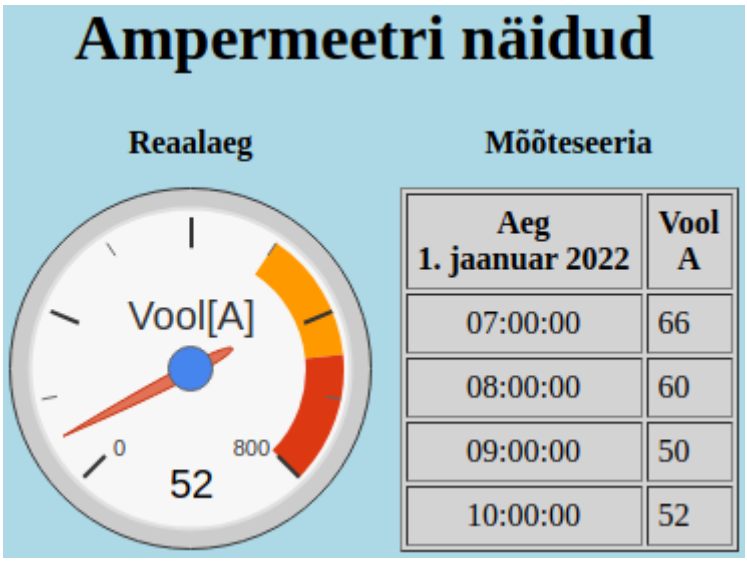
Joonis 15. Tarbitud energiakogused.

Kuupäevad koos kellaaegadega on vormindatud tavainimesele võimalikult loetaval kujul Eesti stiilis. Vormindus teostatakse koodis vaate mudelis. Teisendus on tehtud Java andmetüübist „Timestamp", mis omakorda MySQL andmebaasis hoitakse andmetüübiga „DateTime". Vaate HTML mudelis kasutatakse „setLocale" atribuuti, et kasutaja vaates kujundada kuupäeva stiili vastavalt riigi keelenõuetele. „setLocale" atribuudiga muudetakse kuupäev lõplikus vaates riigi eripärale vastavaks. Eesti „setLocale" väärtuseks on „et_EE" [25] .

5.4 Spetsialisti kasutajaliides

Käesolevas alapeatükis esitatakse elektrijaama töötaja rollis oleva isiku jaoks ettenähtud vaateid.

Mõõdikute paneel: Mõõdikute paneeli vaates teostatakse mõnede mõõteriistade reaalajalised näidud. Alltoodud joonisel 16 on mõõdikute paneeli esmane vaade. Käesolevas töös pole kaardistatud kõikide vajalike mõõteriistade esitamise vajadust. Esitatakse piiratud kogus tähtsamaid mõõteriistu hilisema laiendamise võimalusega. Mõõdikute paneel on ette nähtud elektrijaama spetsialistile vaatamiseks. Brauseris saab andmete põhjal genereerida ka mõõteriista [26] . Selle jaoks on olemas „Google Charts" graafikute teek. Mõõteriista graafikus esitatakse kõige uuemat reaalajalist näitu. Mõõteseeria tabelis esitatakse ka varasemad näidud. Mõõteseeria tabelis esitatakse tavaliselt veelgi rohkem näite. Tabeli päises esitatakse kuupäev, mis on ühine kõigi kellaaegade kohta tabelis.



Joonis 16. Mõõdikute paneeli vaade

Pakutavate võimsuste tabel: Mitmete eelnevalt arvutatud ja andmebaasi kogutud tulemuste põhjal, kui teatakse klientide poolt esitatud tellimusi, arvutatakse välja elektrijaama tööplaan. Näitlik väljavõte tööplaanist esitatakse joonisel 17. See tähendab, et kui teatakse järgneva perioodi tellimuste infot ja kui see tellimuste periood on käes ja tellimused on „lukus" selleks perioodiks, siis saab arvutada kui palju on kliente kokku ja kui palju on klientide koguvõimsus. Siis saab elektrijaama töögraafiku teha selle põhjal. Töögraafik näitab: millisel tunnil mingit võimsust hakatakse välja andma. Välja antav võimsus päeva lõikes muutub. Kui tarbimine on suurem, siis annab rohkem võimsust välja. Elektritootjatel käib see tunni kaupa. Samuti kasutatakse tunniseid vahemikke elektri müümisel ja elektri börside töös. Näiteks kell 06.00 algab väljundvõimsus 3 kW, mida hoitakse järgmise tunni alguseni ja kell 07.00 seatakse jaama väljundvõimsuseks 5 kW. Võimsuste plaani ühes tabelis esitatakse tavaliselt 24 tunni andmed, siinkohal on vaadet lühendatud.

Elektrijaama võimsuste plaan Jaanuar 2022

Tund	Alates	Pakutav võimsus kW	Töövõimsus kW	Arvuta Võimsused
1	00:00:00	42	3	
2	01:00:00	42	3	
3	02:00:00	42	3	
4	03:00:00	42	3	
5	04:00:00	42	3	
6	05:00:00	42	3	
7	06:00:00	42	3	
8	07:00:00	54	5	
9	08:00:00	54	5	
10	09:00:00	60	5	
11	10:00:00	60	5	
12	11:00:00	60	5	

Joonis 17. Väljavõte võimsuste plaanist

6 Tulemuste analüüs ja tulevased tegevused

Algne ülesanne versus teostus: Käesoleva töö eesmärgiks oli luua funktsionaalsust, mis toetaks salvestuselektrijaama tegevust. Võimalikke arendussuundi oli kaks: kas äriline, millega toetataks elektrijaama elektriturul osalemist või elektrotehniline pool, mis hõlmaks reaalajas jälgimist ja tehnilist elektrijaama talitlemist. Käesolevas töös on põhirõhk elektriturul osalemisel, sest see haakub senise IT Kolleži väljaõppega paremini. Käesoleva töö elektripoole taustauuringutega on tegeletud piisavalt, kuid pole uuritud kõike informatsiooni välja. Elektriosa on üldisemas käsitluses. Lähenetud on tarkvara arendamise poolelt. Elektriosas on detaile äärmiselt palju, mida kõike saaks välja uurida. Kui kõike hakata välja uurima, siis tarkvara arenduseni ei jõuakski. Töö rõhk on tarkvara arendusel ja kõigi detailide välja uurimine ei mahu käesoleva töö raamidesse. Niisugune käitumine on kooskõlas ka Kanban metoodikaga, mille kohaselt alustatakse sellest, mida hästi tuntakse. Kuna käesoleva töö autor midagi elektrist juba teab, siis seda midagi hakati kohe tarkvaraliselt arendama. Kanban tarkvara arenduse metoodikaga lahendatakse mitmeid küsimusi. Idee on, et mitte kõike kohe välja uurima hakata, vaid alustada teada olevatest asjadest. Arendatavat tarkvara täiendatakse tagasiside põhjal.

