Tartu Ülikool

Sotsiaalteaduste valdkond

Narva kolledž

Infotehnoloogiliste süsteemide arendus

Anton Buketov

Veebirakenduse loomine dokumendihalduseks huvikoolides

Lõputöö (8 EAP)

Juhendaja: M.Sc. Andre Sääsk

Narva 2023

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Anton Buketov,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose: Veebirakenduse loomine dokumendihalduseks huvikoolides, mille juhendaja on Andre Sääsk, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.
2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Anton Buketov

11.05.2023

Resümee

Programmeerimiskool Edukoht on seni märkimisväärset palju aega ja ressursse kulutanud oma töötajate töötasude arvestamise peale. Peamine probleem seisneb selles, et infovoog on väga suur ja protsessid pole optimeeritud. Selle diplomiprojekti raames püüan ma leida lahenduse sellele probleemile, luues infosüsteemi töötasude arvestamiseks, mis peaks olulisel määral lihtsustama tööprotsesse ja vähendama palgaarvestusega kaasnevaid ressursikulusid.

Minu loodud rakendus kasutab JavaScript, React, CSS ja HTML keeli kliendi osas ning Node.JS ja MySQL serveri osas, lisaks sellele kasutasin selliseid tööriistasid nagu Typeform ja Excel. Tulemusena valmis täielikult funktsionaalne ja jätkusuutlik ärilahendus, mis suurendab oluliselt ettevõtte tootlikkust ja vähendab töövigade arvu. Uurimistöö käigus analüüsisin programmeerimiskooli Edukoht peamisi probleeme ning töötasin välja tõhusaid meetodeid ja lähenemisi nende lahendamiseks.

Töö esimeses osas analüüsin uurimisprobleemi, kirjeldan probleemi sümptomeid ja tagajärgi ning esitan ülevaate olemasolevatest lahendustest. Teises peatükis kirjeldan enda tehtud lahendust rakenduse kujul, mis sisaldab rakenduse arhitektuuri kirjeldust, kasutatavaid tehnoloogiaid ja lähenemisviise ning koodi näiteid.

Võtmesõnad:

Veebirakendus, dokumendihaldus, huvikool, automatiseerimine

Sisukord

Sissejuhatus

Programmeerimiskool Edukoht (PKE) on oma kiire kasvu tõttu silmitsi suurte ajakuludega ja paljude vigadega töötajatele palkade arvestamisel ja maksmisel. Praegu teeb üks ettevõtte juhtidest kogu palgaarvestamise töö ise selle asemel, et kulutada väärtusliku ajaressurssi ettevõtte arengustrateegiaga tegelemiseks.

See lõputöö on äärmiselt oluline tänu asjaolule, et suvel peatub tavaliselt ettevõte PKE väikese perioodi jooksul ning uue õppeaastaga alustatakse tavalist õpet. Seetõttu on suvine periood suurepärane võimalus väljatöötatud funktsionaalsuse järkjärguliseks rakendamiseks "PKE ettevõtte töökeskkonnas.

Töö eesmärk oli luua infosüsteem, mis võimaldaks Moodle'i ja Google Sheet'i mitmeid infoallikaid ühendada ning koondada mentorite tööd puudutavat teavet ühte kohta, et lihtsustada inimese tööd, kes vastutab palkade väljastamise eest, mis praegu käib SmartAccounts süsteemi abiga.

Lisaks on eesmärk automatiseerida täiendavate töötundide info kogumine, sealhulgas tehtud tööde kirjeldused ja kulunud aeg. Selle abil saaks vähendada rühmajuhtide töökoormust ning suurendada info edastamise kiirust.

Minu arendatav infosüsteem pakub lahendust nimetatud probleemile, võimaldades oluliselt vähendada palgaarvestuse koostamiseks kuluvat aega, vähendada käsitsi täidetavate vormide täitmiseks kuluvat aega ja vähendada vigade arvu tööprotsessis. Lisaks hõlbustab süsteem personalijuhtimist ja vähendab võimalike konfliktide tekkimist töökohal. Tulevikus muutub see süsteem laiemaks platvormiks, mis võimaldab töötada mitte ainult mentorite palgaga, vaid ka õpilaste külastamisega.

Minu projekti ainulaadsus seisneb selles, et ma ühendan kaks infokanalit - Moodle ja Google Sheets - ning loon selle põhjal Exceli faili vastavalt nende kahe süsteemi saadetud andmetele. Lisaks sellele, arendan ma välja efektiivsemaid kommunikatsiooniviise ning infotöötluse meetodeid, mis võimaldavad tööprotsesse optimeerida ning suurendada tööjõu tootlikkust.

1 Meetodid

1.1 Mis on õppe juhtimise süsteem

Õppe juhtimise süsteem (LMS) on tarkvara rakendus või veebipõhine tehnoloogia, mida kasutatakse konkreetse õppeprotsessi planeerimiseks, rakendamiseks ja hindamiseks. Seda kasutatakse e-õppe praktikates ja selle kõige tavalisemal kujul koosneb kahest elemendist: server, mis täidab põhifunktsionaalsust ja kasutajaliides (UI), mida kasutavad õppejõud, üliõpilased ja administraatorid. Kirvan ja Brush (2023b) selgitavad seda täpsemalt.

Turul on suur hulk erinevaid õppehaldussüsteemide analooge, mis võimaldavad jälgida nii õpilaste kui ka mentorite edenemist ja statistikat, määrata hindeid ja määrata kodutöid. Kõige populaarsemad ja edukamad LMS süsteemid on Teach 'n Go, Remind, TalentLMS i ja Google Classroom.

Hea LMS-süsteemi omaduste mõistmiseks viidi läbi analüüs nelja platvormi (Teach 'n Go, Remind, TalentLMS, Google Classroom) kohta, mille tulemusena selgusid järgmised faktid:

- “Remind” ja “TalentLMS“ sobib rohkem kursuste formaadile kui huvikoolide süsteemile,
- Peamised põhimõtted õppehaldussüsteemi (LMS) loomisel,
- Millised omadused peaksid olema LMS süsteemil.

LMS süsteem peab omama järgmisi omadusi:

- Selge kasutajaliides ja intuitiivne disain õppuritele ja õpetajatele,
- Mugav ja kiire tagasiside funktsioon õppuritele ja õpetajatele,
- Võimalus luua ja redigeerida õppematerjale, ülesandeid ja teste,
- Õppurite edusammude ja hindamiste jälgimise funktsioonid, samuti õppe edukuse ja osalemise statistika,
- Võimalus integreerida teiste süsteemide ja tööriistadega,
- Paindlik seadistamine ja võimalus individuaalseks kohandamiseks erinevate õppekursuste ja osalejate vajaduste järgi.

Selleks, et mõista, mis on Teach 'n Go ja Google Classroom ning millised on nende eelised kui LMS-süsteemid, otsustasin uurida mõlema süsteemi tugevusi ja esile tuua nende olulisemad eelised.

Teach 'n Go süsteemi analüüsi käigus avastati järgmised eelised:

- Õpilase üksikasjalik ja mugav esitamine,
- Intuitiivne kalender, kus saab koostada tunniplaani,
- On võimalus jälgida kooli kulutusi ja inventari,
- On olemas võimalus suhelda õpilastega vestluse kaudu,

Google Classroom platvormil on samuti mitmeid eeliseid, sealhulgas:

- Google'i tööriistade täielik integreerimine,
- Madal hind, kuid suur funktsionaalsus,
- Suur kasutajate kogukond.

See uuring andis mulle võimaluse hinnata, kas üleminek ühelt süsteemilt teisele on üldse mõttekas, kui keeruline selline üleminek võib olla ning teha põhjendatud valik minu lahenduse kasuks.

1.2 Valmislahendustest loobumise põhjused

Pärast analüüsi otsustati mitte üle minna uuele LMS-süsteemile ja jääda Moodle'i juurde, kuna:

- Moodle'is on juba salvestatud tohutul hulgal teavet mentorite, õpilaste ja ülesannete kohta ning uuele platvormile üleminek võtaks aega ja rahalisi vahendeid personali koolitamiseks,
- Andmete üleviimisel on oht andmete kadumiseks,
- Mõned paremad analoogid nõuavad tasumist, mis hetkel ei ole ettevõtte jaoks mõistlik rahaline väljaminek,
- PKE jaoks on oluline, et saaks kasutada API-d ja kohandada süsteemi vastavalt oma vajadustele. Kuna ei Teach 'n Go ega ka Google Classroom süsteem võimalda seda, siis need ei vasta meie nõuetele.

Kokkuvõttes otsustati, et oma rakenduse väljatöötamine on kõige tõhusam ja jätkusuutlikum strateegia, mis annab tulevikus kõige paindlikuma ja efektiivse lahenduse ettevõtte probleemidele ja vajadustele.

1.3 Ajakulu analüüs

Alltoodud tabeli (Tabel 1) alusel võib näha, et töötajate arvu kasvuga ettevõtte laienemisel suurenes ka palkade arvestamiseks kulunud aeg. Kui 9 töötaja puhul kulus palkade arvestamisele ja kontrollimisele 3 tundi kuus, siis 50 töötaja puhul läks selleks vaja juba 24 tundi kuus.

Tabel 1

Töötajate arv	9	17	37	50
Kulutatud aja kogus (tunnid)	3	5	12	24

Samuti ilmnes uuringu tulemustest, et töötajate vaheline töö kooskõlastamine oli väljakutseks ja erinevate linnade esinduste juhid hilinesid sageli teabe edastamisega, mis põhjustas täiendavaid viivitusi arvete esitamisel. Need probleemid võivad tõsiselt mõjutada organisatsiooni töö efektiivsust, kuna koordineerimine ja õigeaegne informatsiooni vahetamine on ülioluline ülesannete õigeaegseks täitmiseks. Seetõttu on oluline leida lahendused nende väljakutsete ületamiseks ning parandada organisatsiooni tööprotsesse.

