TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Annika Suurna 222646IAIB

Töötajate ajahaldussüsteemi arendamine Sigma Polymer Groupile

Bakalaureusetöö

Juhendaja: Jan Kõrva
andmeanalüütik

Kaasjuhendaja: Ian Erik Varatalu
doktorant-nooremteadur

Tallinn 2025

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: Annika Suurna

19.05.2025

Annotatsioon

Antud bakalaureusetöö eesmärk on arendada Sigma Polymer Groupile uus veebipõhine töötajate ajahaldussüsteem. Rakendus võimaldab töötajatel ja nende otsestel juhtidel jälgida tööajaga seotud statistikat, sealhulgas liikumisi uste kaudu, töötunde ning teisi arvutusi. Kuna olemasolev süsteem põhineb vananenud tarkvaral ja on aastate jooksul saanud mitmeid ebavajalikke lisasid, otsustati luua täiesti uus lahendus, säilitades üksnes olemasoleva andmebaasi.

Töö esimeses osas analüüsitakse olemasolevat süsteemi ja kaardistatakse alternatiivsed lahendused. Teises osas põhjendatakse valitud tehnoloogiad ja määratletakse vajaminevad funktsionaalsused. Kolmandas osas kirjeldatakse arendusprotsessi ning tehakse ülevaade tehnilistest ja disainivalikutest. Neljandas osas käsitletakse valideerimisprotsessi, saadud tagasisidet ja tehtud täiendusi. Viiendas osas tutvustatakse rakenduse kasutajakogemust ning disainivalikuid. Lõpetuseks hinnatakse loodud lahendust ning koostatakse kokkuvõte.

Töö tulemusena valmis kaasaegne ja kasutajasõbralik veebirakendus, mis vastab seatud nõuetele.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 40 leheküljel, 8 peatükki, 11 joonist, 3 tabelit.

Abstract

Development of an Employee Time Management System for Sigma Polymer Group

The aim of this bachelor's thesis is to develop a new web-based employee time management system for Sigma Polymer Group. The application enables employees and their direct managers to monitor work-related statistics, including door entries, working hours, and various calculations. Since the previous system was based on outdated software and had accumulated several unnecessary features over the years, it was decided to build a completely new solution while retaining the existing database.

The first part of the thesis analyzes the existing system and explores alternative solutions. The second part explains the chosen technologies and defines the required functionalities. The third part describes the development process and justifies the technical and design decisions made. The fourth part presents the validation process, user feedback, and the resulting improvements. The fifth part focuses on the user experience and design aspects of the developed application. Finally, the developed solution is evaluated, and a summary is provided.

The result is a modern and user-friendly web application that meets the specified requirements.

The thesis is in Estonian and contains 40 pages of text, 8 chapters, 11 figures, 3 tables.

Lühendite ja mõistete sõnastik

API	Rakendusliides (*Application Programming Interface*)
UI	Kasutajaliides (*User Interface*)
SQL	Struktureeritud päringukeel (*Structured Query Language*)
JWT	JSON-veebimärgis (*JSON Web Token*)
JSON	JavaScripti objektimärgendus (*JavaScript Object Notation*)
PDF	Kantav dokumendivorming (*Portable Document Format*)
XLSX	Exceli arvutustabelivorming (*Excel Spreadsheet Format*)
CSV	Komaga eraldatud väärtused (*Comma-Separated Values*)
IT	Infotehnoloogia (*Information Technology*)
HTML	Hüperteksti märgenduskeel (*HyperText Markup Language*)
CSS	Kaskaadlaadide stiililehed (*Cascading Style Sheets*)
MS	Microsoft
ID	Isikutuvastus (*Identification*)
REST	Representatiivne olekuedastus (*Representational State Transfer*)
API-päringud	Rakendusliidese päringud (*API requests*)
SSMS	SQL Serveri haldustööriist (*SQL Server Management Studio*)
VPN	Virtuaalne privaatvõrk (*Virtual Private Network*)

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Töötajate ajahaldustarkvara suurendab tootlikkust mitmel viisil. See tõstab individuaalset vastutust ja parandab ajakasutust, võimaldades meeskondadel oma töökeskkondades edukalt toimida [1].

Antud bakalaureusetöö eesmärk on luua Sigma Polymer Groupile uus töötajate ajahaldussüsteem. Rakendus peab võimaldama töötajatel ning nende otsestel juhtidel jälgida tööajaga seotud statistikat. Loodav rakendus peab andma kasutajale ülevaate tööl veedetud ajast ning liikumistest. Rakendus peab olema modernne, kergesti kasutatav ning sobituma erinevate ekraanitüüpidega.

Algne probleem tekkis, kuna varem kasutusel olev rakendus on rikkis ning vajab väljavahetamist. Esines tõrkeid uute inimeste andmetega ning kokkujooksmistega.

Pärast alternatiivide kaalutlemist ning varasema rakenduse analüüsimist sai langetatud otsus luua uus rakendus nullist. Selleks analüüsitakse olemasolevaid tehnoloogiaid, et leida probleemile kõige sobivam lahendus. Järgmise sammuna arendatakse ees- ja tagarakendus ning need sobitatakse olemasoleva andmebaasiga. Lõpetuseks valideeritakse tehtud rakendus ning tehakse viimased muudatused saadud tagasiside põhjal.

2 Probleemi püstitus ja analüüs

Töötajate ajakasutuse jälgimise rakendustel on mitmeid eeliseid. Need vähendavad vigade tekkimise võimalust palgaarvestuses, aitavad tagada tööseaduste täitmist ning võimaldavad paremat ressursside planeerimist. Lisaks pakuvad need tööriistad väärtuslikku teavet töötajate tulemuslikkuse kohta, aidates juhtidel tuvastada arendamist vajavaid valdkondi ning langetada teadlikumaid otsuseid personalipoliitika ja tööaja planeerimise osas [2].

2.1 Olemasoleva süsteemi analüüs

Hetkel kasutatakse varem loodud majasisest veebirakendust TimeMaster, mis visualiseerib haldusprogrammi Argos Commander andmeid kasutajasõbralikumal kujul. Andmed asuvad Microsoft SQL Server Express andmebaasiserveris.

Vana rakendus on ehitatud Laravel 5.5 raamistikule, mis eeldab PHP versiooni 7.0 või uuemat. Esirakenduses kasutatakse Vue.js versiooni v2.1.10 ning mitmeid tolle aja levinud teeke, nagu Vuex 2.4.1, Bootstrap-Sass 3.3.7 ja Laravel Mix 1.0. Võrdluseks, töö kirjutamise hetkel on uusimad versioonid vastavalt Laravel 12 [3], PHP 8.4 [4] ja Vue.js 3.5.13 [5]. Nii esi- kui ka tagarakendus põhinevad versioonidel, mille ametlik tugi on lõppenud või millele ei pakuta enam turvauuendusi. See muudab süsteemi haavatavaks ning raskendab edasist arendamist ja laiendamist.

Lisaks on rakenduses ilmnenud probleeme, näiteks tõrked uute kasutajate andmetega ning mitmed lisamoodulid, nagu WageMaster, mis ei ole enam kasutuses. Olemasoleva koodi modifitseerimine nende probleemide lahendamiseks võib olla ajamahukas ning riskantne.

Kõiki eeltoodud probleeme arvestades otsustati olemasolev rakendus asendada täiesti uue, kaasaegsetel tehnoloogiatel põhineva lahendusega.

2.2 Alternatiivsed lahendused

Lisaks uue rakenduse loomisele tuleks valikuna kaaluda ka olemasolevaid alternatiivseid lahendusi. Kuna tegemist ei ole firma jaoks kriitilise teenusega, ei ole ajahalduse jaoks plaanis eraldada rahalisi ressursse, nii et tegemist peaks olema vabavaraga. Kuigi on mitmeid tasuta alternatiive, kaasnevad nendega teatud miinused, mis muudavad need kliendile sobimatuks. Siiski saab alternatiive analüüsides tuvastada potentsiaalseid lisafunktsionaalsusi või disainitäiendusi, mida saab integreerida loodavasse rakendusse, et pakkuda meeldivamat kasutajakogemust.

2.2.1 Excel

Ettevõtte poolt loodi ajutiseks lahenduseks tabelisüsteem Microsoft Excelis. Tegemist on kokkuvõtva tabeliga, kus on võimalik vaadelda töötajaid ning nende liikumisi turvaustest.

Ajutise lahendusena tabel on kasulik mõnes aspektis: seda on võimalik võrdlemisi kiiresti kasutusele võtta, tegemist on tuntud tarkvaraga, milles on lihtne orienteeruda ning vajadusel saab juurde lisada Excelis sisalduvaid lisafunktsionaalsuseid, näiteks arvutusi.

Siiski leitakse, et pikaajalisel kasutamisel tekivad probleemid. Esiteks on tabelit kasutajatel ebamugav vaadata. Tabel on visuaalselt tagasihoidlik ning ebameeldiv kasutada erinevates seadmetes. Samuti on tegemist halvasti skaleeruva lahendusega, kuna suured tabelid muutuvad aeglaseks. Mitme inimese samaaegne kasutamine võib tekitada failikonflikte, mis võivad põhjustada andmete kaduma minekut.

Samuti ei ole võimalik hallata erinevaid kasutajaõiguseid. Firma soov on omada kolme kasutajarolli: administraatorid, osakonnajuhid ja töötajad. Ühe suure Exceli tabeliga ei ole võimalik realiseerida osakonnajuhtide rolli, et otsene juht saaks ainult oma alluvate andmeid vaadata. Alternatiivina saaks luua igale grupile eraldi Exceli tabeli ning anda ligipääsu ainult vastavatele juhtidele, kuid see tekitab lisa haldusprobleeme. Samuti tekib sarnane probleem üksikute töötajatega, kes sooviksid ning tohiksid vaadelda ainult enda detaile. Lühidalt on lahendus ajutiselt hea, kuid pikaajaliseks kasutuseks sobimatu.

2.2.2 Muu vabavaraline lahendus

Enne uue lahenduse loomist on arukas võrrelda olemaolevaid vabavaralisi lahendusi. Nii saab võrrelda, kas uue rakenduse loomine on üldse õigustatud. Kui leitakse, et uue rakenduse loomine on õigustatud, saab siiski analüüsida olemasolevate lahenduste plusse ja miinuseid.

Metabase

Lähim variant soovitud tulemuse saavutamiseks on kasutada Metabase rakendust. Tegemist on populaarse avatud lähtekoodiga ärianalüüsi tööriistaga, mis võimaldab luua otseühendust andmebaasiga ning visualiseerib erinevaid näitajaid, filtreid ja graafikuid. Metabase plussideks on Microsoft SQLi otsene ühendus, on võimalik lingi ja muu abil kergelt jagada ning tasuta versioon on üpriski mahukas. Miinusteks on kohandatavuse piiratus, kuna UI on rakenduse poolt lukus. Samuti on kasutajaliides suunatud juhtidele ja analüütikutele ning võib tavakasutajatele olla ebamugav. Lisaks tuleb koguarvutused, näiteks üle/alla normi, arvutada läbi SQLi queri, mis on raskesti hallatav [6].

Looker Studio

Looker Studio on Google'i pakutav tasuta andmevisualiseerimise tööriist. Rakendust on võimalik otse Microsoft SQLiga ühendada, on lihtsa visuaalse välimusega, standardversioon on tervenisti tasuta ning võimaldab visualiseerid andmeid nii tabelite, diagrammide kui ka graafikute abil. Suurimateks miinusteks on andmebaasiga turvalise ühenduse keeruline seadistamine (võimalus on andmed üles laadida Google Sheeti, aga see on liigselt keeruline ning aeglane). Samuti on kõik loodavad vaated samad ning ei ole võimalik tekitada erinevaid kasutajarolle (puudub rollihaldus) [7].