Vahendite ja meetodite sobivus: jaama elektriturul osalemise tarkvara arendamiseks kasutati Java Eclipse arenduskeskkonda. Võimalikud alternatiivid olid C# Rider keskkonnas, Java IntelliJ keskkonnas. Autori valik oli Java, sest autor oli kõige kogenum just Java-s. Veebirakenduste disainimiseks, IT Kolledži stuudiumi jooksul, olid kõige paremad kogemused Java Spring MVC-ga. IT Kolledžis kasutati Java-s veebirakenduste tegemiseks IntelliJ keskkonda.

Üheks alternatiiviks Spring MVC raamistikule on Spring Boot raamistik. Arendamine töövahendiga Spring Boot. Spring Boot'i kasutamiseks lisatakse Eclipse'ile töövahend „Spring Tool Suite" (vaata lisa 2). Hea metoodiline abivahend: Nam Ha Minh allikas [27] . Praegu võib Spring Boot'i pidada moodsamaks. Käesoleva töö käigus Spring Boot'i prooviti. Võimalik et Spring Boot'iga arendusprotsess on kiirem. Käesolevas

töös osutus efektiivsemaks siiski Spring MVC, seda ilmselt autori eelneva vilumuse tõttu. Iga raamistik ja töövahend vajavad ka kohanemist, mistõttu uute vahendite paremus võib selguda pärast mõnda kuud harjutamist. Alternatiiviks MVC arhitektuurile on REST.

Käesoleva töö praktilist arendamist alustati töökeskkonna seadistamisega, sest arendusvahendid olid lähteülesandes kirjeldatud. Seejärel kasutati meetodeid andmebaasi struktuuri ja äriloogika disainimiseks. Andmebaasi koostamisel kasutatai olemi-suhte diagrammi, äriloogika jaoks IDEF0 protsessiloogika meetodit. Mõlemat korrigeeriti programmeerimise vahetulemustest järelduste tegemise abil.

Äriloogika arendamise meetod valiti käesoleva töö arendamise käigus. See on IDEF0 metoodika. Alguses prooviti tavalist UML protsessivoogu teha. Kirjanduse läbitöötamisel avastati IDEF0 metoodika, mille kasutamine käesolevas töös osutus edukaks.

Turbe osa arendusvahendite valimine teostati käesoleva töö tegemise käigus, seda polnud alguses teada, millega lahendada. Turbe lahendamiseks otsustati kasutada töövahendit „Spring Security". Hetkel peaks „Spring Security" olema kõige täiuslikum arendusvahend Java veebirakenduste turbe arendamiseks. Käesolevas töös tegeletakse „Spring Security" juurutamisega Eclipse keskkonnas. „Spring Security" on rohkete võimalustega ja kasutusele võtmine on keeruline. IT Kolledžis natuke „Spring Security't" õpetatakse, kuid keskkonnana kasutatakse „IntelliJ'd" ja tarkvara arhitektuurina REST-i MVC asemel. Käesolevas töös seotakse „Spring Security't" Eclipse arenduskeskkonnas „Spring MVC" arendusmustriga. Turbe osa on „Spring Security'l" olemas, aga seda tuleb osata kasutada. Vastava vilumuse tekkimine nõuab visa harjutamist. Metoodika pole ilma sisse elamiseta hallatav. Leitud on ka üks näide Internetist „Spring Security" kasutamiseks Eclipse keskkonnas Spring MVC raamistikus. „Spring Security'l" on olemas vajalikud teegid mitmete turbeseadete jaoks. Näiteks sisse logimise vormi näitamine. Nüansside rohkus on suur ja kõikide arendusvahenditega on omakorda natuke erinev. Kusagilt pole leitud täielikku näidist, mille analoogia põhjal kõik ära teha. Kas on turbe osa tehtud ja töö esimest osa pole või on töö esimeses osas midagi tehtud ja turbe osa puudub.

C# programmerimiskeele raamistikes on turbeosa juba alguses sisse ehitatud ja tarkvara arendaja ei pea turbe arendamisega nii palju tegelema kui Java raamistikes. C# raamistikes on turve lihtsamini kasutusele võetav. Java vahendite jaoks kasutatakse „Spring Security't", mis turbe jaoks vajalikke komponente küll sisaldab, kuid nende kasutusele võtmine on keerulisem kui C# vahenditega. Kergem turbe elementide kasutuselevõtt tähendab seda, et teadmised turbe olemasolust ja olemusest jäävad ka pinnapealsemaks. Käesolevas töös kasutati Java-t, millega „Spring Security" turbe kasutusele võtmiseks oli vajalik põhjalikult süveneda, mis turbe õppimise vaatenurgast oli positiivne. Süvateadmised veebirakenduse turbest on tähtsad, sest tarkvara arendaja peaks suutma ka tõestada, et veebirakendus on rünnete vastu kaitstud.

Töös tekkinud probleemid: Selle rakenduse praktilise ehitamisega alustati 4. veebruaril 2022. Esimeste pingutustena olid vajaliku tarkvara paigaldus ja arenduse keskkonnaga harjumine. Arenduskeskkond oli üldiselt juba tuttav, aga vaja oli uuesti paigaldada ja harjuda. Programmeerimisel kasutati mitmeid oskusi, mis ei ole iseenesest tekkivad. Järgnevalt kirjeldatakse mitmeid oskusi, mida tuli täiendavalt meelde tuletada, juurde õppida, või treenida, et vajalikud funktsionaalsused teostada.