1.4 Valitud metoodika

Ettevõtte vajaduste analüüsi tulemusena valiti välja metoodika ja tehnoloogiad, mis kõige paremini vastasid ettevõtte nõuetele ja rakenduse ülesannetele ning võimaldasid tagada rakenduse edasist arengut ja kohanemist ettevõtte nõuetega. Kõige sobivamaks rakenduse formaadiks oli veebirakendus, mis tähendas HTML5, CSS ja JavaScript'i kasutamist rakenduse kliendi osas. Lisaks neile valiti veel React.js, mis on praegu üks parimatest JavaScript'i teekidest veebirakenduste loomisel. Serveriosa jaoks valiti Node.js, mis on samuti üks tänapäeva enamkasutatavatest teekidest, mis sobib hästi just sellise suurusega projektide jaoks. Andmebaasiks valiti MySQL, kuna see on tasuta ning ühilduv Moodle'i andmebaasidega.

Järgmised tegurid olid rakenduse tüübi valikul määravad:

- Võimalus anda kasutajale programmi viimase versiooni ilma uuendust seadmesse laadimata.
- Puudub vajadus laadida rakendus kasutaja seadmesse.
- PKE meeskonna ulatuslik kogemus veebitehnoloogiates, mis võimaldab projekti edaspidi

kergemini arendada ja toetada.

Valitud metoodika ja tehnoloogiad võimaldavad efektiivset automatiseerimist ja vähendavad vigade arvu, mis omakorda parandab ettevõtte toimimist ning aitab vähendada kulutatud aega. Tööriistade valikul arvestati, et need peavad olema kergesti skaleeritavad ja paindlikud, et tagada rakenduse edasise arengu ja kohanemise ettevõtte nõuetega.

2 Rakendus

2.1 Nõudmised rakendusele

Rakenduse loomisel lähtusin funktsionaalsetest ja mittefunktsionaalsetest nõuetest:

Funktsionaalsed nõuded olid järgmised:

- Töötaja info hankimine Moodle'i andmebaasist.
- Töötaja info kopeerimine Moodle'i andmebaasist ettevõtte andmebaasi.
- Mõelda välja viis tundide arv info salvestamiseks.
- Erinevate kategooriate lisatöötundide arvestamise automatiseerimine.
- Mentorite info väljastamine.
- Õpetatud tundide arvu esitamine.
- Exceli faili genereerimine valemite ja arvutustega.
- Kulude kohta info saamine kindlal kuul.
- Google Sheets'ist info saamine.
- Võimalus vaadata lisatöö kategooriaid, mille kallal iga mentor on töötanud.

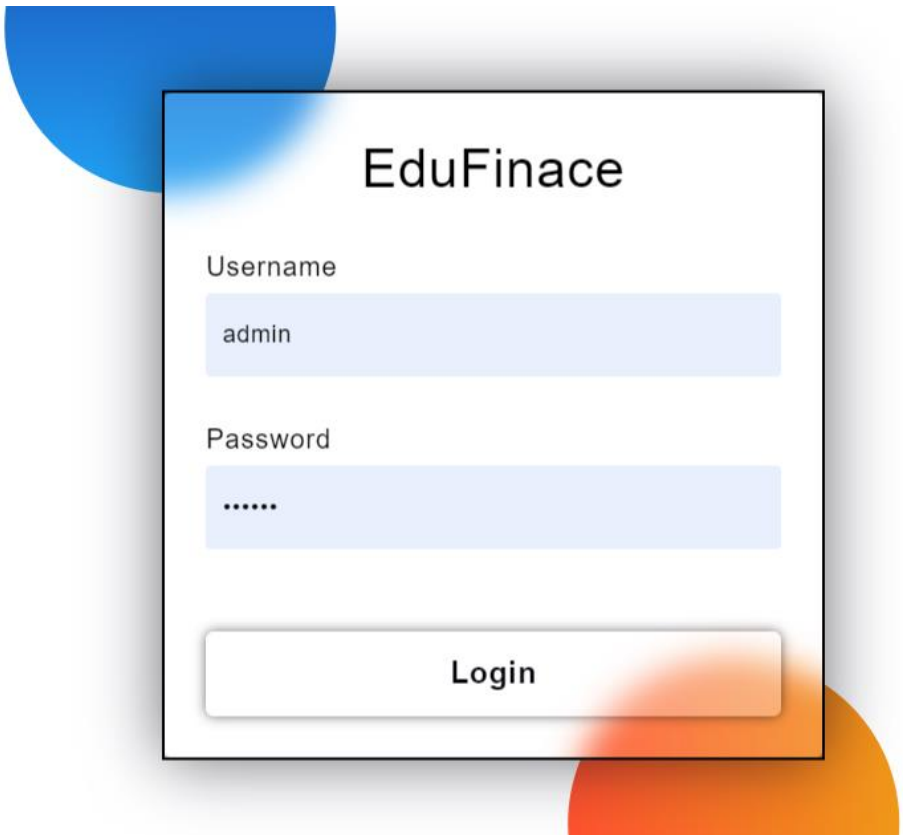
Mittefunktsionaalsed nõuded olid järgmised:

- Kasutusmugavus: Süsteemi lihtne kasutamine ja arusaadavus tavakasutajale.
- Andmete turvalisus: Andmete turvaline ja konfidentsiaalne säilitamine ning kaitse volitamata juurdepääsu eest.
- Skaleeritavus: Süsteemi võimekus laieneda ja kohaneda suurenevate kasutajate või andmemahtudega.

2.2 Rakenduse kasutajaliidese välimus

Infosüsteemi arendamisel seati mittefunktsionaalsed nõuded sõbraliku kasutajaliidese loomisele, mistõttu arendusprotsess algas disainist. Eesmärgi arvesse võtmisel töötati välja rakenduse välimus.

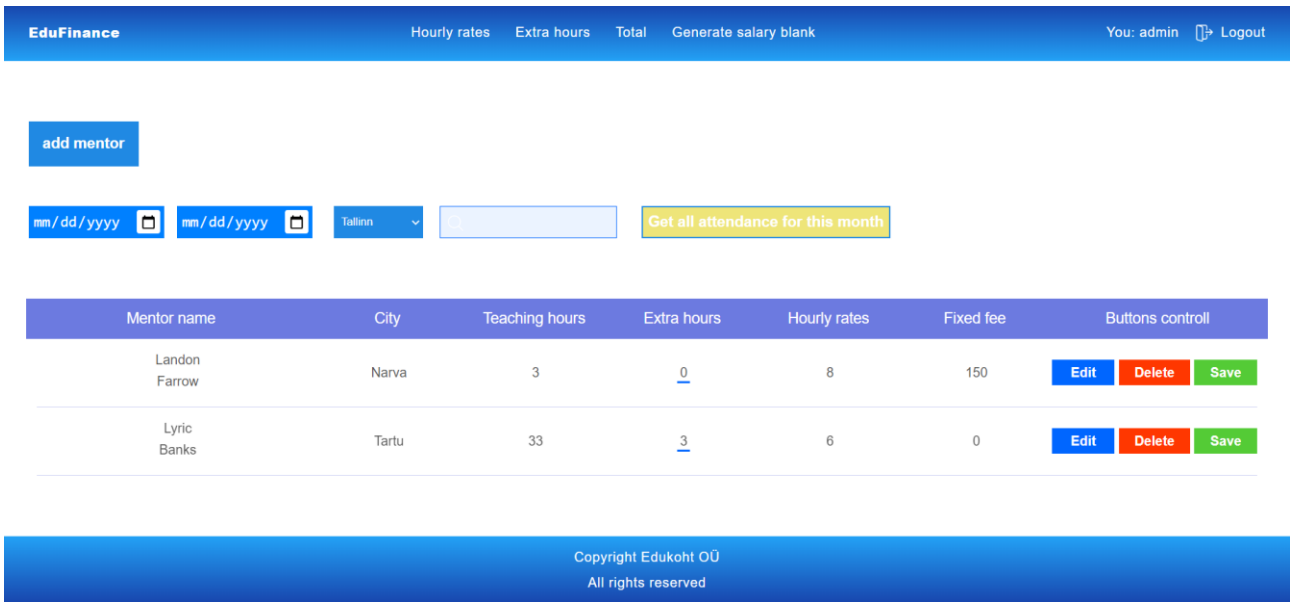
Rakenduse funktsionaalsusele juurdepääsuks on vaja autentimist (Joonis 1), kasutades korrektseid kontosid – sisselogimist ja parooli. Iga kasutaja paroolid salvestatakse andmebaasi krüpteeritult.



Joonis 1

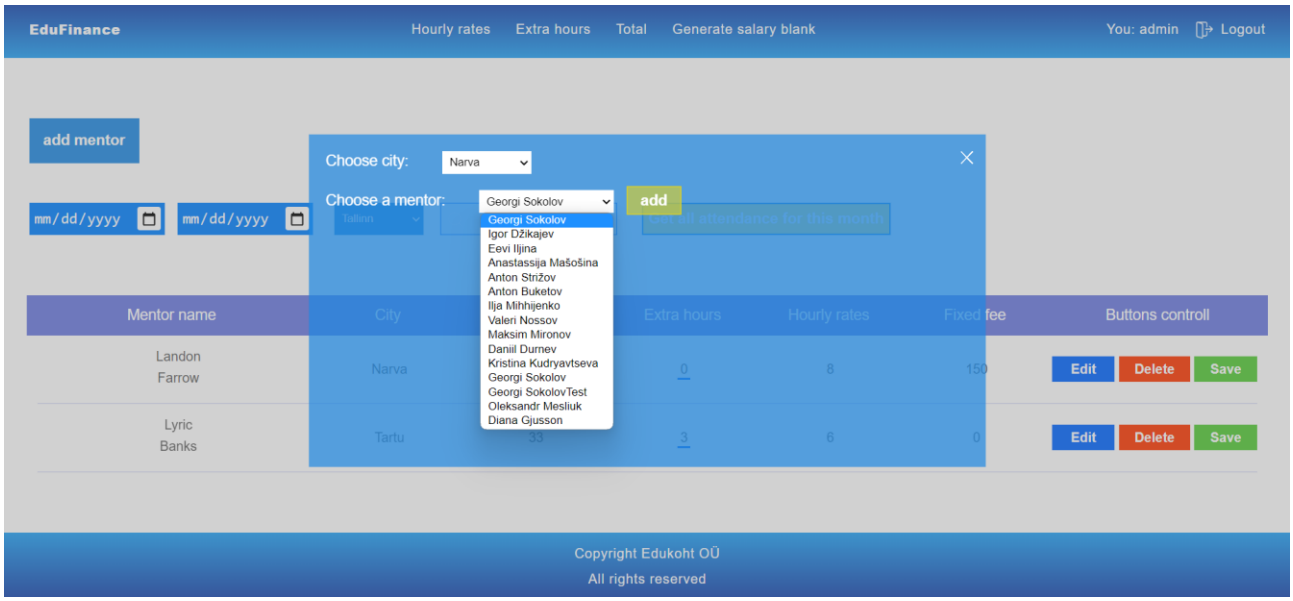
Pärast edukat autentimist suunatakse kasutaja rakenduse avalehele, mis sisaldab järgmisi võimalusi:

- Olemasolevate mentorite filtreerimine ja otsimine.
- Mentorite seisundi jälgimine.
- Mentori info muutmine.
- Mentori info kustutamine.
- Mentori info salvestamine.
- Mentori lisamine Moodle andmebaasist.
- Süsteemist välja logimine.