Kokkuvõttev analüüs valikutest

Kokkuvõttev analüüs loodava rakenduse ja alternatiivide vahel on leitav tabelist 1.

Tabel 1. Loodava rakenduse ja alternatiivide analüüs.

Analüüsitav omadus	**Loodav rakendsus**	**Metabase**	**Looker Studio**
Microsoft SQL ühendus	Otsene	Otsene	Piiratud/Kaudne
Visualiseerimine	Täielik kontroll	Hea	Piiratud
Rollihaldus	Täielik kontroll	Piiratud	Puudub
Arendus/Seadistusaeg	Kõrge	Mõõdukas	Väga kiire
Kohandatavus	Täielik	Keskmine	Väike

Jätkub...

Tabel 1 – *Jätkub järgmisel lehel...*

Analüüsitav omadus	**Loodav rakendsus**	**Metabase**	**Looker Studio**
Kasutajate suunamine	Töötajavaade	Pigem juhtidele	Kõigile sama
Hind	Tasuta	Tasuta*	Tasuta*

Tasuta* tähendab, et on olemas tasuta versioon, kuid ettevõtte suurenedes võib olla vajalik osta tasuline pakett.

3 Lahenduse analüüs

Enne arenduse algust sai paika pandud vajalikud nõuded süsteemile ning kasutatavad tehnoloogiad.

3.1 Nõuded süsteemile

Tähtis on paika panna süsteemi nõuded, millele tuginedes saab rakendust ehitada vastavalt tellija soovile.

3.1.1 Funktsionaalsed nõuded

Funktsionaalsed nõuded on juhised, mis määratlevad süsteemi toimingud, käitumised ning funktsioonid. Lühidalt, need kirjeldavad, mida süsteem peab tegema [8].

- Kasutaja peab saama sisse logida kasutajanime ja salasõnaga.
- Tagarakendus peab looma JWT ehk JSON-veebimärgise, mis sisaldab kasutaja andmeid ning rolli.
- Kasutaja peab saama vaadata oma kuu graafikut.
- Kasutaja peab saama sorteerida oma graafikut kuu ja aasta põhjal.
- Kasutaja peab saama vaadelda konkreetse päeva detaile.
- Süsteem peab kuvama andmed SQLi-andmebaasist.
- Süsteem peab oskama visualiseerida andmeid loetavas formaadis (tabelid, graafikud).
- Probleemi korral peab süsteem kuvama selgitava veateate.
- Kasutaja peab saama enda töötunde eksportida .pdf või .xlsx vormingus.
- Süsteem peab kuvama iganädalast ja igakuist statistikat (keskmine, ületunnid jne).

3.1.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded keskenduvad rakenduse üldistele omadustele ja käitumistele erinevates tingimustes. Lühidalt, need määravad süsteemi toimimise [8].

- Süsteem peab töötama nii arvutis kui ka telefonis
- Rakendus peab olema saadaval 99.9 protsenti ajast
- Kõik kasutajad peavad autentima isikuliselt
- Kasutajaliides peab olema eestikeelne
- Süsteem peab taluma samaaegselt kõiki töötajaid
- Andmevaated peavad olema ligipääsupiiranguga vastavalt rollile
- JWT'id peavad olema ajaliselt piiratud ja kontrollitud
- Rakendus peab olema kasutatav ka mitte-IT tehnilistele inimestele

3.1.3 Kasutajarollid ja ligipääsud

Rakenduses on defineeritud kolme tüüpi kasutajad: töötajad, otsesed juhid ja administraatorid. Enamik vaateid on kolmel grupil samad, kuid erinevus tuleb sisse vaadeldavate kasutajate arvus. Töötajad saavad vaadelda ainult enda graafikut ning tööpäevadega seotud detaile. Juhid saavad vaadata ainult enda ning nende alluvate graafikuid. Samuti peavad nad saama võrrelda töötajate statistikat, nagu näiteks keskmisi töötunde. Admin saab vaadelda kõigi töötajate andmeid. Rollid on määratud Argoses ning andmeid saab vaadata ainult sisselogitud kasutaja.

3.2 Tehnoloogiline valik ja arhitektuur

Tehnoloogia valik on üks viimaseid samme enne arendusprotsessi alustamist. Selleks, et vältida probleeme tulevikus, tuleb tarkvaraotsused hästi läbi mõelda.

3.2.1 Esirakendus

Eesrakenduse arendamiseks sai valitud Vue.js raamistik. Vue.js on lihtsalt kasutatav JavaScript-raamistik, mis põhineb standardsetel veebitehnoloogiatel nagu HTML, CSS ja JavaScript. Samuti pakub see intuitiivset API-d, reaktiivset ja jõudlust optimeerivat renderdussüsteemi ning mitmekülgset ökosüsteemi, mis võimaldab kasutada seda nii kergekaalulise teegina kui ka täisfunktsionaalse raamistikuna [9].

Alternatiivseteks valikuteks oli ka React ja Angular, kaks paljulevinud eesrakenduse raamistikku. React on JavaScript'i teek, mis võimaldab luua kasutajaliideseid väikestest, taaskasutatavatest komponentidest. React võimaldab arendajatel koostada keerukaid ka-

sutajaliideseid, kombineerides neid komponente terviklikeks lehtedeks ja rakendusteks [10]. Angular on arendajate seas populaarne veebiarendusraamistik, mis võimaldab luua turvalisi, ligipääsetavaid ja skaleeruvaid rakendusi [11].

Otsus sai langetatud Vue.js kasuks, kuna autoril on varasemat kogemust ning tegeleb sellega igapäevatöös, mis teeb arenduse protsessi kiiremaks. Samuti on Vue.js kergesti kasutatav, kompaktne ning maailmaklassilise dokumentatsiooniga [12].

3.2.2 Tagarakendus

Tagarakenduse arendamiseks valiti Spring Boot raamistik, kuna see võimaldab kiiresti luua tootmiskõlbuliku veebirakenduse minimaalsete seadistustega. Spring Booti tugevus seisneb tema modulaaruses, andmebaasitoes ja turvavõimalustes, mis on olulised ajahaldussüsteemi loomisel [13].

Alternatiividena kaaluti veel Node.js + Expressi ja Djangot. Node.js koos Express raamistikuga on tasuta, avatud lähtekoodiga ja platvormiülene JavaScripti käituskeskkond, mis võimaldab arendajatel luua servereid, veebirakendusi, käsureatööriistu ja skripte [14]. Django on kõrgetasemeline Pythoni veebiraamistik, mis soodustab kiiret arendust ja puhast, pragmaatilist disaini [15].

Otsus sai langetatud Spring Booti kasuks, kuna see võimaldab luua paremini struktureeritud, skaleeritava ja turvalise veebirakenduse, mis toetub Java ökosüsteemile. Võrreldes Node.js-i ja Djangoga pakub Spring Boot paremat sobivust pikaajalise hooldatavuse ja suuremate süsteemidega integreerimise jaoks. Samuti on autoril pikaajaline kogemus Spring Booti kasutamisega, mis kiirendab arendusprotsessi ja tagab puhtama koodi.

3.2.3 Andmebaas

Ettevõte kasutab andmete hoiustamiseks Microsoft SQL Server Express andmebaasi. MS SQL Express on Microsofti pakutava SQL Serveri relatsioonilise andmebaasi haldussüsteemi versioon, mis on tasuta ning kergem kui originaal [16]. Kuna olemasolev andmebaas on juba ühendatud haldussüsteemi Argos Commanderiga, palus ettevõte vältida selle struktuuri muutmist.

3.2.4 Autentimine

Kasutajate autentimiseks ja rollide haldamiseks valiti JSON-veebimärgis. JWT on kompaktses ja allkirjastatud formaadis JSON-objekt, mida kasutatakse autentimisinfo turvaliseks edastamiseks kahe osapoole vahel. See sisaldab nõudmisel lisatavat kasutajainfot (näiteks kasutajanimi ja roll) ning võimaldab olekuta autentimist, kus server ei pea haldama seansse, vaid valideerib iga päringu juures märgise autentsuse [17].

Alternatiivina kaaluti sessioonipõhist autentimist. Sessioonipõhise autentimise korral salvestab server kasutaja kohta seansiteabe, millele viitab kliendi poolel olev sessiooni-ID. Iga kasutajapäringuga kontrollitakse seda ID-d serveripoolses mälus või andmebaasis. Otsus langetati JWT kasuks, kuna see sobib hästi Spring Booti REST-põhise arhitektuuriga, toetab lihtsat skaleeritavust ning ei vaja serveripoolset seisundit [18].

Otsus sai langetatud JWT kasuks. Spring Security integreerub loomulikult JWT-ga, võimaldades defineerida rollipõhiseid juurdepääsureegleid ja käsitleda märgise valideerimist filtreerimiskihi kaudu, mis lihtsustab arendust ja parandab jõudlust [19].

4 Rakenduse arendusprotsess

Antud veebirakenduse arendus koosnes kuuest põhietapist: disain, algseadistus, esirakenduse arendus, tagarakenduse arendus, andmebaasi analüüs ja kasutuselevõtt ning tootmiskeskonnaks ettevalmistamine.

4.1 Disain ja planeerimine

Esialgne planeerimine on arendusprotsessi üks olulisemaid etappe, kuna see aitab luua arendajale tervikliku ettekujutuse rakenduse vaadetest, info liikumisest ning kasutajavoogudest. Kui disainiprotsess vahele jätta, on keeruline mõista tervikpilti ning see võib negatiivselt mõjutada lõppkasutaja kogemust.

Planeerimine algas varasema rakenduse analüüsiga. Kuna ettevõte soovis säilitada põhivaated funktsionaalselt sarnastena, jäi ka uue lahenduse vaadete disain üldjoontes samaks, sisaldades vaid väiksemaid muudatusi. Täiendavalt soovis tellija lisada statistika lehe, mis võimaldab kasutajal vaadelda isiklikke andmeid või otsestel juhtidel teiste töötajate statistikat ja neid omavahel võrrelda.

Järgmise sammuna visandati vaadete struktuur paberile. Autor joonistas kõik neli peamist vaadet ning lisas iga infoedastuse või üleminekukoha juurde esialgsed API-päringud, et aidata kaasa komponentide vahelise seose mõistmisele. Lõpetuseks koostati ka esialgsed objektimudelid, mille tagarakendus pidi eesrakendusele saatma.

4.2 Töökeskkonna seadistus

Arendusprojekt jagati alguses kaheks eraldiseisvaks rakenduseks – esirakenduseks ja tagarakenduseks, mis suhtlevad omavahel läbi Axiose. Arenduskeskkonnana valiti IntelliJ IDEA, kuna see on võimas ja kasutajasõbralik IDE, mis toetab kogu tarkvaraarenduse elutsüklit ning aitab luua kvaliteetset koodi kiiresti ja mugavalt [20].

Tagarakendus loodi Spring Initializriga ning eesrakendus Vue.js raamistikku kasutades. Versioonihaldusteenusena kasutati Giti – avatud lähtekoodiga hajusat süsteemi, millel on kiire töövõimekus ning lihtne kasutusloogika [21]. Koodi haldamiseks ja koostöö korraldamiseks kasutati GitLabi, mis ühendab DevSecOps-i töövoo ühtseks tervikuks [22].

Failistruktuuris loodi tagarakenduse jaoks kaust `iaib-be` ning eesrakenduse jaoks kaust `iaib-fe`. Osade vahelise ühenduse testimiseks loodi fail `axiosInstance.js` ning vastav testpäring. Kui ühendus edukalt toimis, alustati põhifunktsionaalsuse arendamisega.

4.3 Esirakendus

Esirakendus loodi Vue.js raamistikus ning koosneb mitmest eraldi komponendist ja konfiguratsioonifailist, mis täidavad erinevaid funktsionaalseid eesmärke.