Pärast esimest arendamise kuud tarkvara algeesmärkides loetletud baasfunktsionaalsus suures osa töötab. Olemasolevaid kasutajaliideseid on vaja veel lihvida. Funktsionaalsuses on ka mitmeid puudujääke: näiteks üks klient saab teise eest tellimusi esitada. Siinkohal pole tegemist arendamise veaga, vaid vajadusega olemasolevat tarkvara edasi arendada.

Raskemaid kohti tarkvara arenduses:

- DateTime andmete päring MySQL andmebaasist Eclipse keskkonda.
- Mudeli atribuutide saatmine läbi mitme vaate.
- Elemendi tingimuslik peitmine vaates
- Rakenduse turvanõrkuste kõrvaldamine

DateTime kasutamise kohta saadi lisainfot allikast [21] . Käesoleva töö spetsiifikast tulenevalt kasutatakse DateTime andmetüüpi väga palju, sest arvestite näite kogutakse

iga tunni kohta ja on vaja näitude põhjal teha arvutusi. Seetõttu on ajalise andmetüübiga seotud töötlust ja päringute koostamist väga palju.

Tulevased tegevused: tulevikus on võimalikud mitmesugused arengud, kus käesoleva töö saavutusi saaks juurutada. Käesoleva tarkvara arendamise käigus selgusid mitmed teemad, mida käesoleva töö raames pole jõutud teha. Ühe bakalaureuse mahulise projektiga ei jõuta tervet elektrijaama ära hallata. Käesoleva töö raames tehtud rakenduse andmebaasis on 17 olemit, mis on bakalaureusetöö mahu jaoks sobiv. Edasistes tegevustes saaks lisada funktsionaalsust, mis laiendaks ka andmebaasi. Järgmisi edasi arendamise suundi: reaalajalise bilansi hoidmine, mõõteandmete kogumine, võimsusgraafikute täpsustamine statistika põhjal, graafiliste esitusviiside täiustamine, turvanõrkuste vähendamine, kasumlikkuse hindamine ja maksimeerimine. Arvestite mõõteandmete täielikkuse kontrollimiseks saaks teha graafikuid. Kui on näha, et kuskil päeva jooksul on oluline mõõteandmete lünklikkus, siis reeglina pole mõned andmed sidesüsteemi kaudu andmebaasi kohale jõudnud.

Mõõteriistade paneelil saaks esitada veelgi rohkemate mõõteriistade näite. Olemas olevat näitude tabelit saaks täiendada, kui esitada näite veelgi sagedasemate intervallidega. Samuti saaks juurutada, kui tabelis esineb kuupäeva vahetus. Niisugusel juhul mõned mõõteandmed on vana kuupäevaga ja mõned uue kuupäevaga ehk esineb ühelt kuupäevalt teisele üleminek. Näiteks saaks siis päises esitada kaks kuupäeva. Üheks lisaindikaatoriks näidikute paneelil ja täiendavaks funktsionaalsuseks oleks jaama bilansi hoidmise reaalajaline jälgimine ja juhtimine.

Peamiseks edasi arendamise pinnaseks käesolevas töös on pakutavate võimsuste tabelid. Järgmise päeva võimsuste tabelit peaks hakkama ennem nägema, kui üks päris ära lõpeb.

Arvutada võiks salvestuselektrijaama majanduslikku tasuvust, ehitada füüsiline või virtuaalne elektrienergia salvestussüsteem, et selgitada kas ja kui palju salvestamise kaudu raha säästetakse. Arendada ka tehnilist salvestamise tarkvara.

Haakuvus tulevikutrendidega, märksõnu: elektriautode osalemine elektriturul, „rail baltic” rongi osalemine elektriturul. Käesolevas töös arendati tarkvara ühele kindlale elektrijaama tüübile: salvestuselektrijaam, jaama tüübiks liitium-ioonjaam. Tegelikult on niisugust tarkvara võimalik kasutada ka mitmete teist tüüpi elektrijaamade jaoks.

7 Kokkuvõte

Käesolevas bakalaureusetöös käsitletakse planeeritava elektrijaama tarkvara. Tarkvara arendamiseks kasutati Java Eclipse EE arenduskeskkonda. Andmebaasina kasutatakse lokaalarvutisse paigaldatud MySQL'i andmebaasi. Serverina kasutatakse Apache TomCat'i serverit lokaalarvutis. Arenduseks kasutatakse tavalist sülearvutit operatsioonisüsteemiga Linux Ubuntu 20.04.

Tarkvara arendatakse nii elektrijaama äritegevuse kui ka elektrotehniliselt vajalike toimingute toetamiseks. Suurem rõhuasetus on äritegevuse tarkvaral, sest see haakub IT Kolledži väljaõppega rohkem. Äritegevuses võimaldatakse klientidel osaleda veebirakenduse vahendusel. Veebirakenduste tarkvara ongi kõige otsesem IT süsteemide arenduse eriala väljund.

Äritegevuse tarkvara hõlmab:

- Viieteistkümne olemiga andmebaas kliendihalduse jaoks
- Rakendustarkvara klientide elektriostu tellimuste haldamiseks.
- Arvestite näitude põhjal arvutatakse tegelikult müüdud elektrienergia kogused.
- Kasutajaliides kliendile, kes saab veebibrauseri kaudu vormistada elektriostu tellimust, vaadata oma tarbimise andmeid ja seda, kui palju ta peab tasuma.

Iga eesmärgiks olnud komponenti jõuti töös ka käsitleda. Äritegevust toetavat tarkvaraosa jõuti kaugemale arendada. Eesmärgid täideti suures ulatuses, kuid täiendamise ruumi jäi ka. Elektrotehnilise tarkvara arendamisel jõuti teostada minimaalne mõõdikute paneel ja mõõteandmete ajaloo minimaalne päringusüsteem.

Märkimisväärselt tegeletakse ka vajaliku turbetaseme saavutamisega. Turbe arendamise keerukus pole käesoleva lõputöö teoreetilises osas nähtav, praktilise osa teostamisel oli see siiski üsna aeganõudev.