Joonis 2

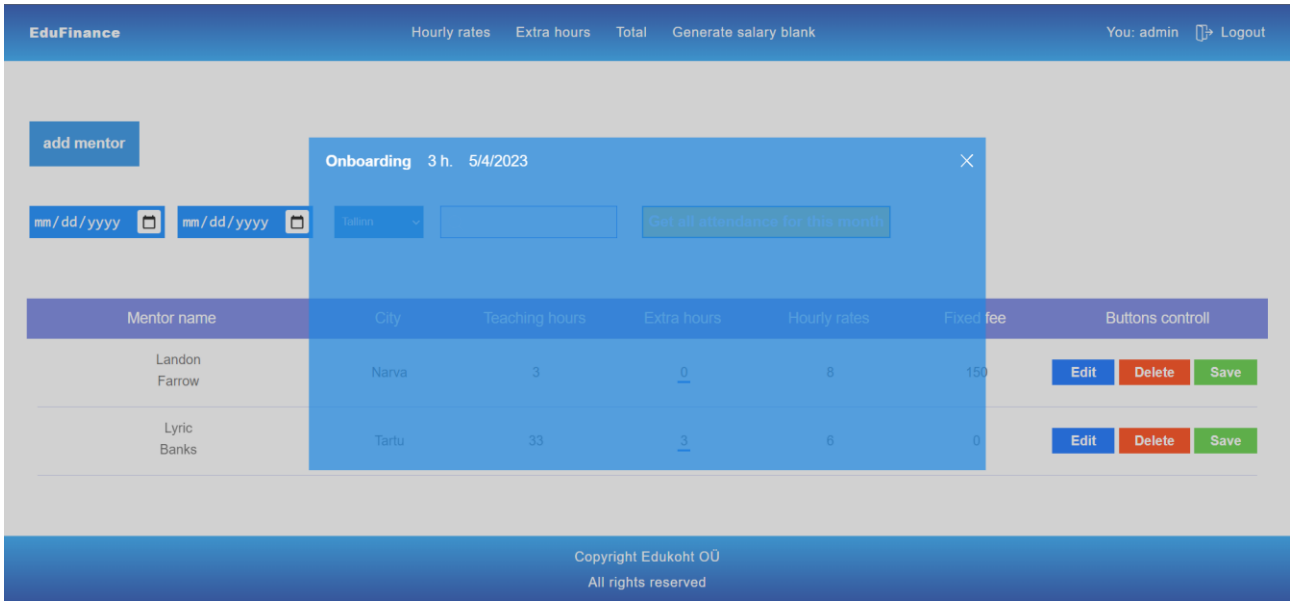
Kui vajutada nupule “add mentor” (Joonis 3), avaneb aken, mis võimaldab valida Moodle'i andmebaasist mentori linna ja nime ning lisada see Edufinance’i andmebaasi.



Joonis 3

Uue mentori lisamise võimaldamine ilma mentorite nimekirja uuendamise vajaduseta võimaldab oluliselt lihtsustada Edufinance'i andmebaasi uuendamist uute mentorite tulekul linna. Seega on uute mentorite lisamine võimalik kiiresti ja ilma kogu andmebaasi uuendamise kuludeta.

Ühekordse klõpsuga konkreetse mentori lahtris "Extra hours" on võimalik vaadata tundide koguarvu (Joonis 4), mis on töötatud üle määratud töögraafiku ja millisesse kategooriasse nende töö tüüp kuulub. Kõik lisatunnid laaditakse automaatselt Google Sheets tabelist, kuhu andmed jõuavad pärast Google'i vormi täitmist mentorite poolt.



Joonis 4

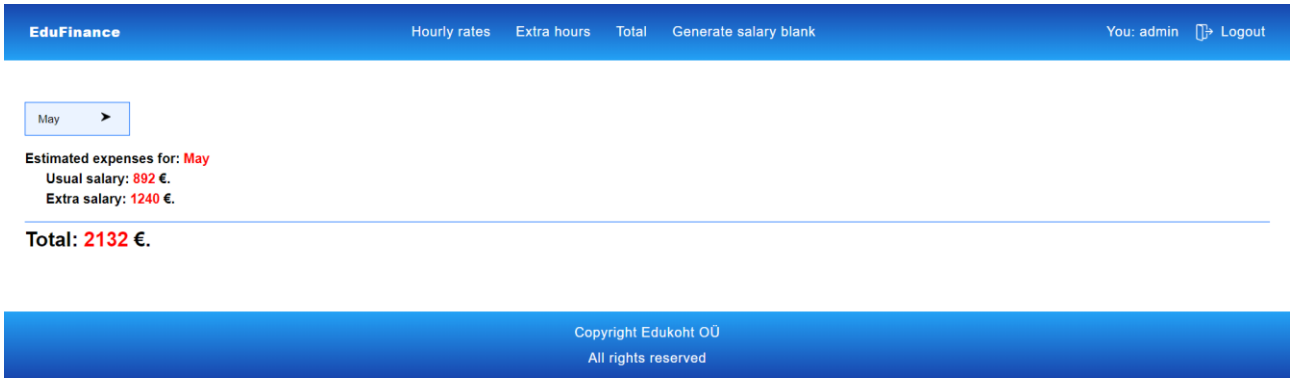
Extra Hours'i lehe eesmärk on hallata mentorite tasustamist erinevates kategooriates. Sellel lehel (Joonis 5) on mitmeid funktsioone, mis võimaldavad neid kategooriaid muuta ja hallata. Nende hulgast võib välja tuua võimaluse:

- lisakategooriate vaatamine;
- kategooriate eemaldamine;
- tunnipanuste ja kategooria nime muutused.

Siiski tuleb märkida, et see funktsioon ei näe ette võimalust uute kategooriate lisamiseks, sest pärast analüüsi on leitud, et praegune kategooriakomplekt on piisav ja ei vaja laiendamist.

Joonis 5

Leht Total (Joonis 6) võimaldab hinnata töötajate palgakulude ligikaudset summat valitud kuu eest, mis võib olla kasulik eelarve planeerimiseks ja kuu ligikaudse maksumuse tutvustamiseks.



Joonis 6

Generate salary blank (Joonis 7) – menüü, mis võimaldab automaatselt luua konkreetse mentori palgatabeli ja kuvada teavet täiendavate töötundide kohta. See tabel genereeritakse Exceli formaadis, mis lihtsustab andmete töötlemist ja analüüsi protsessi.

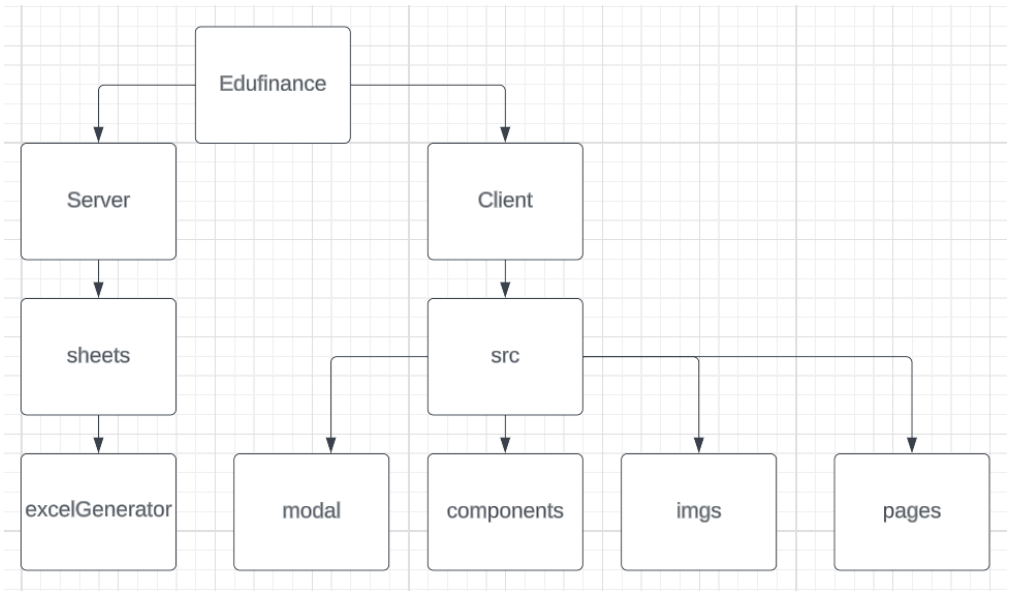
Sellise tabeli loomine on oluline ettevõtte rahavoogude juhtimisel ja võimaldab kiiresti ja täpselt määrata mentori palga suurust ning teha arvutusi täiendavate töötundide põhjal.

Joonis 7

2.3 Projekti struktuur

Kuna rakendus kasutab andmebaasi, siis peab rakenduse jagama kaheks osaks: kliendi- ja serveriosaks. Kliendiosa on see, mida kasutaja näeb ja millega suhtleb, s.t. graafiline kasutajaliides, nupud ja sisestusväljad. Kliendiosa saadab päringuid serveriosale, mille ülesandeks on päringuid töödelda ja vastata nendele. Serveriosa vastutab andmevahetuse eest kliendiosa ja andmebaasi vahel.