Fail `auth.js` sisaldab JWT-dekodeerimise meetodeid, et tuvastada autentitud kasutaja roll ja ID. Failis `router/index.js` määratletakse rakenduse vaated ja marsruudid.

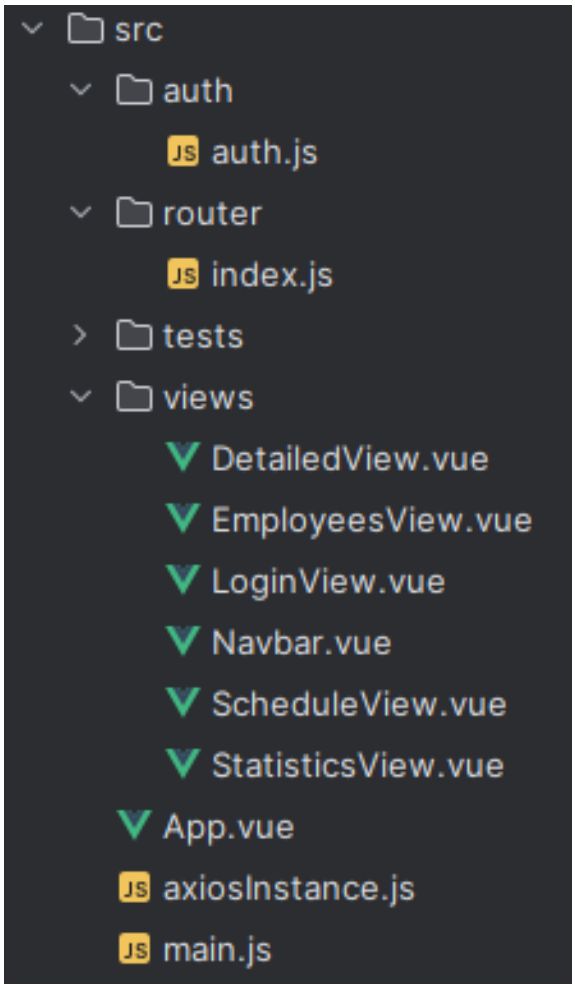
Fail `axiosInstance.js` haldab HTTP-päringute tegemist tagarakendusele. Selle kaudu lisatakse automaatselt iga päringu päisesse kasutaja autentimise käigus saadud JWT, et säilitada sessiooni turvalisus.

Failid`main.js` ja `App.vue` täidavad rakenduse initsialiseerimise ja põhikomponendi rolli. `Main.js` ühendab kokku kõik vajaliku (routeri, axiosi, moodulid), samas kui `App.vue` määratleb rakenduse üldise struktuuri ja sisaldab `<router-view>` elementi, kuhu vaated dünaamiliselt laetakse.

Vaadete kaustas asuvad .vue failid, mis vastavad iga ekraanivaate loogikale (näiteks statistika, töögraafik ja ülejäänud), samuti `Navbar.vue`, mis võimaldab kasutajal rakenduses mugavalt navigeerida.

Iga .vue fail järgib Vue komponendi kolmeosalist struktuuri - `<template>`, `<script>` ja `<style>` - mis tagab koodi loetavuse, hooldatavuse ja loogilise eristuse.

Esirakenduse struktuur on visuaalselt vaadeldav joonisel 1.



Joonis 1. Esirakenduse struktuur.

4.4 Tagarakendus

Tagarakendus on arendatud klassikalise kontroller-teenus-hoidla (Controller–Service–Repository) mustri järgi. Selle üheks suurimaks tugevuseks on selge vastutusjaotus:

- **Kontrollerikiht** haldab välist suhtlust, näiteks esirakenduse või teiste teenustega.
- **Teenusekiht** sisaldab kogu äriloogikat nagu arvutused, andmete kogumine, objektide loomine ja koondamine.
- **Andmekiht** vastutab andmete pärimise eest otse andmebaasist.

Andmeid hangitakse hoidlatest SQL-päringute kaudu ja teisendatakse andmeobjektideks (näiteks `WorkEventRow`). Teenusekiht töötleb neid edasi, arvutades väärtusi, mida andmebaasist otse ei saa (näiteks summeeritud ajad või tingimuslikud võrdlused). Lõpuks teisendatakse andmed sobivasse andmestruktuuri, mis saadetakse läbi kontrolleri

esirakendusele.

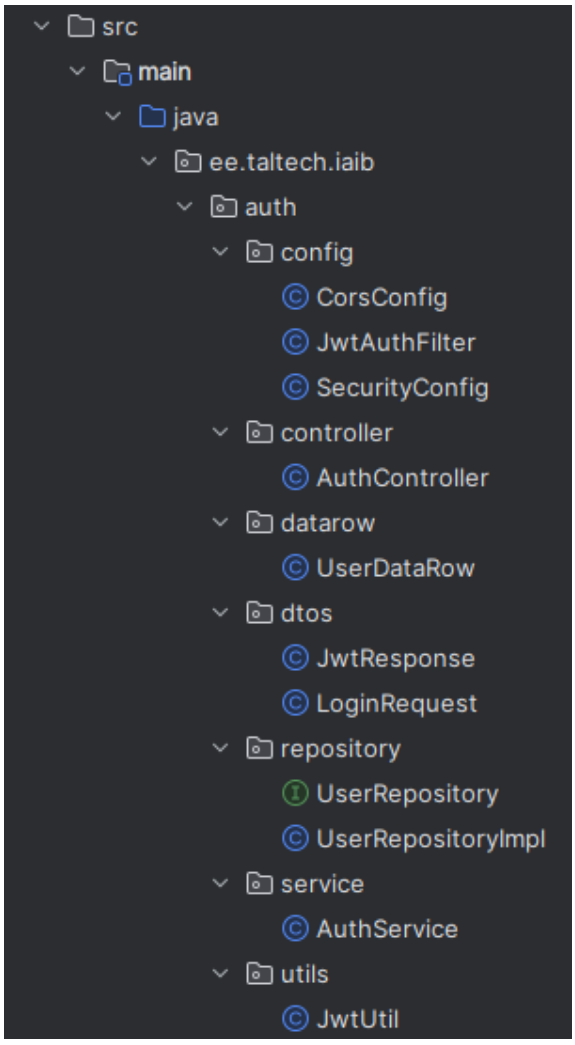
Tagarakendus koosneb kolmest peamisest komponendist: autentimine ja autoriseerimine, andmebaasiga ühendus ning põhiloogika.

4.4.1 Autentimine ja autoriseerimine

Selle alamsüsteemi kutsub välja esirakenduse sisselogimisvaade. Kasutaja sisestab kasutajanime ja salasõna, mis edastatakse tagarakendusele lihtteksti-kujul. Hoidla otsib kasutajanime alusel vastavat kirjet, ning kui see leitakse, võrreldakse sisestatud salasõna andmebaasis oleva krüpteeritud parooliga. Kasutaja sisselogimise kontod tulevad samas serveris asuvast andmebaasist `OeeMaster`. Kui salasõnad ühtivad, genereeritakse JWT, millele lisatakse väljad nagu kasutajanimi, kasutaja ID ning roll.

Seda märgist kasutab esirakendus igas järgnevas API-kutses, lisades selle HTTP päisesse. Märgised kehtivust ja sisu kontrollib `JwtAuthFilter`, mis kontrollib ka aegumisaega ja struktuuri. Turvapoliitika on konfigureeritud Spring Security kaudu, lubades anonüümset ligipääsu ainult `/auth/**` teekonnale. Vale andmete korral visatakse erind ja tagastatakse veakood koos teatega.

Tagarakenduse autentimise ja autoriseerimise osa struktuur on visuaalselt esitatud joonisel 2.



Joonis 2. Tagarakenduse autentimise ja autoriseerimise struktuur.

4.4.2 Andmebaasiga ühendus

Andmebaasiühendus on realiseeritud Docker Compose'i ja `application.yml` konfiguratsioonifailide abil. Kuna rakendus kasutab kahte eraldi andmebaasi - `sigmaLPS` ja `TimeMaster` - loodi spetsiaalne `DataSourceConfig`-klass, mis määratleb iga andmebaasi jaoks oma `JdbcTemplate`-beanid. See võimaldab hoidla-klassidel kasutada õiget andmeallikat sõltuvalt andmetüübist.

4.4.3 Põhiloogika ja vaated

Iga esirakenduse põhivaade omab vastavat loogikat tagarakenduses.

Detailivaade edastab päringus kasutaja ID ja kuupäeva, mille alusel leitakse kõik vastava päeva sündmused. Need loetakse andmebaasist sisse-välja liikumise paaridena, võimal-

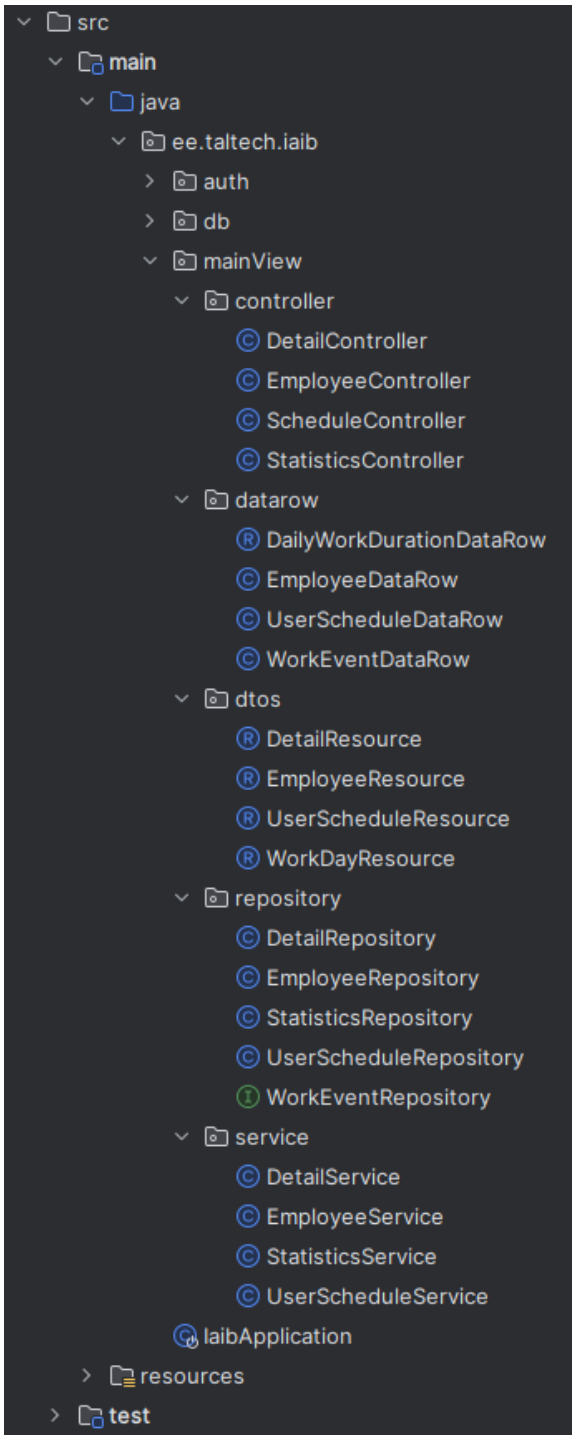
dades teenusekihil arvutada tööaega, pause ja isiklikke katkestusi. Tulemus esitatakse struktureerituna esirakenduse tarbeks selleks, et hoida kogu äriloogika tagarakenduses.

Graafikuvaade töötab sarnaselt: päring sisaldab kasutaja ID-d ja kuud, ning teenus kombineerib info töögraafikutest ja detailandmetest. See võimaldab näidata planeeritud ja tegelikku tööaega koos summeeritudd tööaja, pauside ja isikliku ajaga.