Kasutatud kirjandus

[1] Elering. Elektrituru käsiraamat 2016. https://elering.ee/sites/default/files/attachments/elering_elektrituru_kasiraamat_2016_web_1.pdf (29.03.2022)

[2] Elektrilevi. Tüüpkoormusgraafik. https://www.elektrilevi.ee/ettevottest/elektriturg?tabgroup_1=electricity_market (30.03.2022)

[3] Ali Gulabi. Development of an Embedded SCADA System with PLC and Java. Application for Synchronous Operation of Standard Servo Drives. Hamburg. 2007. https://reposit.haw-hamburg.de/bitstream/20.500.12738/8639/1/Master_Thesi s.pdf (17.02.2022)

[4] Energy Global. Software: an anchor amidst change. https://www.energyglobal.com/special-reports/22032022/software-anchoring-a-changing-energy-storage-landscape/ (07.05.2022)

[5] Wärtsilä. GEMS IntelliBidder. https://storage.wartsila.com/technology/gems/ (07.05.2022)

[6] Wärtsilä. Software Solution Engineer. https://careers.wartsila.com/job/Herndon-Software-Solution-Engineer-VA-20170/781969001/ (07.05.2022)

[7] Origami. Trading and Automation. https://www.origamienergy.com/trading (07.05.2022)

[8] Solarplaza. Realising Storage's Potential. https://www.solarplaza.com/resource/12205/realising-storages-potential/ (07.05.2022)

[9] Software Testing Help. Linux Vs Windows Difference: Which Is The Best Operating System? https://www.softwaretestinghelp.com/linux-vs-windows/ (19.04.2022)

[10] TrustRadius. Eclipse vs IntelliJ Idea vs Rider. https://www.trustradius.com/compare-products/eclipse-vs-intellij-idea-vs-jetbrains-rider (06.05.2022)

[11] Tabnine. IntelliJ IDEA vs Eclipse: Which is Better for Beginners. https://www.tabnine.com/blog/intellij-idea-vs-eclipse/ (06.05.2022)

[12] JetBrains. Rider System Requirements. https://www.jetbrains.com/dotnet/download/system-requirements/ (06.05.2022)

[13] Ram Kulkarni. Java EE8 Development with Eclipse – Third Edition. 2018. https://subscription.packtpub.com/book/application-development/9781788833776/1

[14] Urmas Öövel. Report of the Internship. 2020.

[15] Laurentiu Spilca.Spring Security in Action. Manning Publications. 2020. https://learning.oreilly.com/library/view/spring-security-in/9781617297731/OEBPS/

Text/12.htm#heading_id_3; https://dokumen.pub/qdownload/spring-security-in-action-1nbsped-1617297739-9781617297731.html

[16] AKIT – Andmekaitse ja infoturbe leksikon. https://akit.cyber.ee/ (07.03.2022)

[17] Carlos Polop. Hack Tricks. https://book.hacktricks.xyz/pentesting-web/xpath-injection (10.03.2022)

[18] Peter Mularien. Spring Security 3. Ühendkuningriik. 2010.

[19] Nam Ha Minh. Spring Web MVC Security Basic Example Part 2 with Java-based Configuration. https://www.codejava.net/frameworks/spring/spring-web-mvc-security-basic-example-part-2-with-java-based-configuration (07.04.2022)

[20] Dariawan. java.sql.Timestamp Examples. https://www.dariawan.com/tutorials/java/java-sql-timestamp-examples/_ (06.04.2022)

[21] W3Schools Online Web Tutorial. SQL Between Operator. https://www.w3schools.com/sql/sql_between.asp (08.03.2022)

[22] Thomas O.Boucher, Ali Yalcin. Design of Industrial Information Systems. 2006. https://www.sciencedirect.com/book/9780123704924/design-of-industrial-information-systems

[23] Codeacademy. What is REST? https://www.codecademy.com/articles/what-is-rest (12.02.2022)

[24] W3Schools. JavaScript Number toFixed(). https://www.w3schools.com/jsref/jsref_tofixed.asp (30.03.2022)

[25] LocalePlanet. ICU Locale “eesti (Eesti)” (et_EE) https://www.localeplanet.com/icu/et-EE/index.html (23.03.2022)

[26] Google Charts. Visualization: Gauge. https://developers.google.com/chart/interactive/docs/gallery/gauge (03.04.2022)

[27] Nam Ha Minh. Spring Boot Registration and Login with MySQL Database, Bootstrap and HTML5. https://www.youtube.com/watch?v=aRLoSDOlU3w (06.04.2022)

Lisadest viited:

[28] Canonical Ltd. How to install Eclipse EE on ubuntu? https://askubuntu.com/questions/1363850/how-to-install-eclipse-ee-on-ubuntu/ (08.02.2022)

[29] Nam Ha Minh. Java Spring MVC and JDBC CRUD Tutorial. https://www.youtube.com/watch?v=-wddkBQZuPs (09.02.2022)

[30] Stack Overflow. Ubuntu MySQL Uninstall / Reinstall. https://stackoverflow.com/questions/28687787/ubuntu-MySQL-uninstall-reinstall (08.02.2022)

[31] Computing for Geeks. How To Install MySQL 8.0 on Ubuntu 20.04|22.04. https://computingforgeeks.com/how-to-install-MySQL-8-on-ubuntu// (09.02.2022)

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

Mina, Urmas Öövel

1 Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Salvestuselektrijaama tarkvara arendamine", mille juhendaja on Kaido Kikkas

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.

09.05.2022

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

Lisa 2 – Arenduskeskkonna seadistamine

Eclipse EE paigaldus [28] :

```
$ sudo add-apt-repository ppa:lyzardking/ubuntu-make
$ sudo apt-get update
$ sudo apt-get install ubuntu-make
$ umake ide eclipse-jee
```

Apache paigaldus:

https://crunchify.com/step-by-step-guide-to-setup-and-install-apache-tomcat-server-in-eclipse-development-environment-ide/

https://tomcat.apache.org/download-90.cgi

https://tomcat.apache.org/tomcat-9.0-doc/building.html

Eclipse EE, töölaua seadistus [29] :

Töölaua seadistusel pannakse toimima lihtsaim rakendus veebinäite põhjal. Koosneb: Java Eclipse EE, MySQL, Apache Tomcat 9, Spring MVC.