Rakenduse arendamise tulemusena saadi järgmine rakenduse kaustastruktuur (Joonis 8), milles kausta nimi vastab kausta otstarbele:



Joonis 8

2.4 Andmebaas

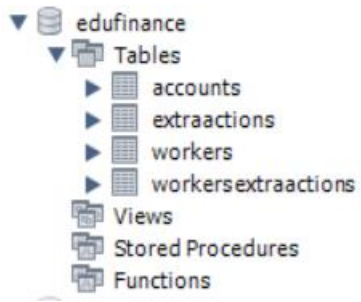
Andmebaasi tööks kasutati MySQL Workbenchi, kuna see on võimas tööriist, millel on mugav graafiline kasutajaliides, mis võimaldab hõlpsalt hallata andmebaase, luua ja muuta tabeleid ja teha päringuid.

Edufinance platvormi arendamise raames otsustati kasutada MySQL andmebaasi. See valik oli tingitud mitmest tegurist, mida süsteemi projekteerimisel kaaluti.

Esimene neist oli see, et Moodle'i süsteem, mida me kasutame Edufinance'i platvormi raames, töötab ka MySQL-i andmebaasis. Seega võimaldab sama andmebaasi kasutamine säilitada süsteemi ühtsuse ja ühilduvuse.

Lisaks plaanime oma platvormi edasiarendamiseks kasutada veebimajutuse teenust, mis sisaldas MySQL-i andmebaasi kasutamise võimalust. See muudab süsteemi juurutamise ja haldamise lihtsamaks. Edufinance'i andmebaas on üles ehitatud vastavalt meie platvormi nõuetele ja vajadustele. See sisaldab mitmeid tabeleid, mis salvestavad teavet kasutajate, mentorite, kategooriate ja tunnitasude kohta. Igal tabelil on oma kordumatu identifikaator ja see on võtmete kaudu seotud teiste tabelitega, mis tagab andmete terviklikkuse ja järjepidevuse.

Üldiselt oli MySQL-i andmebaasi kasutamine teadlik otsus, mis põhines Edufinance'i platvormi nõuete ja vajaduste analüüsil ning selle koostoimel teiste süsteemide ja teenustega.



Joonis 9

Edufinance'i andmebaas sisaldab mitmeid tabeleid, mida kasutatakse erinevate süsteemi toimimisega seotud andmete salvestamiseks. Üks neist tabelitest – “accounts” – salvestab süsteemis autoriseerimiseks vajalikud kasutajanimed ja paroolid räsi.

Teine tabel – „extraactions“ – sisaldab teavet lisatundide kategooriate ja nende määrade kohta. Neid andmeid kasutatakse mentorite palkade arvutamiseks ja finantsaruannete koostamiseks.

Tabelis “workers“ hoitakse andmeid kõigi ettevõtte töötajate kohta, kes on meie platvormi mentorid. Need andmed sisaldavad teavet töötajate täisnime, kontaktandmete ja muude töötajate tööks vajalike atribuutide kohta.

Lõpuks sisaldab tabel “workerextractions“ teavet, mis on saadud Exceli failist, mis salvestab lisatöötundide arvu ja neile vastava kategooria. Seda tabelit kasutatakse mentorite töötasude arvutamiseks lisatöötundide ja nende vastavate määrade alusel.

2.4.1 Ühenduse loomine andmebaasidega:

Andmebaasidega töötamiseks valisin mysql teegi, kuna see Node.js'i teek pakub lihtsat ja mugavat liidest serveri poolel MySQL-i andmebaasiga suhtlemiseks. Sellel on võimsad funktsioonid ja API-d, mis muudavad andmebaasi päringute tegemise ja andmete haldamise lihtsaks.

Arvasin ka, et uut teeki pole vaja kasutada, sest jällegi pean kulutama aega uue dokumentatsiooni uurimiseks.

Rakenduse väljatöötamisel kasutatakse kahte andmebaasi: Moodle’i õppesüsteemi andmebaasi ja Edufinance’i süsteemi andmebaasi. Nende toimimise tagamiseks rakendatakse rakenduses standardseid andmebaaside meetodeid, nagu createConnection, mis võimaldab andmebaaside vahel ühendust luua.

Need meetodid tagavad juurdepääsu andmebaasides hoitavatele andmetele ja võimaldavad nende andmete päringuid teha, neid muuta ja vajalikku teavet välja võtta. Andmekogudega liitumine toimub autoriseerimisrakendusele ja andmebaasidele juurdepääsu võimaldavate kontoandmete abil.

2.4.2 Päringud Moodle süsteemile:

Moodle'i ja Edufinance'i andmebaaside päringute hõlpsamaks kasutamiseks on kõik päringud salvestatud samasse index.js faili.

SQL päringud on oluline osa andmebaasi tööst ja kuna funktsionaalsed nõuded eeldasid, et mentoreid tuleb otsida Moodle'i andmebaasist, siis loodi selleks spetsiaalne päring, mis võimaldas saada vajalikku teavet mentorite kohta. See päring (Joonis 10) aitab kindlaks teha, mil-

listes tundides konkreetne mentor osales, mis omakorda annab meile ülevaate tema töökoormusest ja panusest õppetöösse.

```
const sqlQuery =
"SELECT mdl_user.firstname, mdl_user.lastname, mdl_user.city, mdl_user.email, COUNT(*) * 2.5 AS lesson_number\
FROM mdl_attendance_log \
INNER JOIN mdl_attendance_statuses ON mdl_attendance_statuses.id = mdl_attendance_log.statusid \
INNER JOIN mdl_user ON mdl_user.id = mdl_attendance_log.studentid\
INNER JOIN mdl_attendance_sessions ON mdl_attendance_sessions.id = mdl_attendance_log.sessionid \
WHERE mdl_attendance_statuses.acronym = 'P'\
AND mdl_attendance_sessions.sessdate < ? \
AND mdl_attendance_sessions.sessdate > ?\
AND mdl_user.id IN (\
    SELECT userid\
    FROM mdl_role_assignments\
    WHERE roleid = 3\
)\
GROUP BY mdl_user.firstname, mdl_user.lastname, mdl_user.city, mdl_user.email;"
```

Joonis 10

Päring teeb järgmist:

- Valib kõigi mentorite rolliga õpilaste eesnimed, perekonnanimed, linnad ja meiliaadressid (rolli ID 3);
- Lisab tabeli mdl_attendance_log, mis sisaldab infot mentorite külastuste kohta;
- Lisab tabeli mdl_attendance_statuses, mis sisaldab kohaloleku olekute infot (nt "osalemine", "eemal", "hiline" jne);
- Lisab tabeli mdl_attendance_sessions, mis sisaldab teavet klasside ja nende kuupäevade kohta;
- Filtreerib kirjeid, et valida ainult need, mille kohaloleku olek on 'P' ("osalenud" jaoks);
- Filtreerib kirjeid, et valida ainult need, mis kuuluvad kahe määratud kuupäeva vahel toimunud klassidesse;
- Rühmitab kirjed mentori eesnime, perekonnanime, linna ja e-posti aadressi järgi;
- Funktsiooni COUNT(*) abil loendab iga mentori osaletud tundide arvu;
- Korrutab iga mentori seansside arvu 2,5-ga (mis on iga seansi kestus tundides);
- Tagastab päringutulemused, sealhulgas mentorite eesnimed, perekonnanimed, linnad ja e-posti aadressid, samuti nende osaletud tundide koguarvu.

```
  const sqlQuery =
  "SELECT DISTINCT mdl_user.firstname, mdl_user.lastname, mdl_user.city, COUNT(*) * 2.5 AS lesson_number \
FROM mdl_attendance_log \
INNER JOIN mdl_attendance_statuses ON mdl_attendance_statuses.id = mdl_attendance_log.statusid \
INNER JOIN mdl_user ON mdl_user.id = mdl_attendance_log.studentid \
INNER JOIN mdl_attendance_sessions ON mdl_attendance_sessions.id = mdl_attendance_log.sessionid \
WHERE mdl_user.email = ? AND mdl_attendance_statuses.acronym = 'P'\
AND mdl_attendance_sessions.sessdate < ? \
AND mdl_attendance_sessions.sessdate > ? ";
```

Joonis 11

See päring (Joonis 11) on vajalik ainult ühe konkreetse mentori teabe valimiseks ja selle teabe esitamiseks Edufinance'i andmebaasi salvestamiseks. SQL päring valib kõik unikaalsed (DISTINCT) mentorite eesnimed (firstname), perekonnanimed (lastname) ja linnad (city), kes omavad mentorirolli (rolli ID 3) ja kelle kohaloleku olek ('statusid') on 'P' (osalenud) ning kuuluvad klasside hulka, mis on toimunud kahe määratud kuupäeva vahel (sessdate < ? ja sessdate > ?).

Päring kasutab INNER JOIN-i, et liita andmeid kolmest erinevast tabelist: mdl_attendance_log, mdl_attendance_statuses ja mdl_attendance_sessions, et saada infot mentorite kohal viibimise ja klasside kohta. Samuti liidetakse tabel mdl_user, et saada õpilaste ees- ja perekonnanimed ning linnad.

Lisaks sellele kasutatakse funktsiooni COUNT(*) abil igale mentorile nende osaletud tundide koguarvu arvutamiseks. Seejärel korrutatakse see arv 2,5-ga, et saada kogu tundide arv.

Päring lõpetatakse kahe küsimärgi kohaga, mida kasutatakse SQL'i parameetritena, et sisestada andmed hiljem päringu käivitamise ajal.

Need kaks päringut on väga olulised Moodle'i andmebaasiga töötamisel.

2.4.3 Päringud Edufinance süsteemile

Pärast vajaliku info saamist Moodle'i andmebaasist, on vajalik salvestada need andmed Edufinance'i andmebaasi, sest see on üks rakenduse funktsionaalsetest nõuetest.

```
const readyQuery = "SELECT workers.*, workers.lastName, \
IFNULL(SUM(workersextraactions.extrahours), 0) AS totalExtraHours\
FROM workers\
LEFT JOIN workersextraactions ON workers.firstName = workersextraactions.firstname \
AND workers.lastName = workersextraactions.lastname AND MONTH(workers.startDate) = MONTH(workersextraactions.date)\
GROUP BY workers.workerid";
```

Joonis 12

Päring mis on kujutatud Joonis 12 ühendab info kahest tabelist – "workers" ja "workersextractions". Päring koosneb mitmest osast, millest igaüks täidab kindlat funktsiooni. Esimesel real valime kõik veerud tabelist "workers" ja lisame uue veeru "lastName". Teisel real kasutame funktsiooni IFNULL, et kontrollida tabeli "extrahours" veeru väärtust NULL (tühiväärtus) ja asendada see juhul, kui väärtus on NULL, 0-ga. Kolmandal real kasutame operaatorit LEFT JOIN, mis ühendab tabelid "workers" ja "workersextractions" veergude "firstName", "lastName" ja "date" väärtuste järgi. See võimaldab andmeid mõlemast tabelist välja võtta ja neid omavahel suhestada. Neljandal real kasutame operaatorit GROUP BY, et rühmitada andmed tabeli "workerid" veeru unikaalsete väärtuste järgi.