Töötajate vaates on kaks API-päringut: esimene toob välja kõik aktiivsed töötajad, teine tagastab töötaja täisnime antud ID põhjal.

Statistikavaade tagastab töötatud tundide koondtabeli kuu lõikes. Päring sisaldab kasutaja ID-d ja kuud ning tagastab iga päeva kohta töötundide arvu.

Tagarakenduse põhiloogika struktuur on esitatud joonisel 3.



Joonis 3. Tagarakenduse põhiloogika struktuur.

4.5 Andmebaas

Loodav rakendus kasutab sama MS SQL-andmebaasi, mida kasutas varasem süsteem TimeMaster. Kuna ettevõtte andmebaasiklaster sisaldab mitmeid andmebaase, keskenduti arenduses ainult kahele rakenduse toimimiseks vajalikele andmebaasile.

Esimene ja peamine andmebaas on `sigmaLPS`, mis sisaldab rakenduse põhifunktsionaalsuseks vajalikke tabeleid, sealhulgas töögraafikute, detailide, kasutajarollide ja muude loogiliste üksuste andmeid. Teine kasutatav andmebaas on `TimeMaster`, mis sisaldab autentimisega seotud andmeid, näiteks kasutajanimed ja salasõnad.

Arenduskeskkonnas kasutati ainult neid kahte andmebaasi, mille tarbeks loodi lokaalne versioon. Selleks imporditi vajalikud andmestruktuurid SQL-failide abil Microsoft SQL Server Management Studio (SSMS) tööriista kaudu. Kuigi kaaluti ka otseühendust ettevõtte kaugandmebaasiga, loobuti sellest VPN-i ja ligipääsuõigustega seotud võimalike probleemide tõttu.

Kohaliku andmebaasiversiooni kasutamine võimaldas arendajal luua tagarakenduse jaoks vajalikud SQL-päringud ning testida rakenduse ja andmebaasi vahelist andmevahetust turvalises ja kontrollitud keskkonnas.

Üks tähtsamaid andmebaasitabeleid on `OSYNDMUS`, mis salvestab sündmuste logi, näiteks töötajate liikumised ja sündmused kindlatel kellaaegadel. Alljärgnev SQL-kood näitab selle tabeli loomist arenduskeskkonnas:

Listing 4.1. Tabeli OSYNDMUS loomise SQL-kood

```
CREATE TABLE [dbo].[OSYNDMUS](
    [Paev] [int] NOT NULL,
    [Kell] [int] NOT NULL,
    [Kontroller] [int] NOT NULL,
    [Lukk] [int] NOT NULL,
    [Syndmus] [int] NULL,
    [Suund] [smallint] NULL,
    [Isik_id] [int] NULL,
    [Kl_kood] [smallint] NOT NULL,
    [Aeg] [datetime] NULL,
    [Tegi] [int] NULL,
    [Tehtud] [datetime] NULL,
    [Tyhist] [smallint] NULL,
    [EKSPORT] [datetime] NULL,
```

```
    CONSTRAINT [PK_OSYNDMUS] PRIMARY KEY CLUSTERED
    (
        [Paev] ASC,
        [Kell] ASC,
        [Kontroller] ASC,
        [Lukk] ASC
    )
)
```

Antud tabeli primaarvõti koosneb neljast väljast `Paev`, `Kell`, `Kontroller` ja `Lukk`, mis tagavad sündmuse unikaalsuse. Väljad nagu `IsikId`, `Syndmus` ja `Suund` salvestavad sündmuse tähenduse ning seotud isiku. Tabelit kasutatakse nii logimiseks kui ka tööaja analüüsiks.

4.6 Koodi ettevalmistamine tootmiskeskkonnaks

Alguses loodi rakendus kahes eraldi osas ning kahe eraldi projektina. Hiljem otsustati koodi haldamise lihtsustamiseks need ühendada üheks projektiks nimega `iaib`. Selle ühendamise käigus säilitati kogu senine GitLabi commit'i ajalugu, kandes mõlema rakenduse koodi uude projekti eraldi alamkaustadesse: `frontend` ja `backend`.

Lisaks olemasolevatele kaustadele lisati projekti vajalikud konfiguratsioonifailid. Kuna tundlikke andmeid ei ole turvaline salvestada otse koodi, kasutatakse selleks `.env`-faili, mille väärtusi täidetakse lokaalsetes ja tootmiskeskkondades vastavalt vajadusele.

Et arendajatel oleks mugavam oma `.env` fail luua, lisati ka näidisfail nimega `.env.example`, kus on loetletud kõik vajalikud võtmed ja muutujad. Faili `.gitignore` kaudu tagatakse, et tegelikud tundlikud andmed (`.env`) ei satu kogemata versioonihaldusesse.

Näide keskkonnamuutujatest `.env` failis:

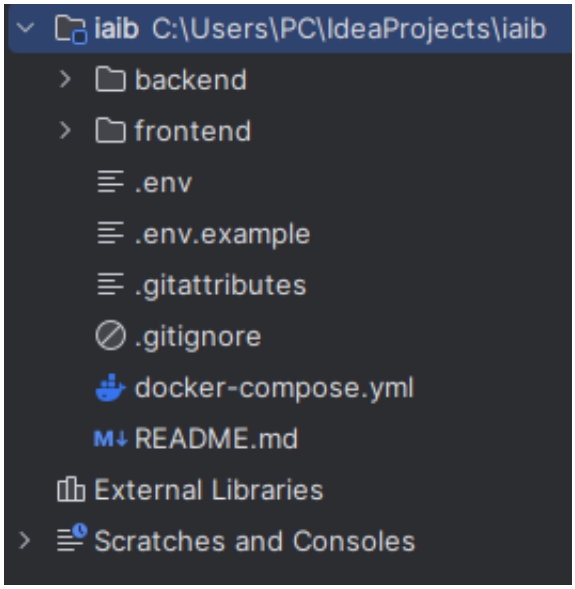
```
DB_SA_PASSWORD=
JWT_SECRET=
ENCRYPTION_KEY=
TIMEMASTER_PASS=
SIGMA_PASS=
FRONTEND_ORIGIN=
JWT_EXPIRATION=
```

4.6.1 Dockeri kasutuselevõtt

Uues projektis kasutatakse konteinerpõhist arhitektuuri, mille keskseks komponendiks on `docker-compose.yml` fail. Selle kaudu kirjeldatakse kogu süsteemi ülesehitus: andmebaas, taustateenus ja kasutajaliides. Igas alamkaustas (`backend` ja `frontend`) asub oma `Dockerfile`, mis määrab vastava komponendi ehitamise ja jooksutamise loogika.

Viimaks lisati `README.md` fail, kus on kirjeldatud rakenduse käivitamise nõuded ning juhend.

Lõpliku rakenduse struktuuri on võimalik vaadelda joonisel 4.



Joonis 4. Loodud rakenduse lõplik struktuur.

Gitlab'i link projektile on leitav Gitlabi link on leitav Lisas 2.

5 Valideerimine

Rakenduse valmimisel on oluline etapp tehtud töö valideerimine ehk testimine, et tagada süsteemi korrektsus ja töökindlus.

5.1 Lokaalsed testid

Lokaalsed testid on olulised arendusprotsessi varases staadiumis. Neid saab teostada arendaja masinas, ilma et oleks vaja ligipääsu tootmiskeskkonnale. Need jagunevad üldjoontes kaheks:

- Automatiseeritud (kirjutatud) testid, mis valideerivad loogikat ja funktsionaalsust.
- Kasutajakogemuse testid, mis aitavad hinnata liidese intuitiivsust ja kasutatavust.

5.1.1 Kirjutatud testid

Kirjutatud testid on integreeritud rakenduse koodibaasi ja neid käivitatakse automaatselt või käsitsi arendaja poolt. Need aitavad kindlustada, et koodi muudatused ei põhjusta probleeme.

Tagarakenduses on testid jagunevad kaheks: üksustestid ja komponenditestid.

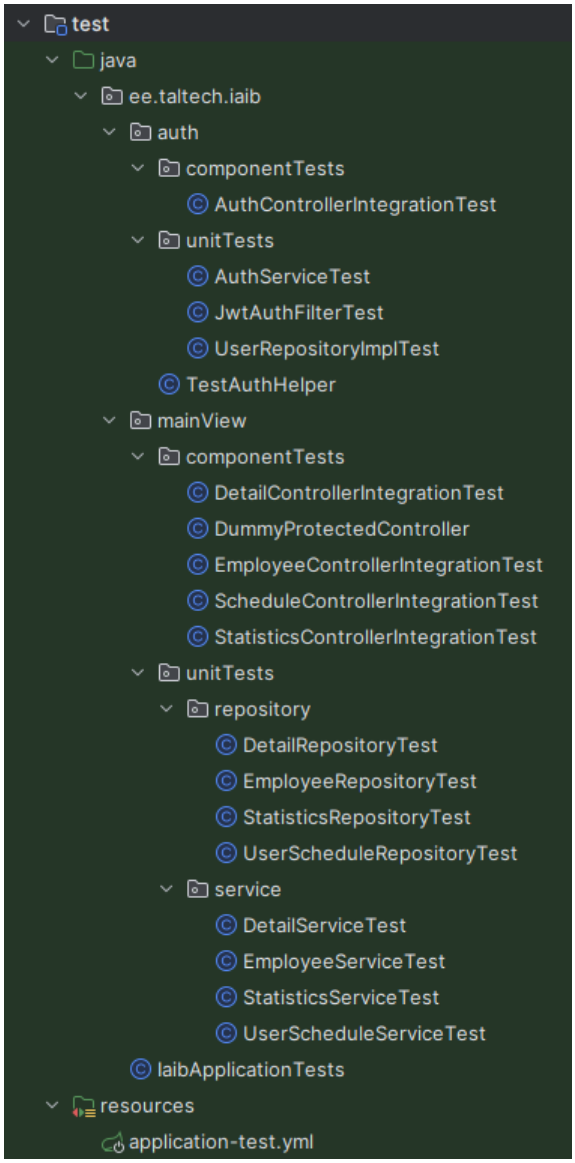
Üksustest kontrollivad individuaalseid funktsioone, loogikaplokke või arvutusi. Need on kiired ja isoleeritud testid, mis võimaldavad arendajatel kiiresti tuvastada vigu konkreetsetes osades.

Komponenttestid katavad suuremaid funktsionaalseid plokke ning valideerivad protsesside terviklikku toimimist. Need testid jäljendavad kasutajateekonda ning aitavad kindlaks teha, kas erinevad komponendid töötavad koos ootuspäraselt.

Testide läbiviimiseks kasutatakse eraldi konfiguratsioonifaili `application-test.yml`, mis määrab testikeskkonna seadistused. Selle eesmärk on isoleerida testkeskkond arenduskeskkonnast, võimaldades testide turvalist ja stabiilset käivitamist.

Fail sisaldab eraldiseisvaid ühendusandmeid andmebaasidele, JWT turvaseadeid ning esirakenduse URLi. Samuti määratakse seal andmete krüpteerimiseks kasutatav parool. See lähenemine tagab, et testide käigus ei mõjutata tootmisandmeid ning testid on taaskäivitatavad samade tingimustega.

Testide ülesehitust tagarakenduses illustreerib joonis 5.



Joonis 5. Tagarakenduse autentimise ja autoriseerimise struktuur.

5.1.2 Kasutajakogemuse testid

Pärast rakenduse arendamist ja automatiseeritud testide läbimist on oluline teostada ka manuaalne funktsionaalne testimine lokaalses keskkonnas. Selle käigus käivitatakse rakendus arendaja arvutis, kasutades spetsiaalselt selleks loodud testandmeid lokaalses

andmebaasis. Parima tulemuse saavutamiseks peaksid testandmed olema võimalikult lähedased reaalsele andmestikule, et simuleerida päriselu olukordi.