MySQL eemaldamine [30] .:

MySQL Community Server-i reinstallimine. MySQL kasutamisel esineb mõnikord tõrkeid, mille korral eemaldatakse MySQL ja installeeritakse uuesti. Mõned näited, millest tõrge võib tuleneda: installeerimise käigus sätestati midagi valesti; MySQL asub küll kusagil arvutis, kuid mitte ettenähtud asukohas. Võimalik on, et MySQL satub arvutis teise kohta kõvaketaste haldamise protseduuride käigus.

Käsud:

```
$ sudo apt-get remove --purge MySQL-server MySQL-client MySQL-common
```

```
$ sudo apt-get autoremove
```

MySQL installeerimine [30] .:

```
$ sudo apt-get install -f MySQL-server
```

Andmebaasi loomine MySQL-is: vaata allikast [31]

Lisa 3 – Andmebaasi tabelid

```
CREATE TABLE user(
    userid INT(11) NOT NULL AUTO_INCREMENT,
    uname VARCHAR(45),
    email VARCHAR(45),
    password VARCHAR(100),
    counter INT(11),
    encrypteduname VARCHAR(200),
    encryptedpword VARCHAR(200),
    PRIMARY KEY(userid)
);
CREATE TABLE contact(
    contactid INT(11) NOT NULL AUTO_INCREMENT,
    firstname VARCHAR(45),
    lastname VARCHAR(45),
    email VARCHAR(45),
    userid INT(11),
    PRIMARY KEY(contactid),
    FOREIGN KEY(userid) REFERENCES user(userid)
    ON UPDATE RESTRICT ON DELETE CASCADE
);
INSERT INTO user(uname, email, password)
 VALUES('user01', 'user01@gmail.com', 'User01+');
INSERT INTO user(uname, email, password)
 VALUES('user02', 'user02@gmail.com', 'User02+');
INSERT INTO user(uname, email, password)
 VALUES('personal01', 'personal01@gmail.com', 'Personal01+');
 INSERT INTO user(uname, email, password)
 VALUES('user03', 'user03@gmail.com', 'User03+');

INSERT INTO contact(firstname, lastname, email, userid)
 VALUES('Mike', 'Farad', 'farad@gmail.com', 1);
INSERT INTO contact(firstname, lastname, email, userid)
 VALUES('Nancy', 'Nichols', 'nichols@gmail.com', 2);
INSERT INTO contact(firstname, lastname, email, userid)
 VALUES('Kaire', 'Kadakas', 'kadakas@gmail.com', 4);
```

```
CREATE TABLE roll(
    rollid INT(11) NOT NULL AUTO_INCREMENT,
    rname VARCHAR(45),
    PRIMARY KEY(rollid)
);

INSERT INTO roll(rname) VALUES('klient');
INSERT INTO roll(rname) VALUES('personal');

CREATE TABLE kasutajarollis(
    kasutajarollisid INT(11) NOT NULL AUTO_INCREMENT,
    kasutajaid INT(11) NOT NULL,
    rollid INT(11) NOT NULL,
    PRIMARY KEY(kasutajarollisid),
    FOREIGN KEY(kasutajaid) REFERENCES user(userid)
    ON UPDATE RESTRICT ON DELETE CASCADE,
    FOREIGN KEY(rollid) REFERENCES roll(rollid)
    ON UPDATE RESTRICT ON DELETE CASCADE
    );

INSERT INTO kasutajarollis(kasutajaid, rollid) VALUES(1, 1);
INSERT INTO kasutajarollis(kasutajaid, rollid) VALUES(2, 1);
INSERT INTO kasutajarollis(kasutajaid, rollid) VALUES(3, 2);
INSERT INTO kasutajarollis(kasutajaid, rollid) VALUES(4, 1);

CREATE TABLE periood(
    perioodid INT(11) NOT NULL AUTO_INCREMENT,
    alates DATETIME,
    kuni DATETIME,
    pernimetus VARCHAR(60),
    PRIMARY KEY(perioodid)
    );
```

```
INSERT INTO periood(alates, kuni, pernimetus)
 VALUES('2022-01-01 00:00:00' , '2022-02-01 00:00:00', '2022 Jaanuar');
INSERT INTO periood(alates, kuni, pernimetus)
 VALUES('2022-02-01 00:00:00' , '2022-03-01 00:00:00', '2022 Veebruar');
INSERT INTO periood(alates, kuni, pernimetus)
 VALUES('2022-03-01 00:00:00' , '2022-04-01 00:00:00', '2022 Märts');
INSERT INTO periood(alates, kuni, pernimetus)
 VALUES('2022-04-01 00:00:00' , '2022-05-01 00:00:00', '2022 Aprill');
INSERT INTO periood(alates, kuni, pernimetus)
 VALUES('2022-05-01 00:00:00' , '2022-06-01 00:00:00', '2022 Mai');
INSERT INTO periood(alates, kuni, pernimetus)
 VALUES('2022-06-01 00:00:00' , '2022-07-01 00:00:00', '2022 Juuni');
INSERT INTO periood(alates, kuni, pernimetus)
 VALUES('2022-07-01 00:00:00' , '2022-08-31 00:00:00', '2022 Juuli');


CREATE TABLE pakkumine(
    pakkumineid INT(11) NOT NULL AUTO_INCREMENT,
    nimetus VARCHAR(36),
    hind INT(11),
    kirjeldus VARCHAR(300),
    perioodid INT(11),
    PRIMARY KEY(pakkumineid),
    FOREIGN KEY(perioodid) REFERENCES periood(perioodid)
    ON UPDATE RESTRICT ON DELETE CASCADE
);

INSERT INTO pakkumine(nimetus, hind, kirjeldus, perioodid)
 VALUES('Stabiilne', 20, 'Max 10 kW', 1);
INSERT INTO pakkumine(nimetus, hind, kirjeldus, perioodid)
VALUES('Stabiilne', 20, 'Max 10 kW', 2);
INSERT INTO pakkumine(nimetus, hind, kirjeldus, perioodid)
VALUES('Stabiilne', 20, 'Max 10 kW', 3);
INSERT INTO pakkumine(nimetus, hind, kirjeldus, perioodid)
 VALUES('Stabiilne', 20, 'Max 10 kW', 4);
INSERT INTO pakkumine(nimetus, hind, kirjeldus, perioodid)
 VALUES('Stabiilne', 20, 'Max 10 kW', 5);
```

```
CREATE TABLE staatus(
    staatusid INT(11) NOT NULL AUTO_INCREMENT,
    nimetus VARCHAR(36),
    PRIMARY KEY(staatusid)
    );
INSERT INTO staatus(nimetus) VALUES('esitatud');
INSERT INTO staatus(nimetus) VALUES('menetluses');
INSERT INTO staatus(nimetus) VALUES('kinnitatud');
