TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Paul-Bryan Tanilsoo 213113IAIB

SPORDIKLUBI TREENINGUTE HALDAMISE VEEBIRAKENDUSE ARENDUS

Bakalaureusetöö

Juhendaja: Tarvo Treier
Magistrikraad

Tallinn 2024

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: Paul-Bryan Tanilsoo

27.05.2024

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua veebirakenduse prototüüp, mis võimaldab mugavalt ja paindlikult hallata spordiklubi õpilaste, treenerite, treeningute andmeid ja õpilastega seotud arveldamist ning seeläbi vähendab spordiklubi haldamisele kuluvat aega.

Selleks analüüsiti konkreetse padeli- ja tenniseklubi praegust süsteemi ja vajadusi. Uuriti olemasolevaid lahendusi, nende eeliseid ja puudusi. Seati nõuded loodavale rakendusele, pandi paika töö metoodika ning seejärel arendati lahendus, mis vastaks seatud nõuetele.

Töö tulemusel valmis rakenduse prototüüp, millega on võimalik hallata spordiklubi õpilaste ja treenerite nimekirja, määrata nii kuupõhiseid kui ka korrapõhiseid treeningtasusid, koostada korraga kogu arvelduperioodi arveid ning luua treeninggraafikuid. Loodud lahendus säästab märkimisväärselt spordiklubi haldamisele kuluvat aega.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 23 leheküljel, 8 peatükki, 5 joonist, 0 tabelit.

Abstract

Sports Club Training Management Web Application Development

The aim of the thesis is to create a prototype of an application for managing sports club students, coaches, trainings data and the billing of students with the goal of reducing the time spent on managing the sport club.

The thesis begins with an analysis of a certain padel and tennis club's current solution and needs. Existing solutions, their advantages and disadvantages were examined. The requirements for the application to be created were set, the work methodology and the development solution that would meet the set requirements were established.

The finished application prototype can be used to manage the list of students and coaches of the sports club, set training fees per month and per training, create invoices for the entire billing period at once, and manage training schedules. The solution significantly reduces the time spent on managing the sports club.

The thesis is written in Estonian and is 23 pages long, including 8 chapters, 5 figures and 0 tables.

Lühendite ja mõistete sõnastik

API	Rakendusliides (*Application Programming Interface*)
JSON	Andmevahetuse vorming (*JavaScript Object Notation*)
JWT	Digitaalselt allkirjastatav andmevahetuse vorming (*JSON Web Token*)
REST	Tarkvaraarhitektuuri laad (*Representational State Transfer*)
SaaS	Tarkvara teenusena (*Software as a Service*)
SPA	Üheleherakendus (*Single-page Application*)
Sõrestikmudel	Madala täpsusastmega joonis rakenduse kasutajaliidesest

Sisukord

Joonistе loetelu

1. Sissejuhatus

Spordiklubi haldamine on keeruline protsess, mis nõuab planeerimist, arvepidamist ja muudatustele reageerimist. Selleks on vaja ülevaadet õpilastest, treeneritest, väljakutest, treeninggraafikutest, tuludest ja kuludest. Kõik nimetatud tegurid on pidevas muutuses. Näiteks sõltuvad treeningtasud treeningute arvust, väljakute hindadest, treenerite töötasudest jpm faktoritest. Õpilased ei saa aeg-ajalt treeningutes osaleda, uutele õpilastele tuleb leida sobiv treeninggrupp, lisaks regulaarsetele treeningutele on vaja korraldada eratreeninguid – need on vaid mõned näited olukordadest, milledele spordiklubi peab lahenduse leidma.

Töö autor arendas tuttava treeneri tennise- ja padeliklubile veebilehte, mille käigus selgus, et treener pole siiski treeningute haldamiseks sobivat lahendust leidnud. Klubi õpilaste, treenerite ja treeningute andmeid hoitakse tavalises tekstidokumendis ning iga kuu lõpus arvutatakse treening- ja töötasud käsitsi. Protsessile kulub väga palju aega, tihti tekib selle käigus vigu, mis põhjustavad majanduslikku kahju. Treener on katsetanud spetsiaalseid klubihaldustarkvarasid, kuid need pole olnud piisavalt paindlikud. Näiteks puudub võimalus määrata iga õpilase jaoks erinevat arve selgitust või mugavalt rakendada nii kuukui ka korrapõhise tasu arvestust.

Uuriti ka teistes spordiklubides kasutatud lahendusi, millest selgus, et spetsialiseeritud tarkvara kasutamine ei ole eriti populaarne. Jõuti otsuseni luua rakendus, mis pakuks konkurentsi olemasolevale tarkvarale ning lahendaks praegused murekohad klubi haldamisel.

Käesoleva bakalaureusetöö eesmärgiks on luua veebirakenduse prototüüp, mis võimaldab mugavalt ja paindlikult hallata spordiklubi õpilaste, treenerite, treeningute andmeid, õpilastega seotud arveldamist ning seeläbi vähendab spordiklubi haldamisele kuluvat aega.

Töö skoobi piiramiseks keskendutakse reketiklubidele ning luuakse klubisisesed vaated, mis on mõeldud kasutamiseks vaid klubi juhatusele. Prototüübi nõuded määratakse peamiselt spordiklubide Tennisõpe ja PadelTartu (edaspidi klient) vajaduste põhjal, aga suheldakse ka teiste spordiklubide treeneritega.