Testimise käigus: Avatakse rakendus brauseris (nt `http://localhost:3000`) ning liigutakse läbi erinevate vaadete ja funktsionaalsuste.

Kontrollitakse, kas süsteem käitub ootuspäraselt: näiteks, kas andmete sisestamine toimib, kas valideerimised toimivad õigesti, kas andmebaasist küsitakse infot korrektselt ja kas loogilised arvutused (nt koguarvutused, filtreerimised, staatused) töötavad nõuetekohaselt.

Lokaalsel testimisel tehtud sammud:

1. Loodi testandmed, mis sisestati lokaalsesse andmebaasi Microsoft SQL Server Management Studio abil.
2. Rakendus käivitati käsuga `docker-compose up`.
3. Kui rakenduse käivitamine toimus tõrgeteta, alustati manuaalse testimisega.
4. Testimise käigus kontrolliti vaadete vahetumist, rakenduse erinevate osade vahelist suhtlust, andmetega tehtud arvutusi ning visuaalsete komponentide korrektsust.

Selline testimine võimaldab hinnata rakenduse kasutatavust ja töövoogude terviklikkust olukordades, mis sarnanevad reaalsele kasutuskeskkonnale. Lisaks aitab see avastada probleeme, mida automatiseeritud testid ei pruugi katta, näiteks liidese reageerimiskiirus, erinevate komponentide koostoime või ootamatu kasutajakäitumise mõju süsteemile.

Manuaalne testimine lokaalselt võimaldab ka visuaalselt hinnata andmete esitamist ning komponentide paiknemist lõppkasutaja vaates, muutes rakenduse kontrollimise intuitiivsemaks ja praktilisemaks.

5.2 Tellija tagasiside

Ideaalis tuleks rakendust pärast lokaalset testimist testida ka tootmiskeskkonnas, kuna see võimaldab kontrollida süsteemi toimimist pärisandmete põhjal ning tuvastada võimalikke probleeme, mis võivad ilmneda ainult reaaltingimustes.

Kahjuks ei olnud tootmiskeskkonnas testimine võimalik, kuna rakenduse testimisfaas langes ajaliselt kokku ettevõttes läbiviidavate auditite ga ja õppustega. Seetõttu jäi täie-

lik tootmiskeskkonna testimine semestri jooksul tegemata, kuid toimub hiljem kõigile osapooltele sobilikul ajal.

Sellegipoolest saatis tellija:

- Ekraanipilte ja vaateid, mis olid genereeritud andmebaasis sisalduvate pärisandmete põhjal.
- SQL-päringu, mis oli varem kasutusel Exceli tabelis andmete kuvamiseks.

Antud abimaterjalid aitasid parandada andmebaasist info küsimist, tööaegadega tehtud arvutuste kontrollimist ning aitasid osaliselt asendada tootmiskeskonna testimist.

5.2.1 Ettevõttepoolne tagasiside

Manuaalse testimise käigus loodi demovideod, mis edastati ettevõtte esindajale. Nende ja mõningate täpsustavate küsimuste põhjal saadi järgmine tagasiside:

1. Navigatsiooniriba võiks olla visuaalselt eraldatud näiteks varjundiga, et see ei sulanduks muude valgete aladega.
2. Graafiku- ja detailtabelites võiks üle- ja alatunnid olla paremini visuaalselt eristatud.
3. Tööaja väärtused võiks esitada tundide ja minutite formaadis, mitte ainult minutites.
4. Päeva esimene ja viimane liikumine võiks olla silmatorkavamalt esile toodud.
5. Pausiruumiks loetakse "Suitsuruum"ja "Söögisaal". Isikliku aja alla loetakse päeva keskel toimuvad "Peauksest"väljumised näiteks arstivisiidid.
6. Andmete eksportimise võimalus võiks olla .xlsx ja .pdf vormingus, .csv pole hetkel vajalik.

5.2.2 Tehtud muudatused

Ettevõtte tagasiside alusel viidi rakenduses sisse järgmised muudatused:

1. Navigatsiooniriba taust muudeti tumesiniseks ja tekst valgeks, mis eristab selle visuaalselt ülejäänud leheküljest.
2. Ületundide ja alatundide lahtrite taustad värviti vastavalt roheliseks ja punaseks. Samuti muudeti detailtabeli ridade tausta veergudes „Tööl”, „Pausil” ja „Isiklik”, kui vastavates lahtrites ei olnud nullväärtust ehk ".

3. Ajaväljad teisendati vormingusse hh:mm, et väärtused oleksid kasutajale selgemini loetavad.
4. Päeva esimene ja viimane liikumine märgiti vastavalt kollase triibuga kellaja juures ning rohelise ja punase ringiga sündmuse nimetuse kõrval.
5. Pausiruumide loogika muudeti vastavalt täpsustusele ning muudeti sellega seotud arvutused.
6. Andmete eksportimine on nüüd võimalik nii .xlsx kui ka .pdf vormingus. Allalaetavaid faile saab lihtsalt edasi jagada teiste isikutega.

6 Kasutajakogemus

On tähtis, et planeerimise tööriistad teenindaksid kasutajat, mitte vastupidi. Tööriistad peaksid sobituma kasutajale, sobima erinevate seadmetega ning olema kergesti kättesaadavad [23]. Seega peab lõpptulemus olema eelkõige mugav ja arusaadav kasutada andes kasutajale selle, mida nad vajavad. Järgnevalt lahatakse loodud rakenduse kasutajavoogu vaade haaval, keskendudes funktsinaalusesele ning kasutajakogemusele

6.1 Sisselogimine

Kasutaja esimene kokkupuude rakendusega toimub sisselogimislehel, mis on kujundatud puhtalt ja professionaalselt (joonis 6). Liideses on ainult kaks sisestusvälja - kasutajanimi ja salasõna - ning selgelt rohelisega esile tõstetud „Logi sisse“ nupp. Selline minimalistlik lähenemine aitab vältida kasutaja liigset koormamist üleliigse teabega ning toetab fokusseeritud kasutusmugavust.

Kuna kasutajakontod luuakse ja hallatakse ettevõtte teises süsteemis Argoses, ei ole vajadust rakenduses eraldi registreerimislehe järele. Kasutaja sisestab autentimiseks OEEs kehtiva kasutajanime ja parooli.

Väljadele on lisatud sisendivihjed, mis aitavad kasutajal mõista, millist infot tuleb sisestada. Vorm toetab klaviatuuri kasutamist – kasutaja saab liikuda väljade vahel `Tab`-klahviga, mis muudab kasutamise kiireks ja sujuvaks. Samuti on võimalik sisselogimise nuppu aktiveerida `Enter`-klahviga.

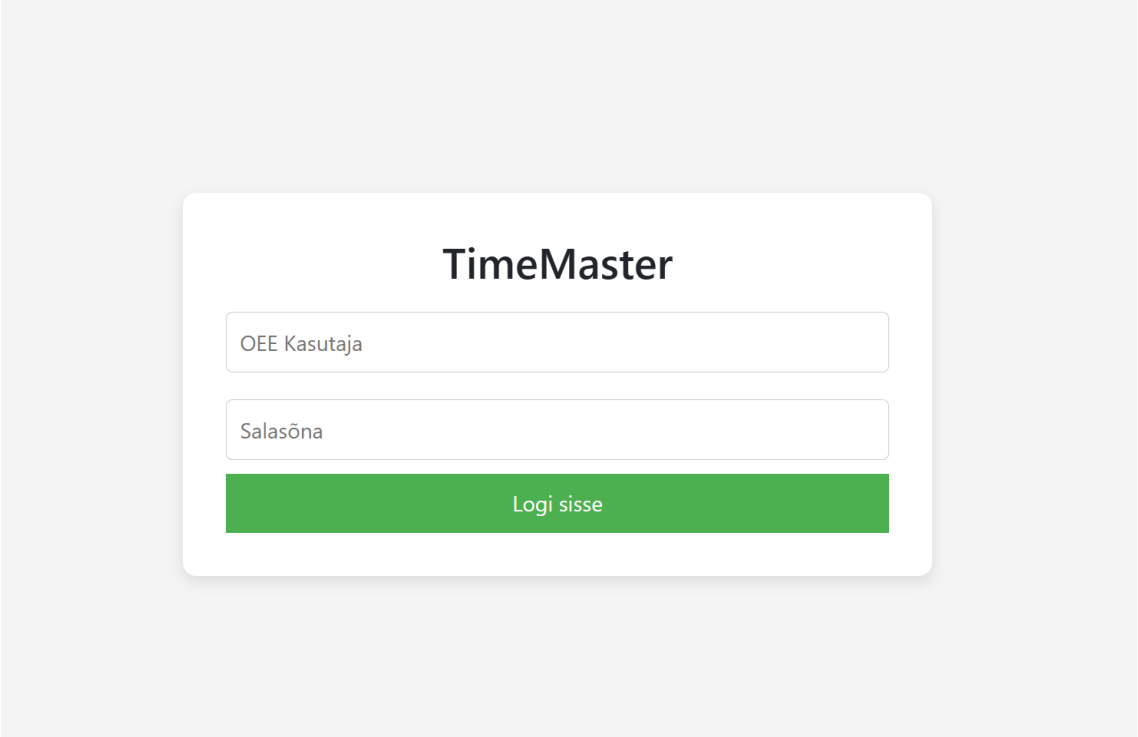
Kasutajakogemuse parandamiseks on rakendatud järgmised sisendkontrollid:

- Kui mõni väli jäetakse täitmata, kuvatakse vastav hoiatussõnum.
- Kui sisestatud kasutajat ei leita või parool ei vasta salvestatud väärtusele, kuvatakse selgesõnaline veateade (nt „Vale kasutajanimi või parool“).

Hoiatussõnumid on vastavalt oranži ning punase värvusega, et kasutajale rohkem silma

paista.

Sisselogimisprotsessi tulemusena saadetakse kasutajalt sisestatud andmed tagarakendusele, kus toimub autentimine. Eduka sisselogimise korral väljastatakse kasutajale JWT, mis salvestatakse brauseri mällu, mille alusel toimub edasine suhtlus rakendusega.



Joonis 6. Rakenduse sisselogimisleht.

6.2 Navigatsiooniriba

Rakenduse eri vaateid ühendab ülevalt fikseeritud navigatsiooniriba, mis võimaldab kasutajal liikuda kiiresti eri funktsionaalsuste vahel (joonis 7).

Navigatsioonimenüü vasakus servas asuvad kolm peamist linki: **Graafik**, **Töötajad** ja **Statistika**. Neist igaüks suunab kasutaja vastavale lehele. Päevaste detailide vaade ei ole otse navigatsiooniribalt ligipääsetav. Tegemist on teadliku disainivalikuga, kuna vaade nõuab kuupäevapõhist konteksti, mida ei ole otstarbekas küsida otse navigeerimismenüüst. Selle asemel suunatakse kasutaja esmalt graafikuvaatesse, kus tal on mugavam sobiv päev visuaalselt üles leida (näiteks töörežiimi või töötundide põhjal) ning sealt edasi detailandmeid avada.

Aktiivne vaade on kujunduslikult esile tõstetud: valitud vaate nimi kuvatakse paksus kirjas

ning selle alla ilmub peen valge joon nagu on näha joonisel 7 graafiku vaate kohta. See aitab kasutajal kiiresti orienteeruda ja mõista, millises rakenduse osas ta hetkel asub.

Paremas servas paikneb Logi välja nupp, millel on valge raam ja ümarad nurgad. Nupp erineb stiililt ülejäänud menüüelementidest, et kasutaja suudaks selle kiiresti leida. Kui kasutaja viib hiire nupu kohale, muutub selle taust siniseks – visuaalne vihje, et tegemist on interaktiivse elemendiga. Nupule vajutamisel logitakse kasutaja välja, eemaldatakse JWT brauseri mälust ning suunatakse tagasi sisselogimislehele.