INSERT INTO staatus(nimetus) VALUES('teostamisel');
INSERT INTO staatus(nimetus) VALUES('teostatud');
INSERT INTO staatus(nimetus) VALUES('tühistatud');
CREATE TABLE ampermeeter(
    ampermeeterid INT(11) NOT NULL AUTO_INCREMENT,
    number VARCHAR(36),
    ipaddress VARCHAR(36),
    PRIMARY KEY(ampermeeterid)
    );
CREATE TABLE ameetrinait(
    ameetrinaitid INT(11) NOT NULL AUTO_INCREMENT,
    ampermeeterid INT(11) NOT NULL,
    aeg DATETIME,
    vool INT(11),
    PRIMARY KEY(ameetrinaitid),
    FOREIGN KEY(ampermeeterid) REFERENCES ampermeeter(ampermeeterid)
    ON UPDATE RESTRICT ON DELETE CASCADE
    );
INSERT INTO ampermeeter(number, ipaddress) VALUES('Amp001', null);
INSERT INTO ameetrinait(ampermeeterid, aeg, vool)
 VALUES(1, '2022-01-01 07:00:00', 66);
INSERT INTO ameetrinait(ampermeeterid, aeg, vool)
 VALUES(1, '2022-01-01 08:00:00', 60);
INSERT INTO ameetrinait(ampermeeterid, aeg, vool)
 VALUES(1, '2022-01-01 09:00:00', 50);
INSERT INTO ameetrinait(ampermeeterid, aeg, vool)
 VALUES(1, '2022-01-01 10:00:00', 52);
```

```
CREATE TABLE jaamavoimsus(
    jaamavoimsusid INT(11) NOT NULL AUTO_INCREMENT,
    perioodid INT(11),
    aeg TIME,
    pakutav INT(11),
    tellitud INT(11),
    jaak INT(11),
    too INT(11),
    PRIMARY KEY(jaamavoimsusid),
    FOREIGN KEY(perioodid) REFERENCES periood(perioodid)
    ON UPDATE RESTRICT ON DELETE CASCADE
);

INSERT INTO jaamavoimsus(perioodid, aeg, pakutav, tellitud, jaak, too)
 VALUES(1, '00:00:00', 20, 15, 5, 15);

CREATE TABLE reegel(
    reegelid INT(11) NOT NULL AUTO_INCREMENT,
    nimetus VARCHAR(36),
    kehtib boolean,
    kirjeldus VARCHAR(300),
    PRIMARY KEY(reegelid)
    );

INSERT INTO reegel(nimetus, kehtib, kirjeldus)
VALUES('Kliendi suunamine', 1,
 'Elektrijaamal on õigus tellimust mitte kinnitada.');

CREATE TABLE bilansihaldur(
    bilansihaldurid INT(11) NOT NULL AUTO_INCREMENT,
    nimetus VARCHAR(36),
    PRIMARY KEY(bilansihaldurid)
    );

INSERT INTO bilansihaldur(nimetus) VALUES('Electrum');
```

```
CREATE TABLE arvesti(
    arvestiid INT(11) NOT NULL AUTO_INCREMENT,
    number VARCHAR(36),
    peavoimsus INT(11),
    bilansihaldurid INT(11),
    ipaddress VARCHAR(36),
    PRIMARY KEY(arvestiid),
    FOREIGN KEY(bilansihaldurid) REFERENCES bilansihaldur(bilansihaldurid)
    ON UPDATE RESTRICT ON DELETE CASCADE
    );

INSERT INTO arvesti(number, peavoimsus, bilansihaldurid)
VALUES('Arv001', 4, 1);

CREATE TABLE arvestinait(
    arvestinaitid INT(11) NOT NULL AUTO_INCREMENT,
    arvestiid INT(11) NOT NULL,
    aeg DATETIME,
    energia INT(11),
    PRIMARY KEY(arvestinaitid),
    FOREIGN KEY(arvestiid) REFERENCES arvesti(arvestiid)
    ON UPDATE RESTRICT ON DELETE CASCADE
    );

INSERT INTO arvestinait(arvestiid, aeg, energia)
 VALUES(1, '2022-01-01 07:00:00', 10);
INSERT INTO arvestinait(arvestiid, aeg, energia)
 VALUES(1, '2022-01-01 08:00:00', 20);
INSERT INTO arvestinait(arvestiid, aeg, energia)
 VALUES(1, '2022-01-01 09:00:00', 30);

CREATE TABLE kliendiarvesti(
    kliendiarvestiid INT(11) NOT NULL AUTO_INCREMENT,
    klientid INT(11) NOT NULL,
    arvestiid INT(11) NOT NULL,
    PRIMARY KEY(kliendiarvestiid),
    FOREIGN KEY(klientid) REFERENCES contact(contactid)
    ON UPDATE RESTRICT ON DELETE CASCADE,
    FOREIGN KEY(arvestiid) REFERENCES arvesti(arvestiid)
    ON UPDATE RESTRICT ON DELETE CASCADE
    );
```

```
INSERT INTO kliendiarvesti(klientid, arvestiid) VALUES(1, 1);

CREATE TABLE tellimus(
    tellimusid INT(11) NOT NULL AUTO_INCREMENT,
    klientid INT(11) NOT NULL,
    pakkumineid INT(11) NOT NULL,
    alates DATETIME,
    kuni DATETIME,
    perioodid INT(11),
    staatusid INT(11),
    arvestiid INT(11),
    PRIMARY KEY(tellimusid),
    FOREIGN KEY(klientid) REFERENCES contact(contactid)
    ON UPDATE RESTRICT ON DELETE CASCADE,
    FOREIGN KEY(pakkumineid) REFERENCES pakkumine(pakkumineid)
    ON UPDATE RESTRICT ON DELETE CASCADE,
    FOREIGN KEY(perioodid) REFERENCES periood(perioodid)
    ON UPDATE RESTRICT ON DELETE CASCADE,
    FOREIGN KEY(staatusid) REFERENCES staatus(staatusid)
    ON UPDATE RESTRICT ON DELETE CASCADE,
    FOREIGN KEY(arvestiid) REFERENCES arvesti(arvestiid)
    ON UPDATE RESTRICT ON DELETE CASCADE
    );
INSERT INTO tellimus(
klientid, pakkumineid, alates, kuni, perioodid, staatusid, arvestiid)
 VALUES(1, 1, '2022-04-01 00:00:00', '2022-05-01 00:00:00', 1, 1, 1);

CREATE TABLE tarbitudenergia(
    tarbitudenergiaid INT(11) NOT NULL AUTO_INCREMENT,
    arvestiid INT(11) NOT NULL,
    alates DATETIME,
    kuni DATETIME,
    energia INT(11),
    hind INT(11),
    summa INT(11),
    pernimetus VARCHAR(60),
    perioodid INT(11) NOT NULL,
    PRIMARY KEY(tarbitudenergiaid),
    FOREIGN KEY(arvestiid) REFERENCES arvesti(arvestiid)
    ON UPDATE RESTRICT ON DELETE CASCADE,
    FOREIGN KEY(perioodid) REFERENCES periood(perioodid)
    ON UPDATE RESTRICT ON DELETE CASCADE
    );
```