Selle päringu tulemusel genereeritakse tabel, mis sisaldab kõiki "workers" tabeli kirjeid ja lisaveeru "totalExtraHours", kus näidatakse iga unikaalse "workerid" veerus summaarset tundide arvu tabelis "workersextractions" oleva "extrahours" veeru alusel. Kui mingi töötaja jaoks pole tabelis "workersextractions" vastava kuu kirjet, siis tema jaoks on veerus "totalExtraHours" null.

```
let querySelect = "SELECT COUNT(*) AS count FROM workersextraactions WHERE firstname = ? AND lastname = ? AND MONTH(date) = MONTH(?)";
let query = "INSERT INTO workersextraactions (idextraActions, date, extrahours, firstname, lastname) VALUES (?, ?, ?, ?, ?)";
```

Joonis 13

Mentorite lisatundidega töötamise päringu väljatöötamise raames otsustati see jagada kaheks. Päringu esimene osa kontrollib konkreetse mentori andmete olemasolu andmebaasis ning päringu teine osa lisab puuduva mentori andmed andmebaasi, kui see on olemas.

Siin (Joonis 14) näidatud võtmepäring, mis on oluline mentori andmete hankimisel Edufinance andmebaasist.

```
const readyQuery = "SELECT workers.*, workers.lastName, \
IFNULL(SUM(workersextraactions.extrahours), 0) AS totalExtraHours\
FROM workers\
LEFT JOIN workersextraactions ON workers.firstName = workersextraactions.firstname \
AND workers.lastName = workersextraactions.lastname AND MONTH(workers.startDate) = MONTH(workersextraactions.date)\
GROUP BY workers.workerid";
```

Joonis 14

See päring kasutab vasakut ühendusoperaatorit (LEFT JOIN), et liita tabelid "workers" ja "workersextraactions" kokku. Selleks kasutatakse veerge "firstName", "lastName" ja "startDate", et sobitada kirjeid mõlemas tabelis. Seejärel kasutatakse funktsiooni IFNULL, et asendada "totalExtraHours" veerus puuduvad väärtused 0-ga. Päring rühmitab kirjeid "workerid" veeru järgi, et näidata iga töötaja kogu lisatundide arvu.

Viimane oluline päring, millele tuleks tähelepanu pöörata, on päring, mis kogub ülejäänud andmed Edufinance'i andmebaasi teistest väljadest, et luua tulevikus Exceli fail.

```
"SELECT w.*, e.extraName, e.extraRate \
FROM workersextraactions w \
INNER JOIN extraactions e ON w.idextraActions = e.idextraActions WHERE w.firstname = ? AND w.lastname = ? AND MONTH(w.date) = ?",
```

Joonis 15

See päring valib andmed kahest tabelist "workersextraactions" ja "extraactions", kasutades INNER JOIN operaatorit, et ühendada read, mille väärtused veerus "idextraActions" langevad kokku mõlemas tabelis. Seejärel valib päring ainult teatud töötaja ja kuu andmed, mis on määratud päringu parameetrites, kasutades tulemuste filtreerimiseks operaatoreid WHERE ja MONTH. Valitud veerud sisaldavad kõiki veerge tabelist "workersextractions" ja kahte veergu ("extraName" ja "extraRate") tabelist "extractions".

2.5 Serveri osa

Rakenduse serveriosa on lahutamatu osa mistahes kaasaegsest veebirakendusest. See osa rakendusest vastutab andmebaasiga suhtlemise, kliendipäringute töötlemise ja klientidele vastuste saatmise eest.

Oma projektis valisin Expressi teegi, kuna see on üks populaarsemaid ja laialdasemalt kasutatavaid teeke Node.js veebirakenduste loomiseks. Expressil on palju pistikprogramme ja mooduleid, mis lihtsustavad oluliselt arendusprotsessi.

Lisaks on Expressil väga hea dokumentatsioon ja suur arendajate kogukond, mis teeb info leidmise ja probleemide tõrkeotsingu lihtsaks.

Teine Expressi eelis on selle paindlikkus. See võimaldab luua RESTful API-sid, veebirakendusi, veebisaite ja muid rakendusi. Lisaks toetab Express paljusid mallimootoreid, nagu Pug, EJS ja teised, et lihtsustada kasutajaliidese arendusprotsessi.

Lisaks sellele mul on kogemusi töös Expressi teegiga ja teadsin, et ma ei pea kulutama palju aega uue teegi õppimisele. See võimaldab mul keskenduda rakenduse arendamisele ja lõpuprojekti eesmärkide saavutamisele.

Rakenduse serveriosa vastutab ka Exceli aruannete genereerimise eest. Selleks kasutan ExcelJS teeki, mis võimaldab teil luua ja redigeerida dokumente Exceli formaadis JS-keeles. Üldiselt on minu rakenduse serveriosa funktsioonide ja moodulite kogum, mis tagab andmebaasi-

ga suhtlemise, päringute töötlemise ja aruannete genereerimise. Samuti loodi rakenduse põhjaosas eraldi fail, mis salvestab funktsioone, et seda rakenduse eri osades uuesti kasutada.

Üks keerulisemaid päringuid serveriosas on POST-päringu töötlemine aadressiga "/hourlyRates/mentorFilter". Antud päring sisaldab mentorite filtreerimist edastatud andmete alusel ning täidab veel üht funktsionaalset nõuet, näidates infot mentorite kohta.

```
app.post("/hourlyRates/mentorFilter",(req,res)=>{
    const { city: city, startDate:startDate, endDate:endDate, mentorName:mentorName } = req.body;
    let query = "SELECT * FROM workers WHERE 1=1";
    let queryParams = []
    if (req.body.params['city']) {
        query +=  " AND (city = ? OR city IS NULL)";;
        queryParams.push(req.body.params['city']);
    }

    if (req.body.params['startDate'] && req.body.params['endDate']) {
        query += " AND startDate <= ? AND endDate <= ?";
        queryParams.push(req.body.params['startDate'],req.body.params['endDate']);
    }else if(req.body.params['startDate'] && !req.body.params['endDate']){
        query += " AND startDate <= ?";
        queryParams.push(req.body.params['startDate'])
    }else if(!req.body.params['startDate'] && req.body.params['endDate']){
        query += " AND endDate >= ?";
        queryParams.push(req.body.params['endDate']);
    }

    if (req.body.params['mentorName']) {
        const [firstName, lastName] = req.body.params['mentorName'].split(" ");
        if (lastName) {
            query += " AND firstName REGEXP ? AND lastName REGEXP ?;";
            queryParams.push(`^${firstName}|\\s${firstName}`, `^${lastName}|\\s${lastName}`);
        } else {
            query += " AND firstName REGEXP ?;";
            queryParams.push(`^${firstName}|\\s${firstName}`);
        }
    }
```

Joonis 16

Selles koodis (Joonis 16) võetakse kõigepealt välja päringu parameetrid, nagu linn, alguskuupäev, lõppkuupäev ja mentori nimi, kliendi saadetud päringu kehast. Seejärel luuakse SQL-päring, mis algab "SELECT * FROM workers WHERE 1=1".

Järgnevalt kontrollitakse iga päringu parameetrit ja kui see on määratud, lisatakse see SQL-päringusse ja selle väärtus lisatakse parameetrite massiivi.

Kui on olemas valik "startDate" ja "endDate", siis täiendatakse SQL-päringut reaga "AND startDate <=? AND endDate >=?", kus mõlemad parameetrid peavad olema täidetud ja nende väärtused lisatakse parameetrite massiivi. Kui ainult üks neist on määratud, siis lisatakse SQL-päringusse vastav tingimus.

Kui on olemas valik "mentorName", siis kõigepealt jagatakse nimi eraldi osadeks, et mentori ees- ja perekonnanimes oleks võimalik vasteid otsida. Seejärel lisatakse SQL-päringusse tingimus, kasutades regulaaravaldisi, et otsida mentori ees- ja perekonnanime väljadelt vasteid.

Lõpuks edastatakse SQL-päring ja parameetrite massiiv funktsioonile db.query (), mis täidab andmebaasi päringu ja tagastab.

Teine POST-päringu töötleja (Joonis 17) on /sendSalary, millel on oluline funktsioon: faili loomine xlsx-vormingus, see mängib minu rakenduses olulist rolli.

```
app.post("/sendSalary", (req, res) => {
    const mentor = req.body.mentor;
    const { firstName, lastName, startDate} = mentor;
    const monthNumber = parseInt(functions.getMonthNumber(functions.formatDateYYMMDD(startDate)))+1
    console.log("firstName, lastName, startDate,monthNumber ---- " + firstName, lastName, startDate,monthNumber)
    db.query(
      "SELECT w.*, e.extraName, e.extraRate \
      FROM workersextraactions w \
      INNER JOIN extraactions e ON w.idextraActions = e.idextraActions WHERE w.firstname = ? AND w.lastname = ? AND MONTH(w.date) = ?",
      [firstName, lastName, monthNumber],
      (err, extraData) => {
        console.error("extraData: " + JSON.stringify(extraData));
        if (err) {
          console.error(err);
          return res.send("Error retrieving extra data");
        }
        mentor.extraActions = extraData.map(({extraName, extraRate, extrahours}) => ({extraName, extraRate, extrahours}));
        excelGenerator.generateExcel(mentor)
          .then(() => {
            console.log("Excel file created successfully");
            res.send("Salary file generated successfully");
          })
          .catch((error) => {
            console.error(error);
            res.send("Error generating salary");
          });
      }
    )
  });
```

Joonis 17

Kui serverile saabub /sendSalary aadressipäring, saab töötleja funktsioon päringu kehast mentori andmed, sealhulgas tema eesnimi, perekonnanimi ja alguskuupäev. Seejärel saadakse andmebaasist teave mentori alguskuupäevale vastava kuu lisamaksete (extraData) kohta, kasutades funktsioone teisest moodulist (functions). Kui see teave täiendavate väljamaksete kohta lisatakse mentori andmetega objektile, kutsutakse välja meetod generateExcel. Meetod generateExcel (mentor), mis loob mentori palgateabega Exceli faili. Kui fail on edukalt loodud, saadab server vastuse sõnumiga "Salary file generated successfully". Vastasel juhul saadab server vastuse veateatega.