Joonis 7. Rakenduse navigatsiooniriba koos aktiivse vaate ja väljalogimise nupuga.

6.3 Graafiku vaade

Pärast kasutaja sisselogimist kuvatakse vaikimisi tema isiklik graafik. Selline valik on tehtud põhjusega, kuna graafiku lehekülg on tõenäoliselt enim kasutatav vaade, kust töötaja vajalikku infot otsib.

Vaate vasakus ülanurgas on nähtav, kellele graafik kuulub (näiteks Aadam Mägi graafik). Kuigi see ei pruugi tavakasutajale eriliselt kasulik olla, aitab see nii juhtidel kui ka administraatoritel hõlpsamini erinevate töötajate vaadete vahel navigeerida.

Järgmisena on kuvatud kuupäevavalik kahe rippmenüü kujul, kus saab valida kuu ja aasta. Aastate loend sisaldab kõiki aastaid, mille kohta on andmeid, kuni käesoleva aastani. Vaikimisi on valitud jooksva kuu ja aasta vaade, kuna see on tavaliselt kõige olulisem ja aktuaalsem.

Lisaks on olemas võimalus eksportida tabel nii PDF kui ka XSLX vormingus. See aitab vajadusel graafikut töötajatega jagada.

Vaate keskseks elemendiks on tabel ise, mis sisaldab järgmisi välju:

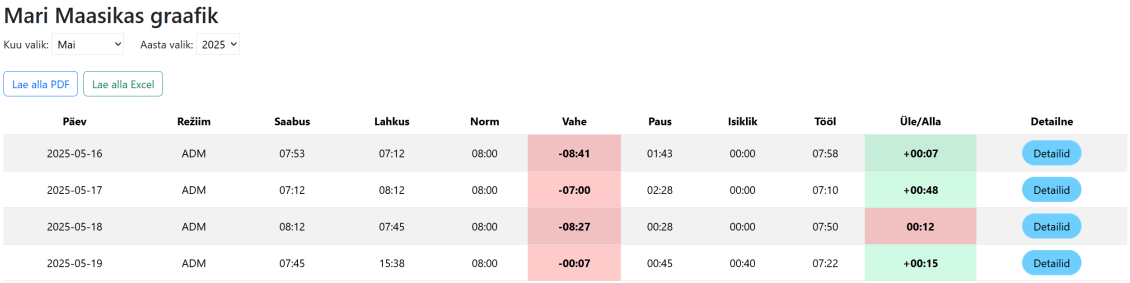
- **Päev** - Töötamise kuupäev vormingus `dd.mm.yyyy`
- **Režiim** - Näitab, mis režiim töötajal antud päeval on näiteks P - puhusel või TÕh - tootmine õhtul

- **Saabus** - Kellaaeg, millal töötaja esimest korda sel päeval peauksest sisenes
- **Lahkus** - Kellaaeg, millal töötaja viimast korda sel päeval peauksest lahkus
- **Norm** - Ettenähtud tööaeg päeva kohta (8 tundi koos pausidega; 7 tundi ja 40 minutit ilma pausideta)
- **Vahe** - Töötatud aeg miinus norm
- **Paus** - Kogusumma ajast, mis veedeti puhkepausidel
- **Isiklik** - Aeg, mis veedeti väljaspool tööülesandeid isiklikel põhjustel
- **Tööl** - Aeg, mis veedeti aktiivselt tööülesandeid täites (ei sisalda pause ega isiklikku aega)
- **Üle/Alla** - Saabumisaeg miinus tavaalgusaeg
- **Detailne** - Nupp, mis viib konkreetse päeva detailse vaate juurde

Tabeli päised on esile toodud paksus kirjas, et neid oleks visuaalselt kergem eristada. Samuti on päevade read kujundatud vahelduva hallika taustaga, mis lihtsustab erinevate ridade eristamist. Kõik ajad on vormindatud ühtselt kas `hh:mm:ss` või `hh:mm` formaadis.

Kui veerus **Üle/Alla** on väärtus suurem kui null, kuvatakse vastava kasti taust rohelisena. Kui väärtus on alla nulli, on taust punane. See visuaalne eristus annab kasutajale kiire ülevaate kuu tööseisust.

Nupp **Detailne** on kujundatud helesinises toonis, et see eristuks ülejäänud tekstist ning viitaks võimalusele vaadata lisainfot.



Joonis 8. Rakenduse graafiku vaade.

6.4 Päeva sündmuste vaade

Kui kasutaja vajutab graafiku vaates nupule "Detailid", suunatakse ta vastava päeva detailsele vaatele.

Vaate vasakus ülanurgas on suurelt kuvatud pealkiri "Detailne päeva vaade"ning selle all kuupäev formaadis `dd-mm-yyyy`. Lisaks on olemas võimalus eksportida tabel nii PDF kui ka XSLX vormingus. See aitab vajadusel graafikut töötajatega jagada.

Vaate keskseks elemendiks on tabel, mis kajastab päeva jooksul toimunud liikumisi. Tabel sisaldab järgmisi veerge:

- **Kell** – Sündmuse kellaaeg
- **Uks** – Ukse nimetus, kust liikumine toimus
- **Suund** – Liikumise suund (sisse/välja)
- **Sündmus** – Toimunud sündmus (nt võtmega avamine)
- **Tööl** – Aeg, mis on möödunud tööl olekust alates eelmisest töölt lahkumisest
- **Pausil** – Aeg, mis on möödunud puhkeajal olekust alates eelmisest pausilt naasmisest
- **Isiklik** – Aeg, mis on möödunud isiklikul ajal olekust alates eelmisest tööle naasmisest
- **Eemal kokku** – Summaarne aeg, mis on seni töölt eemal oldud (pausid + isiklik aeg)

Kõik kellaajad on esitatud vormingus `hh:mm`. Päeva esimene ja viimane liikumine on tähistatud vasakul kollase joonega ning sündmuse juures on visuaalselt eristuv punane või roheline ring, et tõsta esile nende olulisust.

Suund veerus on kõik töö-, pausi- või isikliku aja algused tähistatud rohelise tausta ja paksu kirjaga, kui tegemist on sisenemisega. Väljumised on märgitud punase taustaga. Tavalised liikumised tööalaste ruumide vahel (nt tootmisest fuajeesse) on esitatud neutraalse valge/halli taustaga ja tavakirjas.

Veergudes **Tööl**, **Pausil** ja **Isiklik** on erinevad perioodid visuaalselt eristatud värviga:

- Tööl – roheline taust
- Pausil – sinine taust
- Isiklik – kollane taust

Eesmärk on muuta jälgimine lihtsamaks ja visuaalselt arusaadavamaks.

Eriloogika puhkeruumide puhul

Algandmete põhjal tähistas liikumine puhkeruumidesse (näiteks söögisaali) algselt "väljumist"ning seejärel "sisenemist", kuna töölt eemalduti. Et kasutajale oleks loogika arusaadavam, muudeti rakenduses puhkeruumide liikumiste suunda nii, et liigutakse puhkeruumi "sisse"ja seejärel "välja", mis vastab füüsilisele tegelikkusele. **Isikliku aja** arvestus jääb siiski samaks: liikumine algab peauksest väljumisega ning lõpeb peauksest sisenemisega.

Aja arvestuse loogika

Veergudes **Tööl**, **Pausil** ja **Isiklik** ei ole näidatud arvud kumulatiivsed. Iga rida esitab konkreetset sessiooni kestvust – s.t ajavahemikku ühe tegevuse algusest selle lõpuni.

- Kui kasutaja läheb tööle, hakkab n-ö jooksma tööaeg.
- Kui minnakse pausile (näiteks suitsuruumi sisenemisel), peatub tööaja arvestus. Pausile mineku reale lisatakse selleks hetkeks kogunenud tööaeg.
- Kui paus lõpeb (st puhkeruumist väljutakse), arvutatakse pausi kestus ning lisatakse see vastavale reale. Samal hetkel alustatakse uut tööaja arvestust.

Näiteks joonisel 9 lahkub kasutaja kell 09:47 söögisaalist ning suundub isiklikule ajale majast välja kell 17:22. Selle vahemiku jooksul oldi tööl **7 tundi ja 35 minutit**, mis on kajastatud peauksest väljumise rea **Tööl** lahtris.

Detailne päeva vaade

2025-05-19

Laadi alla PDF | Laadi alla Excel

Kell	Uks	Suund	Sündmus	Tööl	Pausil	Isiklik	Eemal kokku
07:45	Peauks	**Sisse**	Võtmega avamine	-	-	-	-
08:01	Tootmine	Sisse	Võtmega avamine	-	-	-	-
10:15	Tootmine	Välja	Võtmega avamine	-	-	-	-
10:45	Suitsuruum	Sisse	Võtmega avamine	03:00	-	-	-
10:55	Suitsuruum	Välja	Võtmega avamine	-	00:10	-	00:10
12:00	Söögisaal	**Sisse**	Võtmega avamine	01:05	-	-	-
12:35	Söögisaal	**Välja**	Võtmega avamine	-	00:35	-	00:45
12:37	Fuajee koridor	Sisse	Võtmega avamine	-	-	-	-
14:45	Fuajee koridor	Välja	Võtmega avamine	-	-	-	-
14:58	Peauks	**Välja**	Võtmega avamine	02:23	-	-	-
15:38	Peauks	**Sisse**	Võtmega avamine	-	-	00:40	01:25
16:32	Peauks	**Välja**	Võtmega avamine	00:54	-	-	-

Tagasi

Joonis 9. Rakenduse graafiku päeva detailvaade.

6.5 Töötajate vaade

Töötajate vaade kuvab kõik ettevõttes aktiivsed töötajad.

Ekraani vasakus ülanurgas asub pealkiri **"Töötajad"**. Selle all paiknevad kolm kõrvuti asetsevat otsingu- ja sorteerimisriba. Kõige vasakpoolsem on rippmenüü, mis võimaldab valida otsingu tüübi. Otsingut saab teha järgmiste parameetrite alusel: **Nimi**, **ID**, **Grupp** ja **Otsese juhi ID**. Vaikimisi on valitud otsingutüübiks **Nimi**, kuna see on kasutajale kõige intuitiivsem ja tuntum parameeter.

Tüübi valiku kõrval asub otsingukast, kuhu saab sisestada soovitud väärtuse. **ID** ja **Otsene juht ID** järgi otsides leitakse ainult täpne vaste. **Nime** ja **Grupi** järgi otsides leitakse kõik tulemused, mis sisaldavad sisestatud otsingusõnet (näiteks otsing "ko"leiab nii grupid "Kollane"kui ka "Kontor").

Kolmas väli on sorteerimise rippmenüü, mille kaudu saab valida tabeli järjestamise viisi. Valikus on kuni kaheksa sorteerimise võimalust: **ID**, **Nimi**, **Grupp** ja **Otsene juht ID** järgi. Igaüks neist kas kasvavas või kahanevas järjestuses (sh tähestikulises või vastupidises tähestikulises järjekorras).

Tabeli ülesehitus

Tabel koosneb järgmistest veergudest:

- **ID** – Töötaja unikaalne identifikaator
- **Nimi** – Töötaja täisnimi
- **Grupp** – Töötaja kuuluvusgrupp
- **Otsene juht** – Töötaja otsese juhi ID
- **Graafik** – Nupp, mis viib vastava töötaja graafiku vaatele

Tabeli disain järgib graafiku vaate visuaalset stiili: päised on paksus kirjas, read vahelduva hallika taustaga ning **Graafiku** nupp on esile tõstetud helesinise värviga, et oleks kasutajale hästi nähtav.