Lisa 4 – Eclipse'is arendatud skripte

Vaate genereerimine

```
<table border="1" cellpadding="5">
    <tr>
        <th>Nr</th>
        <th>Pakkumine</th>
        <th>Klient</th>
        <th>Periood</th>
        <th>Staatus</th>
        <th>
            <form:form action="perioodtellimuseleForm" method="get"
            style="margin-top:0px">
            <input onclick="update()" type="submit" value="Uus Tellimus"/>
            </form:form></th></tr>
    <c:forEach items="${listTellimusView}" var="tellimusview" varStatus="status">
    <tr style="font-size:18px">
        <td>${status.index + 1}</td>
        <td>${tellimusview.pakkumine}</td>
        <td>${tellimusview.klient}</td>
        <td>${tellimusview.periood}</td>
        <td>${tellimusview.staatus}</td>
        <td>
            <a class="linkk" style="font-size:18px" href="edittellimus?id=
            ${tellimusview.id}&rollid=${tellimusview.rollid}">Muuda</a>
              
            <a class="linkk" style="font-size:18px"
             href="deletetellimus?id=${tellimusview.id}">Kustuta</a></td></tr>
    </c:forEach>
</table>
```

Joonis 18. Kliendi tellimuste vaate genereerimise lähtekood

|Avalehele |Ajalugu

Tellimuste loend

No	Pakkumine	Klient	Periood	Staatus	Uus Tellimus
1	**Pakett:** Stabiilne, **Hind:** 20 [s/kWh], **Kirjeldus:** Max 10 kW	Mike Farad	Juuni 2022	esitatud	Muuda Kustuta

Joonis 19. Kliendi tellimuste vaade

```
<form:form action="savetellimus" method="post" modelAttribute="tellimus">
<table cellpadding="5">
<form:hidden path="id" />
    <tr>
        <td>Klient</td>     <td>
            <form:hidden path="klientid" value="${klient.id}"/>
            ${klient.name} ${klient.lastName}</td></tr>
    <tr>
        <td>Periood</td>    <td>
            <form:hidden path="perioodid" value="${periood.id}"/>
            ${periood.perNimetus}   </td></tr>
    <tr>
        <td>Staatus</td>    <td>
            <form:hidden path="staatusid" value="${staatus.id}"/>
            ${staatus.nimetus}  </td></tr>
    <tr>
        <td>Pakkumine</td>  <td>
            <form:select path="pakkumineid">
                <form:option value="${pakkumine.id}"
                 label="${ pakkumine.nimetus }"/>
                <form:options items="${listPakkumine}"
                itemValue="id" itemLabel="nimetus"/>
            </form:select></td></tr>
    <tr>
        <td>Arvesti</td>    <td>
            <form:select path="arvestiid">
                <form:option value="${arvesti.id}" label="${arvesti.number}"/>
                <form:options items="${listArvesti}"
                itemValue="id" itemLabel="number"/>
            </form:select>      </td>   </tr>
    <tr>
        <td colspan="2" align="center">
            <input type="submit" value="Kinnita"/></td></tr>
</table>
</form:form>
```

Joonis 20. Muutmise eeltäidetud vaate genereerimise lähtekood ja vaade

Perioodi valiku vaate mudel:

```
<div align="center">
<a class="linkk" href="userWelcomeSec?user=${userName}"
   style="background-color:lightgreen">|Avalehele</a>
    <h1>Perioodi valik tellimusele</h1>
    <form:form action="perioodtellimuseleForm" method="post"
    modelAttribute="tellimus">
    <table cellpadding="5">
        <!--<form:hidden path="klientid" />-->
        <form:hidden path="userName" />
        <form:select path="perioodid">
            <form:option value="0"
             label="--Vali periood"/>
            <form:options items="${ filtListPeriood }"
            itemValue="id" itemLabel="perNimetus"/>
        </form:select></td>
        <td style="color:red"><form:errors path="perioodid" /></td>
        </tr>
        <tr>
            <td colspan="2" align="center"><input
             type="submit" value="Kinnita valik"/></td>
        </tr>
    </table>
    </form:form>
</div>
```

Joonis 21. Perioodi valiku vaate mudel ja vaade

Kontroller

```
@RequestMapping(value = "/minutellimused", method = RequestMethod.GET)
@PreAuthorize("#request.getParameter('user') == authentication.name ||"
                + "authentication.name=='personal01'")
public ModelAndView minuTellimused(ModelAndView model,
                HttpServletRequest request, Authentication authentication) {
        String userName = authentication.getName();
        Integer klientid = 0;
        if(!userName.equals("personal01")) {
                Integer userid = userDAO.getUserIdByUName(userName);
            Contact klientA = klientDAO.getByUserId(userid);
            klientid = klientA.getId();
        }
        if(userName.equals("personal01")) {
                klientid = Integer.parseInt(request.getParameter("id"));
        }
        Integer rollid = Integer.parseInt(request.getParameter("rollid"));
        List<Tellimus> listTellimus = tellimusDAO.listByKlient(klientid);
        List<TellimusView> listTellimusView = new ArrayList<>();
        listTellimusView = koostaListTellimusView(listTellimus, klientid, rollid);
        model.addObject("listTellimusView", listTellimusView);
        model.addObject("rollid", rollid);
        model.addObject("userName", userName);
        model.setViewName("tellimused");
        if(rollid == 2) {
                model.setViewName("personal/ptellimused");
        }
        return model;
}
```