2.5.1 Faili Excel genereeriminec

Excel-faili genereerimine on osa rakendusest ja on kirjas selle funktsionaalsetes nõuetes ning selleks et genereerida Exceli formaadis faili, kasutati Node.js teek nimega ExcelJS. Selle teegi abil oli võimalik programmeerimiselt luua XLSX formaadis faile. ExcelJS-i kasutamine võimaldab protsessi automatiseerimist ning lihtsustab failide loomise ja töötlemise protsessi. Failide loomisel kasutati eelnevalt loodud mallifaili, kus olid juba koostatud kõik vajalikud valemite ja seadete kohad. Uute failide genereerimisel lisatakse automaatselt kõik vajalikud andmed ning valemite arvutamine toimub vastavalt uutele sisestatud väärtustele. Selline lähenemine võimaldab tunduvalt kiirendada uute failide loomise protsessi ning lihtsustada andmete töötlemist.

2.5.2 Sageli kasutatavad funktsioonid

Nagu eelnevalt mainitud, on projektis fail functions.js, mis on eraldi moodul, mis sisaldab erinevaid funktsioone, mida kasutatakse kogu rakenduses. Funktsioonide paigutamine eraldi faili võimaldas muuta koodi struktureeritumaks ja loogilisemaks. Lisaks vähendas see koodi dubleerimist ja lihtsustas selle haldamist. Näiteks, kui on vaja teha muudatusi teatud funktsioonis, saab seda teha ainult ainsas kohas – failis functions.js.

```
function formatDateYYMMDD(dateString) {
  let year, month, day;
  if (dateString.includes('/')) {
    const dateParts = dateString.split('/');
    if (dateParts.length !== 3) {
      throw new Error('Invalid date format');
    }
    [month, day, year] = dateParts.map(part => parseInt(part));
    if (isNaN(month) || isNaN(day) || isNaN(year)) {
      throw new Error('Invalid date format');
    }
  } else if (dateString.includes('-')) {
    const dateParts = dateString.split('-');
    if (dateParts.length !== 3) {
      throw new Error('Invalid date format');
    }
    [year, month, day] = dateParts.map(part => parseInt(part));
    if (isNaN(month) || isNaN(day) || isNaN(year)) {
      throw new Error('Invalid date format');
    }
  } else {
    throw new Error('Invalid date format');
  }
  return `${year}-${month.toString().padStart(2, '0')}-${day.toString().padStart(2, '0')}`;
}
```

Joonis 18

See kood (Joonis 18) on funktsioon nimega formatDateYYMMDD, mida kasutatakse kuupäeva teisendamiseks kas MM/DD/YYYYY või YYYY-MM-DD formaadis. Funktsioon tagastab kuupäeva YYYY-MM-DD formaadis, mis sobib SQL-päringutes andmete filtreerimiseks kuupäeva järgi.

Funktsioon võtab ühe argumendi nimega dateString, mis on rida, mis sisaldab kuupäeva kas MM/DD/YYYYY või YYY-MM-DD formaadis. Meetod includes kasutatakse selleks, et kontrollida, kas dateString sisaldab sümbolit / või -, et kindlaks teha, millises vormingus kuupäev on esitatud.

Kui kuupäeva vorming on määratud, siis kasutatakse meetodit split, et jagada kuupäev eraldi osadeks ning need määratakse muutujatele year, month ja day vastavalt kindlale vormingule. Massiivi destruktureerimist ja map-meetodit kasutatakse selleks, et muuta osad terviklikeks väärtusteks. Kui mõni väärtus ei ole arv, visatakse välja Invalid date format.

Seejärel vormindatakse kuu ja päeva väärtused toString ja padStart meetodite abil kahe numbri sisaldavate ridadena ning need ühendatakse YYYY-MM-DD formaadis aastaga.

See funktsioon aitab vältida vigu SQL-päringutes ja tagab kuupäevade turvalise ja korrektse vormindamise.

Järgmine getCurrentMonthRange funktsioon (Joonis 19) on mõeldud aktiivse kuu alguseks ja lõpuks sekundite vormingus. See on tingitud asjaolust, et Moodle'i andmebaas salvestab kuupäevade ja kellaaegade väärtused sekundite vormingus. Funktsioon võimaldab selle andmebaasiga töötada, määrates kuu alguse ja lõpu sekundites. Seega annab getCurrentMonthRange Moodle-is mugava viisi kuupäevadega töötamiseks.

```
function getCurrentMonthRange() {
  const now = new Date();
  const startOfMonth = new Date(now.getFullYear(), now.getMonth(), 1);
  const endOfMonth = new Date(now.getFullYear(), now.getMonth() + 1, 0);

  return {
    startOfMonth: Math.floor(startOfMonth.getTime() / 1000),
    endOfMonth: Math.floor(endOfMonth.getTime() / 1000)
  };
}
```

Joonis 19

Funktsioon getCurrentMonthRange kasutab kuupäeva ja kellaaja määramiseks objekti Date. Seejärel luuakse Date objektid käesoleva kuu alguseks ja lõpuks, kasutades getFullYear () ja getMonth () Date objekti meetodeid.

Funktsioon tagastab objekti, millel on kaks omadust: startOfMonth ja endOfMonth, mis esindavad käesoleva kuu algust ja lõppu sekundivormingus. Selleks kasutatakse meetodit getTime (), mis tagastab millisekundite arvu alates 1. jaanuarist 1970, 00:00:00 UTC. See väärtus jagatakse 1000-ga, et saada sekundite arv.

Seega pakub getCurrentMonthRange mugavat viisi, kuidas töötada kuupäevade sekundiformaadiga ja seda saab kasutada Moodle'i andmebaasi või muude süsteemide jaoks, kus kuupäevad ja kellaajad salvestatakse sekundiformaadis.

Viimane huvitav funktsioon selles failis on getMonthNumber (Joonis 20). See võimaldab teil saada jooksva kuu arv.

```
function getMonthNumber(dateString) {
  const date = new Date(dateString);
  const monthIndex = date.getUTCMonth() + 1;
  const monthNumber = monthIndex < 10 ? `0${monthIndex}` : monthIndex;
  return monthNumber;
}
```

Joonis 20

Kõigepealt loob funktsioon andmeobjekti, kasutades edastatud argumenti dateString. Seejärel saab ta jooksva kuu meetodiga getUTCMonth (), mis tagastab jooksva kuu indeksi 0-11. Kahekohalise kuu numbri saamiseks kontrollitakse, kas kuu indeks on väiksem kui 10. Kui jah, siis lisatakse numbrile 0, et saada kahekohaline formaat.

Näiteks kui jooksvaks kuuks on veebruar, tagastab funktsioon rea "02".

See funktsioon võib olla kasulik juhul, kui soovite kasutada kuu numbrivormingut, näiteks andmebaasi päringu kuupäeva kujundamisel.

2.6 Kliendi osa

Rakenduse kliendiosa kirjutamiseks kasutati Axiose teeki. Axios on teek, mida kasutatakse API päringute tegemiseks ja kaugallikast andmete saamiseks. Seda moodulit kasutatakse ra-

kenduse backend-osaga suhtlemiseks ja info kuvamiseks kliendiosas. See kasutab liidese loomiseks komponente ja Axiose teeki API-päringute tegemiseks.

Kui loon kliendipoolsesse rakendusse liidese, siis kasutan komponente, mis saavad andmeid rakenduse backend-osa kaudu. Andmete saamiseks rakenduse backend-osast kasutan Axios teeki API-päringute tegemiseks. Axios võimaldab hõlpsalt teha asünkroonseid päringuid ja hallata neid tagasikutsete abil, nii et saan andmed saata ja vastused vastu võtta ilma lehekülje laadimist blokeerimata. Axios on ka väga kohandatav, mis võimaldab mul konfigureerida selle käitumist vastavalt rakenduse nõuetele. Seetõttu on Axios minu valik teegiks kliendipoolses rakenduses andmete saamiseks.

Rakenduse komponentide infoga manipuleerimiseks kasutasin Edufinance GET-i ja POST-i päringuid. Komponendi “HourlyRates” on järgmised päringud:

- GET-päring "/hourlyRates" kõigi mentorite nimekirja saamiseks andmebaasist;
- GET-päring "/getMentor" ühe mentori kohta andmebaasist;
- POST-päring "/saveMentor" mentori info salvestamiseks andmebaasi;
- POST-päring "/saveMentorWithout" salvestada teavet mentori kohta, arvestamata töötatud tundi. POST-päring "/getSheets" saada andmeid Google Sheets;
- POST-päring "/hourlyRates/filterByCity" mentorite filtreerimiseks linnas;
- POST-päring "/hourlyRates/mentorFilter" mentorite filtreerimiseks teiste parameetrite järgi;
- POST-päring "/saveMentorPMS" mentori salvestamiseks Edufinance andmebaasi;
- POST-päring "/getAllMentors" kõigi mentorite nimekirja saamiseks viimase kuu jooksul.

Need päringud tagavad süsteemi andmete haldamise ning pakuvad erinevaid filtreerimise ja mentorite kohta teabe hankimise viise.

ExtraRates'i komponentide koostises on rakendatud kolm POST-päringut:

- POST-päring /updateaction – eesmärk on uuendada kategooria hetkeseisu.
- POST-päring /extraRates – võimaldab saada teavet kõigi saadaolevate kategooriate ja nende maksumuse kohta.
- POST-päring /deleteaction – kategooriate nimekirjast kustutamiseks.