Ligipääsuõigused

Nagu ka muudes rakenduse vaadetes, sõltub andmete nähtavus kasutaja rollist:

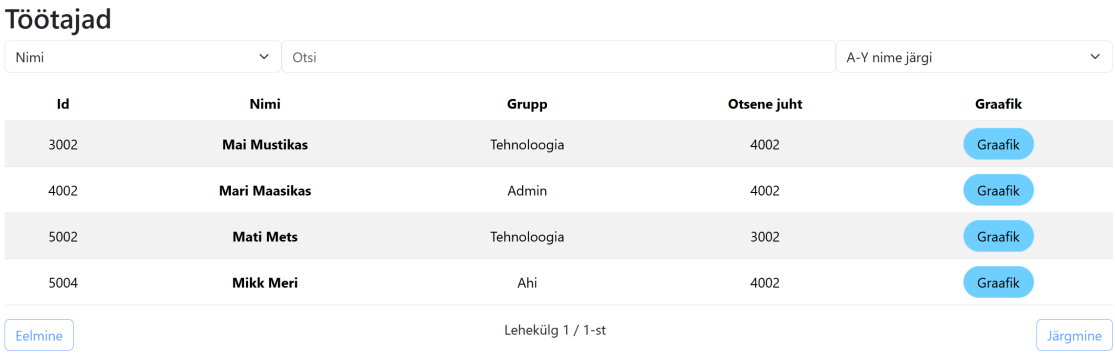
- **Tavatöötajad** ei näe teiste töötajate juures **Graafiku** nuppu. Kogu veerg on nende jaoks peidetud.
- **Otsesed juhid** näevad **Graafiku** nuppu ainult oma alluvate ridades.
- **Administraatorid** näevad **Graafiku** nuppu kõikide töötajate juures.

Kasutajarollide info saadakse JWT kaudu.

Lehekülgede haldus

Kuna töötajate arv on suur, on tabelile lisatud lehekülgede jagamine (pagination). Igal leheküljel kuvatakse kuni 12 töötajat. Tabeli allservas asuvad navigeerimisnupud:

- Vasakul **Eelmine** – liikumiseks eelmisele leheküljele
- Paremal **Järgmine** – liikumiseks järgmisele leheküljele
- Keskel kuvatakse leheinfo kujul: `Lehekülg X / Y`, mis näitab, mitmendal leheküljel parasjagu ollakse ja kui palju lehekülgi kokku on



Joonis 10. Rakenduse töötajate vaade.

6.6 Statistika vaade

Statistika vaade on ainulaadne ja uus funktsionaalsus, mida varasemas rakenduses ei eksisteerinud. Vaade koosneb kahest põhiosast: vasakul paiknevast graafikust ning paremal asuvast otsingu ja filtreerimise sektsioonist.

Ekraani parempoolses kolmandikus asub halli kastiga eraldatud otsingu ja filtreerimise paneel, mis tõstab selle visuaalselt esile. Kasti ülaservas on rippmenüü graafiku tüübi valikuks. Hetkel on saadaval vaade päevastest töötundidest, kuid tulevikus on plaanis lisada erinevaid statistika tüüpe.

Tüübi valiku all saab statistikat kitsendada kindlale kuule ja aastale. Nagu ka graafiku vaates, on valikus kõik kuud ja aastad, mille kohta on andmeid kuni käesoleva ajani. Vaikimisi kuvatakse hetke kuu ja aasta.

Kasti alumises osas paikneb töötajate valiku sektsioon. Seal asub otsinguriba, mille abil saab leida töötajaid, kelle andmeid soovitakse vaadelda ja/või omavahel võrrelda. Otsingu käivitamiseks tuleb sisestada vähemalt kaks tähte, mille põhjal kuvatakse sobivaimad tulemused. Valida saab kuni kolm töötajat korraga. Töötaja lisamiseks tuleb vajutada nuppu „Lisa“ tema nime juures.

Lisamisnupp on vaikimisi sinise tekstiga ja valge taustaga ning muutub hiirega peale liikudes valge tekstiga ja sinise taustaga, andes kasutajale visuaalse vihje vajutamiseks. Valitud töötajad kuvatakse nimekirjana otsingukasti all. Iga nime kõrval on punane nupp „Eemalda“, millega saab töötaja valikust eemaldada, misjärel kaob tema andmestik ka graafikult.

Kui kasutaja üritab valida rohkem kui kolm töötajat, kuvatakse punane hoiatus teavitusega, et enne tuleb mõni valik eemaldada. Piirang valida maksimaalselt kolm töötajat korraga tuleneb graafiku visuaalsest selgusest – see aitab vältida andmete ülekattumist ja säilitada head loetavust.

Ekraani vasakpoolsel kahel kolmandikul asub graafik, millele kuvatakse valitud info. Graafiku ülaservas, keskel, on suurelt ja paksus kirjas selgelt märgitud, millist infot joonis esitab. Paremas ülanurgas asub kolme joonega menüüikoon, mille kaudu saab graafiku eksportida `.png` vormingus – see võimaldab graafikut hõlpsasti salvestada ja teistega jagada.

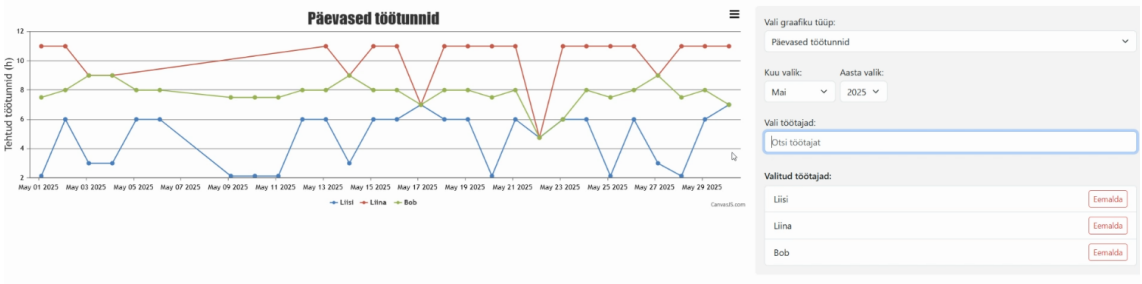
Graafiku vasak serv kujutab y-telge, millel on kuvatud mõõdetav info (nt „Tehtud töötunnid”) ning vastav ühik (nt „tund”). X-teljel on ajaskaala kuu algusest kuu lõpuni. Graafiku allosas asub legend, mis näitab iga töötaja nime ja temale vastavat värvi joonisel. Graafiku jooned on sirged, andmepunktidega tähistatud tööpäevadel. Päevadel, mil andmeid ei ole (nt nädalavahetused), jätkub joon sujuvalt sirgena.

Ligipääsuõigused

Sarnaselt teistele rakenduse vaadetele sõltub nähtav info kasutaja rollist:

- **Tavatöötajad** näevad ainult oma andmeid ning ei saa teisi töötajaid graafikule lisada.
- **Otsesed juhid** näevad oma alluvate andmeid ja saavad neid graafikule lisada.
- **Administraatoritel** piiranguid ei ole – nad saavad vaadata ja võrrelda kõigi töötajate statistikat.

Kasutajarollide info saadakse JWT-st.



Joonis 11. Rakenduse statistika vaade, mis kuvab töötajate päevaseid töötunde.

7 Analüüs ja järeldused

Pärast valideerimist viiakse läbi analüüs, mille eesmärk on hinnata, kas loodud rakendus vastab eelnevalt seatud nõuetele. Lisaks tuuakse välja võimalikud tulevased arendussuunad ning põhjendatakse, miks neid ei olnud võimalik käesoleva töö raames ellu viia.

7.1 Nõuete täitmise analüüs

Algses probleemi analüüsimise protsessis sai püstitatud loodavale rakendusele nõuded. Rakenduse valmides tuleb teha analüüs, kas antud nõuded said täidetud.

7.1.1 Funktsionaalsete nõuete täitmise analüüs

Analüüsiprotsessis sai rakendusele seatud kümme funktsionaalset nõuet, mida loodud rakendus peab täitma. Tabelis 2 on võimalik näha kõiki seatud nõudeid nende staatuse ning lahendusviisiga.

Tabel 2. Funktsionaalsete nõuete täitmise analüüs.

Nr	Nõude kirjeldus	Staatus	Lahendusviis
1	Kasutaja saab sisse logida kasutajanime ja parooliga	Täidetud	Login-vaade koos autentimise loogikaga on realiseeritud. Kasutatakse JWT, mis genereeritakse edukal sisselogimisel.
2	Tagarakendus peab looma JWT, mis sisaldab kasutaja andmeid ja rolli	Täidetud	JWT sisaldab kasutajat identifitseerivaid andmeid ning kasutaja roll määratakse serveripoolselt.
3	Kasutaja saab vaadelda oma kuu graafikut	Täidetud	Kuvatakse vaikimisi aktiivne kuu ja aasta.
4	Kasutaja saab sorteerida oma graafikut kuu ja aasta põhjal	Täidetud	Graafikut on võimalik sorteerida eraldi kuu ja aasta alusel.
5	Kasutaja saab vaadelda konkreetse päeva detaile	Täidetud	Klikkides kuupäeval avaneb detailne info tööaja, pauside ja muu seotud teabe kohta.

Jätkub...

Tabel 2 – *Jätkub järgmisel lehel...*

Nr	Nõude kirjeldus	Staatus	Lahendusviis
6	Süsteem kuvab andmed SQL-andmebaasist	Täidetud	Andmed pärinevad reaalajas SQL-andmebaasist, kasutades REST API päringuid.
7	Süsteem visualiseerib andmeid loetavalt (tabelid, graafikud)	Täidetud	Rakenduses on kasutusel tabeli- ning graafikuvaated, mis annavad selge ülevaate töötundidest.
8	Probleemi korral kuvab süsteem veateate ja/või teavitab admini	Osaliselt täidetud	Kuvatakse kasutajale selgitav veateade; administraatori automaatne teavitamine on kavandatud, kuid veel realiseerimata.
9	Kasutaja saab eksportida oma tööajad PDF või XLSX formaadis	Täidetud	Kasutaja saab eksportida nii graafiku kui detailandmed PDF- või XLSX-vormingus.
10	Süsteem kuvab igakuist statistikat (keskmine, ületunnid jms)	Täidetud	Rakenduses kuvatakse igakuine töötundide kogusumma, keskmine ja ületunnid.

Kuna enamik funktsionaalsetest nõuetest on täidetud ning ainult üks on osaliselt täidetud, võib üldiselt lugeda nõuded täidetuks.

7.1.2 Mittefunktsionaalsete nõuete täitmise analüüs

Analüüsiprotsessis sai rakendusele seatud kaheksa mittefunktsionaalset nõuet, mida loodud rakendus peab täitma. Tabelis 3 on võimalik näha kõiki seatud nõudeid nende staatuse ning lahendusviisiga.

Tabel 3. Mittefunktsionaalsete nõuete täitmise analüüs.

Nr	Nõude kirjeldus	Staatus	Lahendusviis
1	Süsteem peab töötama nii arvutis kui ka telefonis	Täidetud	Rakenduse kasutajaliides on kohanduv erinevatele ekraanisuurustele ja töötab sujuvalt eri tüüpi seadmetes.
2	Rakendus peab olema saadaval 99.9 protsenti ajast	Osaliselt täidetud	Süsteem on töökindel arenduskeskkonnas, kuid puudub eraldi töökindluse testimine ja seiremehhanismid.

Jätkub...