Joonis 22. Tellimuse loendi kontroller

```
@RequestMapping(value = "/savetellimus", method = RequestMethod.POST)
public void saveTellimus(ModelAndView model, @ModelAttribute Tellimus tellimus,
        HttpServletResponse response, Authentication authentication)
                throws IOException {
    String teade = "Tellimuse esitamine oli edukas";
    if (tellimus.getId() == null) {
        tellimus.setStaatusid(1);
        tellimusDAO.save(tellimus);
    } else {
        tellimus.setStaatusid(7);
        tellimusDAO.update(tellimus);
        teade = "Tellimuse muutmine oli edukas";
    }
    Integer klientid = tellimus.getKlientid();
    Contact klient = klientDAO.get(klientid);
    String userName = authentication.getName();
    response.sendRedirect(
            "/Station01/userWelcomeSec?user="+userName+"&message="+teade);
}
```

Joonis 23. Tellimuse salvestamise kontroller

DAO

```
@Override
public int save(TarbitudEnergia t) {
    String sql = "INSERT INTO tarbitudenergia ("
            + "arvestiid, alates, kuni, energia, hind, summa, pernimetus, perioodid)"
            + " VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
    return jdbcTemplate.update(sql, t.getArvestiid(), t.getAlates(),
            t.getKuni(), t.getEnergia(), t.getHind(), t.getSumma(),
            t.getPerNimetus(), t.getPerioodid());
}
@Override
public List<TarbitudEnergia> list(Integer arvestiid, Integer perioodid) {
    String sql = "SELECT * FROM tarbitudenergia WHERE arvestiid=" + arvestiid
            + " AND perioodid=" + perioodid;
    RowMapper<TarbitudEnergia> rowMapper = new RowMapper<TarbitudEnergia>() {
        @Override
        public TarbitudEnergia mapRow(ResultSet rs, int rowNum) throws SQLException {
            Integer id = rs.getInt("tarbitudenergiaid");
            Integer arvestiid = rs.getInt("arvestiid");
            Timestamp alates = rs.getTimestamp("alates");
            Timestamp kuni = rs.getTimestamp("kuni");
            Integer energia = rs.getInt("energia");
            Integer hind = rs.getInt("hind");
            Integer summa = rs.getInt("summa");
            String perNimetus = rs.getString("pernimetus");
            Integer perioodid = rs.getInt("perioodid");
            return new TarbitudEnergia(id, arvestiid, alates, kuni, energia,
                    hind, summa, perNimetus, perioodid);
        }
    };
    return jdbcTemplate.query(sql, rowMapper);
}
```

Joonis 24. Tarbitud Energia DAO

Mudeli atribuudi saatmine

Atribuudi „userName" saatmine: episood 1

```
@Override
public void onAuthenticationSuccess(HttpServletRequest request,
        HttpServletResponse response,
        Authentication authentication) throws IOException, ServletException {
    String userName = request.getParameter("userName");
    String password = request.getParameter("password");
    if(hasRole("ROLE_personal")) {
        response.sendRedirect("/Station01/personalWelcomeSec");
    }
    if(hasRole("ROLE_klient")) {
        response.setHeader("userName", userName);
        response.sendRedirect("/Station01/userWelcomeSec?user="+userName);
    }
}
```

Atribuudi „userName” saatmine: episood 2

```
<a class="linkk" style="font-size:20px"
href="minutellimused?rollid=1&user=${userName}">TELLIMUSED|</a>
```

Atribuudi „userName” saatmine: episood 3

```
@RequestMapping(value = "/minutellimused", method = RequestMethod.GET)
@PreAuthorize("#request.getParameter('user') == authentication.name ||"
                + "authentication.name=='personal01'")
public ModelAndView minuTellimused(ModelAndView model,
                HttpServletRequest request, Authentication authentication) {
        String userName = authentication.getName();
        Integer klientid = 0;
        if(!userName.equals("personal01")) {
                Integer userid = userDAO.getUserIdByUName(userName);
            Contact klientA = klientDAO.getByUserId(userid);
            klientid = klientA.getId();
        }
        if(userName.equals("personal01")) {
                klientid = Integer.parseInt(request.getParameter("id"));
        }
        Integer rollid = Integer.parseInt(request.getParameter("rollid"));
        List<Tellimus> listTellimus = tellimusDAO.listByKlient(klientid);
        List<TellimusView> listTellimusView = new ArrayList<>();
        listTellimusView = koostaListTellimusView(listTellimus, klientid, rollid);
        model.addObject("listTellimusView", listTellimusView);
        model.addObject("rollid", rollid);
        model.addObject("userName", userName);
        model.setViewName("tellimused");
        if(rollid == 2) {
                model.setViewName("personal/ptellimused");
        }
        return model;
}
```

Valideerimine

```
@PostMapping("/perioodtellimuseleForm")
@PreAuthorize("#tellimus.getUserName() == authentication.name")
public String valiPerioodTellimusele(
        @Valid @ModelAttribute("tellimus") Tellimus tellimus,
        Errors errors,
        Model model,
        HttpServletRequest request,
        Authentication authentication
        ) {
    String userName = authentication.getName();
    boolean selectError = false;
    if(errors.hasErrors()) {
        selectError = true;
    }
    if(selectError==false) {
        userName = authentication.getName();
        model = modifitseeriModelKorrektne(model, userName, tellimus);
        return "tellimusForm";
    }
    //errors == true
    List<Periood> filtListPeriood = new ArrayList<Periood>();
    filtListPeriood = filtreeriPerioodid(userName);
    model.addAttribute("filtListPeriood", filtListPeriood);
    model.addAttribute("userName", userName);
    return "perioodtellimuseleForm";
}
```