Komponendis “Salary” on ainult üks POST päring:

- POST-päring /getInfo päring, mis võimaldab saada teavet viimase kuu kohta ja kes mento-

ritest on antud kuul töötanud.

Viimane rakendatud komponent on nimega "Total", millel on järgnevad päringud:

- POST-päring /getInfoFor võimaldab saada spetsiifilist infot konkreetse kuu kohta ning tagastab kogu lisategevuste ja töötajate tavapärase palga maksumuse.
- POST-päring /getMonthInfo võimaldab saada teavet kõigi kuude kohta, mille jaoks on olemas kirjed EduFinance andmebaasis.

Seoses vajadusega kuvada erinevat teavet erinevates tabelites ja erinevate päringutega iga komponendi kohta otsustati välja töötada erinevad komponendid tabeli jaoks. “TableRow”i komponent loodi teabe kuvamiseks “HourlyRates”i täielikus ülevaatetabelis, samas kui “RatesRow”i komponent on mõeldud “ExtraRow”i kategooriatega tabeli jaoks. Lisaks sellele töötati “MntRow” komponent välja Exceli formaadis failide genereerimiseks “Salary” tabelis.

Ja selleks, et vastavalt funktsionaalsetele nõuetele kuvada infot mentorite ja nende töötatud tundide kohta, kasutati funktsiooni .map, mis integreerib loendis iga elemendi üle ja loob nende elementide põhjal komponendi TableRow, mis kuvab iga mentori kohta teavet. Sama põhimõtet kasutatakse ka teistes kahes Salary, ExtraRates tabelis.

```
<tbody>
  {workers?.data?.map(item => (
    <TableRow
      key={item.workerid}
      item={item}
    />
  ))
  }
</tbody>
```

Joonis 21

Minu projekti kliendiosa raames töötati välja Modal'i komponent, mis võimaldab modaalakna abil lehel lisainfot välja tuua. See komponent on üks levinumaid veebiliideste elemente ja seda kasutatakse toote, teenuse või funktsiooni üksikasjalike andmete kuvamiseks, samuti vormide kuvamiseks, teavituste ja muude sisutüüpide täitmiseks.

Modal'i komponendi loomisel võtsin arvesse mitmeid tegureid, sealhulgas kättesaadavust, kasutusmugavust ja disaini. Lisaks on vaja tagada modaalse akna õige käitumine ekraani mõõtmete ja muude sündmuste muutmisel, nagu hiireklõps väljaspool modaalset akent.

```jsx
const Modal = ({active,setModalActive,children}) => {
  return (
    <div className={active ? "modal active" : "modal"} onClick={()=>setModalActive(false)}>
        <div className={active ? "modal-content active" : "modal-content"}>
            <span className='modal-close' onClick={()=>setModalActive(false)}></span>
            {children}
        </div>
    </div>
  )
}
```

Joonis 22

„Modal“ komponent on funktsionaalne komponent, mis võtab sisse kolm parameetrit: active, setModalActive ja children.

active – boolean väärtus, mis määrab, kas modaalne aken peab olema aktiivne või varjatud. Kui active on õige väärtus, kuvatakse modaalne aken ja kui false, siis peidetud.

setModalActive – funktsioon, mis määrab active väärtuse. Kui seda kutsutakse väärtuseks false, siis modaalne aken peidetakse.

children – alamkomponendid, mida näidatakse modaalses aknas.

Modaalne komponent kasutab modaalse akna kuvamise juhtimiseks CSS-stiile. Sellel on kaks tütarelementi: modal ja modal-content. Active-klassid kehtivad neile elementidele modaalse akna kuvamiseks või peitmiseks.

Selleks, et teavitada kasutajat ekslikest või edukatest operatsioonidest, töötati välja “Popupinfo“ komponent. See on kasutajaliidese element, mis väljastab vastava sõnumi ja mida saab kasutada vormide saatmisel, andmete valideerimisel ja muudel tagasisidet vajavatel toimingutel.

```jsx
const Popupinfo = ({ nameclass, text, active, setShowPopup }) => {
  useEffect(() => {
    if (active) {
      setTimeout(() => {
        setShowPopup(false);
      }, 5000);
    }
  }, [active, setShowPopup]);

  return (
    <div className={`popup ${nameclass === 'success' ? 'success' : 'error'} ${active ? 'active' : ''}`}>
      <span>{text}</span>
    </div>
  )
}
```

Joonis 23

Komponent võtab järgmised propsid:

nameclass – rida, mis näitab kirja tüüpi (edu või viga).

tekst – tekst, mida tuleb teates näidata.

active – loogiline lipp, mis määrab, kas teade kuvatakse lehel.

setShowPopup – funktsioon, mis määrab active väärtuse ja mida kutsutakse välja, kui teade suletakse.

Komponent kasutab funktsiooni useEffect, et määrata väärtus active false 5 sekundi pärast, kui see on paigaldatud true. See võimaldab teate automaatselt sulgeda pärast ilmumist. Ta määrab ka nameclass'i ja active'i väärtustel põhinevad stiiliklassid, mis võimaldavad muuta teavituse välimust sõltuvalt sõnumi tüübist ja hetkeseisust

Süsteemi kasutaja autoriseerimiseks kasutatakse komponenti Login.

```
const Login = () => {
    const navigate = useNavigate();
    const [username, setUsername] = useState("")
    const [password, setPassword] = useState("")

    const [loginStatus,setLoginStatus]=  useState(localStorage.getItem(localStorage.getItem("loginStatus")|| false));
        const login = async ()=> {
            try {
                const res = axios.post("http://localhost:8080/login",{
                    username:username,
                    password:password,
                }).then((response)=>{
                    if(response.data.message){
                        setLoginStatus(response.data.message)
                        localStorage.setItem("loginStatus", false);
                        navigate("/");
                    }else{
                        setLoginStatus(true)
                        localStorage.setItem("username", username);
                        localStorage.setItem("loginStatus", true);
                        navigate("/HourlyRates");
                    }
                })
            } catch (error) {
                console.log(error)
            }
        }
```

Joonis 24

Komponent (Joonis 24) sisaldab olekut kasutajanime ja parooli salvestamiseks ning autoriseerimise staatust, mis võib olla õige (kui autoriseerimine õnnestus) või veateade (kui autoriseerimine ebaõnnestus).

Nupule "sisse" klõpsates helistatakse funktsioonile login, mis saadab POST-meetodil päringu serverisse ja edastab päringu kehas kasutajanime ja parooli.

Eduka sisselogimise korral tagastab server vastuse väljaga message, mis sisaldab loogilist väärtust true. Sellisel juhul salvestatakse brauseri kohalikku salvestusruumi kasutajanimi ja autentimisstaatus, mis on võrdne tõega, ning suunatakse “HourlyRates” lehele. Kui autoriseerimine ebaõnnestub, tagastab server vastuse veateatega, mis salvestatakse autoriseerimise staatuses ja suunatakse rakenduse avalehele.

Turvalisuse mittefunktsionaalsete nõuete täitmiseks rakendasin paroolide räsi-funktsiooni. Paroolide räsimiseks, nagu ma juba ütlesin, kasutasin Bcryptjs teeki. Bcryptjs on Node.js-i teek, mis võimaldab turvaliselt räsida kasutajate salvestatud paroole andmebaasis. See on oluline, sest kui paroole ei räsita, võivad ründajad neile lihtsalt ligi pääseda ja kasutada neid kasutajakontodele volitamata juurdepääsu saamiseks.

Ma kasutan teeki Bcryptjs, sest see on üks populaarsemaid ja usaldusväärsemaid Node.js-i teeke paroolide hajutamiseks. Bcryptjs pakub kaitset "brute force" rünnete vastu, kus proovitakse leida õige parooli kombinatsioon katsetades.

See kood (Joonis 25) vastutab parooli uuendamise eest andmebaasis.

```
const updateQuery = "UPDATE accounts SET password = ? WHERE accountID = ?";
let salt = bcrypt.genSaltSync(10);
let hashedPass = bcrypt.hashSync(obj[key2], salt)
db.query(updateQuery, [hashedPass, obj.accountID], (err,result)=>{
    if (err) {
        console.log(err);
        return;
    }
    console.log("Password updated successfully");

})
```

Joonis 25

Koodi esimene rida määrab SQL-päringu, mida kasutatakse parooli uuendamiseks. Seejärel luuakse Bcryptjs-teegi abil juhuslik soolarida, mida kasutatakse parooli räsimiseks. Järgmise koodirea parooli räsitakse genereeritud soola abil, kasutades funktsiooni hashSync. Seejärel toimub SQL-päringu täitmine funktsiooni query abil, mis täidab andmebaasipäringu esitatud parameetrite abil. Kui päring on edukas, saadetakse konsoolile sõnum "Password updated successfully". Kui päringu tegemisel tekib viga, kuvatakse veateade.