Tabel 3 – *Jätkub järgmisel lehel...*

Nr	Nõude kirjeldus	Staatus	Lahendusviis
3	Kõik kasutajad peavad autentima isikuliselt	Täidetud	Kasutajad autentitakse isikupõhiselt parooli ja JWT'i abil, mis seob sessiooni kindla kasutajaga.
4	Kasutajaliides peab olema eestikeelne	Täidetud	Kogu kasutajaliides on eestikeelne, sh veateated ja menüüd. Mitmekeelsuse tugi on kavandatud, kuid ei kuulu hetkel töömahtu.
5	Süsteem peab taluma samaaegselt kõiki töötajaid	Osaliselt täidetud	Süsteem töötab väikese hulga kasutajatega probleemideta; koormusteste suurema kasutajamahu jaoks pole läbi viidud.
6	Andmevaated peavad olema ligipääsupiiranguga vastavalt rollile	Täidetud	Vaadeldavad andmed sõltuvad kasutaja rollist, mis määratakse JWTi alusel (töötaja, juht, admin).
7	JWT'id peavad olema ajaliselt piiratud ja kontrollitud	Täidetud	Märgistele on määratud aegumistähtaeg ning need kontrollitakse igal päringul.
8	Rakendus peab olema kasutatav ka mitte-IT tehnilistele inimestele	Täidetud	Kasutajaliides on üles ehitatud võimalikult lihtsasti mõistetavaks, testitud tagasiside põhjal mittetehniliste kasutajatega.

Kuna enamik funktsionaalsetest nõuetest on täidetud ning ainult kaks on osaliselt täidetud, võib üldiselt lugeda nõuded täidetuks.

7.2 Valminud lahenduse võrdlus alternatiividega

Nüüd, mil loodud rakendus on valmis, saab hinnata selle sobivust võrreldes töö alguses analüüsitud alternatiividega. Võrreldes Exceli-tabeliga pakub loodud lahendus oluliselt paremat kasutajakogemust, paremat andmete visuaalset esitust ning rollipõhist ligipääsu, mis vastab ettevõtte soovitud töökorraldusele. Samuti on eemaldatud mitmed Exceli-tabeli piirangud nagu failikonfliktide oht, halb skaleeruvus ja visuaalse kasutajaliidese puudumine.

Võrreldes Metabase'i ja Looker Studioga annab loodud lahendus oluliselt suurema kontrolli kasutajaliidese, visualiseerimise ning funktsionaalsuse üle. Kuigi Metabase ja Looker Studio sobivad hästi andmeanalüüsiks ning kiireks raportite koostamiseks, ei paku nad piisavat kohandatavust ega rollihaldust, mis on antud juhtumil oluline.

Seega võib järeldada, et loodud rakendus on hetkevajadusi arvestades sobivaim lahendus. Samas tõi alternatiivide analüüs esile mitmeid ideid edasisteks arendusteks, näiteks kasutajate isiklikud ülevaated või täiustatud graafikute lisamine, mida võiks tulevikus kaaluda funktsionaalsuse laiendamisel.

7.3 Tulevikusuunad ja soovitused

Arenduse käigus tekkis mitmeid ideid, mille realiseerimine võiks muuta veebilehe kasutamise mugavamaks. Järgnevalt on toodud olulisemad võimalikud arengusuunad koos lühikese põhjendusega, miks neid ei olnud võimalik käesoleva bakalaureusetöö raames ellu viia.

7.3.1 Mitmekeelsuse tugi

Praegu on veebirakendus saadaval ainult eesti keeles. Üks võimalik täiendus oleks mitmekeelsuse toetamine, mille rakendamiseks võiks kasutada tõlkehaldustarkvara, nagu näiteks Phrase. Tõlked saaks salvestada keskseks hallatavasse süsteemi ja importida rakendusse vastavalt kasutaja eelistustele. Andmebaasis on juba olemas veerg kasutaja keele salvestamiseks, kuid see sisaldab hetkel ainult väärtust "et". Tulevikus oleks võimalik selle põhjal kuvada tekst vastavas keeles. Samuti võiks lisada rippmenüü, mille abil kasutaja saaks keelt käsitsi muuta.

Miks ei rakendatud lõputöö raames? Mitmekeelsuse lisamine eeldab mahukat ettevalmistust tõlkematerjalidega, samuti muudatusi rakenduse arhitektuuris ja kasutajaliideses. Ettevõte ei pidanud seda rakenduse kasutuselevõtu takistuseks.

7.3.2 Haldusloogika integreerimine rakendusse

Hetkel toimub kogu andmete haldamine Argose süsteemis ning loodud rakendus keskendub andmete kuvamisele. Edaspidi võiks kaaluda osade haldusfunktsioonide, nagu näiteks uue kasutaja loomise, viimist otse rakendusse administraatori tasemel.

Miks ei rakendatud lõputöö raames? Selle realiseerimine oleks nõudnud sügavamat sekkumist olemasolevatesse äriprotsessidesse ning kasutajate rollide ja õiguste haldamise süsteemi loomist, mis oleks märkimisväärselt laiendanud töö mahtu.

7.3.3 Täiendav statistika

Lõputöö kirjutamise hetkeks oli statistikaosa piiratud töötundide keskmise vaatlemisega. Tulevikus võiks lisada täiendavaid mõõdikuid, nagu pauside ja isiklike aegade statistika, mis annaks parema ülevaate töötajate ajakasutusest.

Miks ei rakendatud lõputöö raames? Statistiliste näitajate lisamine eeldab ettevõttesisest analüüsi, et selgitada välja, milliste andmete visualiseerimine toetab kõige paremini juhtide otsustusprotsessi. Loodud vaade ja selle aluseks olev taustaloogika moodustavad tugeva aluse, millele on lihtne uusi näitajaid lisada.

7.3.4 Andmebaasi korrastamine ja optimeerimine

Kasutatav andmebaas on aja jooksul kogunud vananenud tabeleid ning sisaldab dubleerivaid andmeid. Edaspidi oleks otstarbekas andmebaas üle vaadata, eemaldada mittevajalikud struktuurid ja optimeerida olemasolevat, et tagada süsteemi töökindlus ja arusaadavus.

Miks ei rakendatud lõputöö raames? Andmebaasi korrastamine on iseseisev ja aeganõudev protsess, mis nõuab detailset arusaama olemasolevatest sõltuvustest ja andmekasutusest. See ei olnud planeeritud antud lõputöö raamesse.

8 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli arendada Sigma Polymer Groupile uus veebipõhine töötajate ajahaldussüsteem, mis asendaks varasema vigade ja piirangutega süsteemi. Vana lahendus oli tehniliselt vananenud, sisaldas mittevajalikke funktsionaalsusi ning esines tõrkeid uute kasutajate andmete töötlemisel. Seetõttu otsustati ehitada uus süsteem nullist, säilitades vaid olemasoleva andmebaasi.

Töö alguses analüüsiti eelnevat lahendust ja kaardistati võimalikud alternatiivid nagu Metabase ja Looker Studio. Selle tulemusel valiti välja tehnoloogiad, mis vastaksid seatud nõuetele ning võimaldaksid luua skaleeruva ja kasutajasõbraliku rakenduse. Esirakenduse arenduseks valiti Vue.js ning tagarakenduse raamistikuks Spring Boot. Järgnevates etappides töötati välja rakenduse arhitektuur, arendati esi- ja tagarakendus ning viidi läbi testimine ja valideerimine.

Loodud süsteem võimaldab jälgida töötajate tööaega ja liikumisi uste kaudu, pakkudes nii töötajatele kui ka juhtidele reaalajas statistikat ning ülevaateid. Rakendus on kohandatud erinevatele seadmetele ja arvestab kaasaegsete kasutajaliideste disainipõhimõtetega.

Töö tulemusena valmis funktsionaalne ja kasutajasõbralik veebirakendus, mis täidab seatud eesmärgid ja pakub organisatsioonile tööaja haldamiseks usaldusväärset ning tõhusat tööriista.

Kasutatud kirjandus

[1] Richard Patey. „How Employee Monitoring Software Boosts, Not Hinders, Productivity“. *Cloudica* (2025). [Accessed: 18-05-2025]. URL: https://cloudica.com/blog/employee-monitoring-software-boosts-productivity.

[2] M Ajay Srivathsan ja R Rajesh. „WORK TRACK: AN AUTOMATED EMPLOYEE ACTIVITY TRACKING AND MONITORING SYSTEM“. *Journal of Current Research in Engineering and Science* (2024). [Accessed: 18-05-2025]. URL: www.psvpec.in/jcres/2024_1/35.pdf.

[3] Tighten Co. *Laravel Versions*. [Accessed: 21-04-2025]. URL: https://laravelversions.com/en.

[4] The PHP Group. *PHP: Supported Versions*. [Accessed: 21-04-2025]. URL: https://www.php.net/supported-versions.

[5] Vue.js Team. *Vue.js Releases*. [Accessed: 21-04-2025]. URL: https://vuejs.org/about/releases.

[6] Metabase. *Metabase*. [Accessed: 02-05-2025]. URL: https://www.metabase.com/.

[7] Google. *Welcome to Looker Studio!* [Accessed: 02-05-2025]. URL: https://cloud.google.com/looker/docs/studio?hl=en&visit_id=638818145415584794-2977955580&rd=1.

[8] Visure Solutions, Inc. *Funktsionaalsed VS mittefunktsionaalsed nõuded*. [Accessed: 02-05-2025]. URL: https://visuresolutions.com/et/n%C3%B5uete-haldamise-j%C3%A4lgitavuse-juhend/funktsionaalsed-ja-mittefunktsionaalsed-n%C3%B5uded/.

[9] Evan You. *The Progressive JavaScript Framework*. [Accessed: 04-05-2025]. URL: https://vuejs.org/.

[10] Meta Platforms, Inc. *React*. [Accessed: 04-05-2025]. URL: https://react.dev/.

[11] Google. *Angular*. [Accessed: 04-05-2025]. URL: https://angular.dev/.

[12] Vue.js. *Introduction*. [Accessed: 04-05-2025]. URL: https://vuejs.org/guide/introduction.html.

[13] Broadcom. *Spring Boot*. [Accessed: 04-05-2025]. URL: https://spring.io/projects/spring-boot.

[14] OpenJS Foundation. *Run JavaScript Everywhere*. [Accessed: 04-05-2025]. URL: https://nodejs.org/en.

[15] Django Software Foundation. *Meet Django*. [Accessed: 04-05-2025]. URL: https://www.djangoproject.com/.

[16] InformaTecDigital. *SQL Express Server: funktsioonid ja mis on uut*. [Accessed: 04-05-2025]. URL: https://informatecdigital.com/et/sql-express-server/.

[17] Okta, Inc. *JSON Web Tokens*. [Accessed: 04-05-2025]. URL: https://auth0.com/docs/secure/tokens/json-web-tokens.

[18] Mozilla Corporation. *A typical HTTP session*. [Accessed: 04-05-2025]. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Session.

[19] Broadcom. *OAuth 2.0 Resource Server JWT*. [Accessed: 04-05-2025]. URL: https://docs.spring.io/spring-security/reference/servlet/oauth2/resource-server/jwt.html.

[20] JetBrains. *The IDE for Professional Development*. [Accessed: 10-05-2025]. URL: https://www.jetbrains.com/idea/.

[21] Git. *Git*. [Accessed: 10-05-2025]. URL: https://git-scm.com/.

[22] GitLab Inc. *About GitLab – The DevSecOps Platform*. [Accessed: 10-05-2025]. URL: https://about.gitlab.com/.

[23] Bill Thomack. *Time Management for Today's Workplace Demands*. [Accessed: 11-05-2025]. URL: https://journals.sagepub.com/doi/10.1177/216507991206000503.

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

Mina, Annika Suurna

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Töötajate ajahaldussüsteemi arendamine Sigma Polymer Groupile", mille juhendaja on Jan Kõrva and Ian Erik Varatalu
    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.

19.05.2025

[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 – Gitlab'i link

Loodud ajahaldusrakendus on saadaval GitLabis: `https://gitlab.cs.ttu.ee/asuurn/iaib`