2.7 Automatiseerimine Google-i vormi abil.

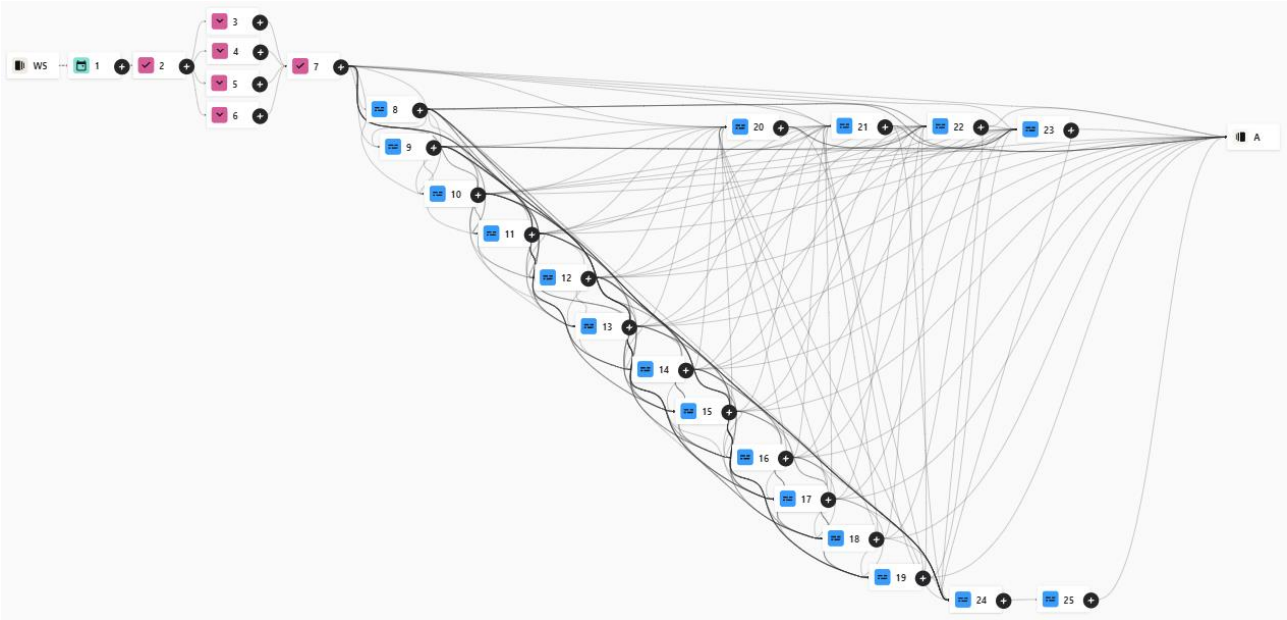
Töötundide arvu salvestamiseks lisakategooriate kaupa on loodud tabel Google Sheetsis (Joonis 26). See tabel on mõeldud tundide andmete hõlpsaks salvestamiseks ja analüüsimiseks erinevates kategooriates. Selle abil täitsin veel ühe funktsionaalse nõude, mis hõlmab lisatundide ja nende kategooriate automatiseerimist.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Kuu	Linn	Nimi	Nimi	Nimi	Nimi	Kategooriad	Automatisation	Bookkeeping	Communication	Community	Curriculum	Design	Engineering	Events	omework sendin
2	5/3/2023			Landon Farrow												
3	5/4/2023			Harper Kingsley												
4	5/2/2023			Kade Blackburn												
5	5/2/2023	Kohtla-Järve			Vars Feelton		Social media									
6	5/4/2023	Kohtla-Järve			Leif Sinclair		Engineering							2		
7	5/4/2023	Tartu	Lyric Banks				Onboarding									
8	5/4/2023	Tartu	Ronan Archer				Onboarding									
9	5/2/2023	Tallinn				Callan Mercer	Bookkeeping		22			3				
10	5/1/2023	Tallinn				Emery Thorn	Communication			1						
11	5/2/2023	Tallinn				Saylor Brighton	Community, Con	4	7	8	9					
12	5/2/2023	Tartu	Tinsley Everett				Social media, Onboarding, Schedule									
13	5/5/2023	Narva		Anton Buketov			Bookkeeping, A	4	4							

Joonis 26

Teabe lisamiseks nimekirja tuleb eelnevalt täita „Google"i vorm, mis on loodud rühmajuhtide ja mentorite töö hõlbustamiseks. Selle vormi täitmine aitab vältida rutiinsete andmeedastusoperatsioonide ajakadu ning tagab täpsema ja tõhusama andmetöötluse. Sellise vormi loomine lihtsustab teabe kogumise ja töötlemise protsessi.

Selle vormi täitmisel tuleb järgida toimingute järjekorda, mille loogika on illustreeritud Joonis 27. Esmalt antakse kasutajale tekstiinfot, misjärel ta peab sisestama kindla kuupäeva. Seejärel valib kasutaja linna ja leiab end selle linna mentorina rippmenüüst. Seejärel valib kasutaja oma lisatoimingu kategooria, sõltuvalt sellest, kui palju kategooriaid olete valinud, peate täitma lisatunnid eraldi iga kategooria jaoks. Pärast vormi täitmist salvestatakse andmed Google Sheetsi.



Joonis 27

Selleks, et luua uus Exceli töövihik-kasutan ExcelJS-i Workbooki konstruktorit. (Joonis 28) Seejärel loetakse mallifail sisse, kasutades readFile() meetodit ja funktsiooni path.resolve(). Pärast seda saame kasutada getWorksheet() meetodit, et juurde pääseda konkreetsele töölehele Exceli failis, kuhu me tahame uusi andmeid lisada. Seejärel saame kasutada töölehel olevaid lahtrid otse ja täita need uute andmetega vastavalt vajadusele. Selline lähenemine võimaldab automaatselt luua Exceli faile ja täita neid vajalike andmetega.

```
const newTable = new ExcelJS.Workbook();
await newTable.xlsx.readFile(path.resolve(__dirname, './template_sheet.xlsx'))
const worksheet = newTable.getWorksheet('Sheet1');
```

Joonis 28

Koodis kasutatakse ka meetodit writeFile() ExcelJS-i objektil xlsx, et salvestada loodud Exceli fail kettale. Faili asukoht ja nimi moodustatakse Node.js teegi funktsiooniga path.resolve (), mis muudab suhtelise tee absoluutseks teekonnaks. Faili nimeks määratakse töötaja ees- ja perekonnanimi ning töö lõppemise kuu arv, mis lisatakse stringina loodava faili laiendiga .xlsx.

Seega luuakse pärast selle funktsiooni täitmist arvuti kettale Exceli fail, millel on töötaja ees- ja perekonnanimi ning töö lõpetamise kuu ja mis sisaldab teavet töötundide kohta kindlaksmääratud aja jooksul.

2.8 Rakenduse kood

Kogu rakenduse kood on kättesaadav aadressil:

https://github.com/KotBegemotEST/diplomaPoject

Kokkuvõte

Edukoht'i äriprotsesside analüüsi lõputöö kinnitab nende protsesside eripära ja vajadust teha põhjalik analüüs ning andmete kogumine ja töötlemine enne vastava lahenduse väljatöötamist. Uuringu käigus leiti nii praeguste äriprotsesside tugevad kui ka nõrgad küljed. Selgus, et oluline osa ajast kulub vajalike dokumentide vormistamisele ja info edastamisele õigele saajale, samuti selgus, et vigade ilmnemisele aitas kaasa kõige käsitsi täitmine ja kõige kontrollimine.

Samuti saavutati tehtud töö tulemusena kõik seatud eesmärgid, nagu:

- Täiendavate töötundide informatsiooni saamise automatiseerimine erinevates kategooriates,
- Kahe infoallika ühendamine rakenduse loomise kaudu.

Soovin väljendada, et loodud lahendus vastab täielikult nii funktsionaalsetele kui ka mittefunktsionaalsetele nõuetele, mis olid sellele rakendusele esitatud. Antud nõuded olid peaaegu peamised kriteeriumid, millele selles töös tähelepanu pöörati. Saadud lahendus võib olla aluseks süsteemi loomiseks PKE ettevõttele pikaajalises perspektiivis. Rakenduse arendamine toimus, võttes arvesse ettevõtte soove ja nõudeid, mis on kriitiliselt olulised selle projekti jaoks, mis annab lahendusele ainulaadsuse.

Tulevikus pööratakse süsteemi edasiarendamisel kindlasti tähelepanu uuringute käigus tehtud järeldustele analoogide kohta, millest peamised on:

- Google'i tööriistade integreerimine on äärmiselt oluline ülesanne, mis mõjutab süsteemi populaarsust ja kasutusmugavust,
- Raha eraldamise täpne jälgimine on äärmiselt oluline vahend, mida saab kasutada PKE finantstehingute täpseks jälgimiseks ja aruandluseks,
- Visuaalne õppetundide ajakava esitlus on oluline kasutajamugavuse ja selguse seisukohalt teiste kasutajate jaoks.

See töö võiks tulevikus olla paremaks muudetud refaktoreerimise teel:

- Mõned detailid andmebaasipäringutes on kõvasti kodeeritud ja nende eraldamine konstantideks võib olukorda parandada.
- Filtreerimise eest vastutavad nupud võib eraldi komponentidesse viia, et parandada koodi selgust ja funktsionaalsust.

- On soovitatav katsetada põhifunktsionaalsust testidega, et tagada süsteemi stabiilne töö.

Kuna seda otsust alles ettevõttele tutvustati, siis on raske kindlaks teha, kui palju on koormus vähenenud. Siiski on juba järgmised tegurid, mis näitavad, et sellel otsusel on positiivne mõju:

- Vähenenud on info andmise viivitus palga eest vastutava isiku ja erinevate asukohtade meeskonnajuhtide vahel;
- Infovoog on vähenenud, info on nüüd struktureeritud;
- Stressi vähendamine juhil. Talle ei ole enam vaja meelde tuletada erinevatest linnadest pärit rühmajuhtidele info andmise vajadust.

Raha- ja ajaresursside säästmiseks on oluline jätkata loodud lahenduse edasiarendamist ja täiustamist vastavalt uuringute käigus saadud järeldustele. Uuringud näitavad, millistele aspektidele tuleks tähelepanu pöörata ning annavad soovitusi edasiste paranduste kohta, mistõttu on see oluline samm süsteemi edasise arendamise suunas

Allikad

1 MYSQL dokumentatsioon. https://dev.mysql.com/doc/refman/8.0/en/join.html Vaadatud 17.03.2023

2 React dokumentatsioon. https://react.dev/learn#sharing-data-between-components Vaadatud 01.04.2023

3 Exceljs. (n.d.). GitHub - exceljs/exceljs: Excel Workbook Manager. GitHub. https://github.com/exceljs/exceljs#create-a-workbook Vaadatud 14.04.2023

4 npm: bcryptjs. (n.d.). Npm. https://www.npmjs.com/package/bcryptjs Vaadatud 01.03.2023

5 Express 4.x - API Reference. (n.d.). https://expressjs.com/en/4x/api.html#express.json Vaadatud 13.04.2023

6 React dokumentatsioon. https://react.dev/learn#using-hooks Vaadatud 03.04.2023

7 React dokumentatsioon. https://legacy.reactjs.org/docs/state-and-lifecycle.html Vaadatud 14.04.2023

8 Brute Force Attack: Definition and Examples. (2023, April 19). www.kaspersky.com. https://www.kaspersky.com/resource-center/definitions/brute-force-attack Vaadatud 01.05.2023

9 Kirvan, P., & Brush, T. (2023b). Understanding Learning Management Systems: An Overview. E-learn Magazine. Retrieved from https://elearnmag.acm.org/featured.cfm?aid=3550477