Töö käigus analüüsitakse kliendi praegust süsteemi, et saada täpne ülevaade selle probleemidest. Tehakse taustauuring olemasolevatele lahendustele ning pannakse paika nõuded

loodavale rakendusele. Seejärel luuakse ülevaade töö tehnoloogisest poolest ning kirjeldatakse tähtsamaid arenduskäike. Tulemuse valideerimiseks katsetab klient loodud rakendust ning kogutakse tagasisidet kliendilt ja teistelt treeneritelt. Lõpetuseks pannakse paika edasine arendusplaan.

Loodav prototüüp on aluseks suuremale projektile, mille käigus soovitakse luua täislahendusena rakendus spordiklubi haldamiseks. Prototüübi põhjal plaanitakse teha laiem turu-uuring, seada järgmised eesmärgid, et pakkuda tulevikus lahendust, mis põhineb SaaS (*Software as a Service*) mudelil. See tähendab, et kogu rakenduse majutuse ja hooldamise eest vastutab teenusepakkuja, spordiklubil tuleb teenuse eest tasuda iga kuu vastavalt kasutusele. Peamiseks sihtgrupiks on reketispordiga tegelevad klubid, ent lahendus on sobilik ka teiste spordialade treeningute haldamiseks.

2. Taust

Käesolevas peatükis kirjeldatakse kliendi praegust spordiklubi haldamise protsessi ja selle kitsaskohti. Seejärel analüüsitakse olemasolevaid lahendusi, nende vastavust kliendi vajadustele ning võimalusi nende kohandamiseks.

2.1 Kliendi praegune süsteem

Kliendiks on tennise- ja padeliklubi, kus õpib ligi 150 õpilast ning toimub üle 170 treeningu kuus. Spordiklubi andmeid hoitakse Google Docs rakenduses loodud dokumendis. Selles on nimekiri õpilaste arveldusandmetest ning treenerite kontaktidest. Iga treeningu kohta tuleb sisestada treeningu aeg, treeneri nimi, õpilaste nimed, väljakutasu, treeneri töötasu ning õpilaste treeningtasu (vt näidist Joonis 1). Spordiklubi treenerid peavad dokumendist jälgima, milliseid treeninguid tuleb neil läbi viia.

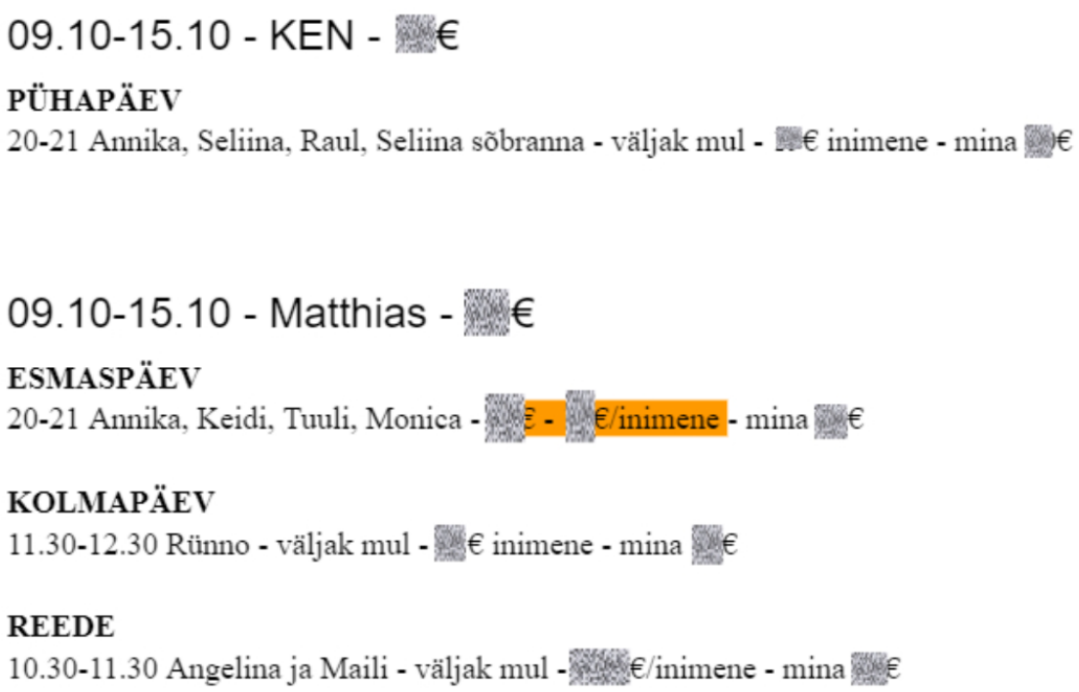
Kuu lõpus arvutatakse käsitsi iga õpilase treeningtasude summa, koostatakse raamatupidamistarkvaras arve, saadetakse see õpilase e-posti aadressile ning märgitakse dokumendis arve koostatuks. Treeningtasusid arvestatakse nii kuu- kui ka korrapõhiselt, mõned õpilased soovivad arvet juriidilisele isikule, arveldatakse üldiselt pärast treeningute toimumist, aga teatud juhtudel ka enne. Arvete koostamiseks kulub terve tööpäev ning tihti tekib selle käigus vigu.

Lisaks tuleb iga treeneri kohta arvutada tehtud töötunnid ning edastada see info raamatupidajale. Kogu protsess on tüütu, aeganõudev ja vigaderohke ning puudub täpne ülevaade klubi asjaajamistest.

2.2 Kliendi vajadused

Praeguse olukorra murekohtade lahendamiseks on vaja süsteemi, mis võimaldab mugavalt hallata klubi liikmete andmeid, treeninggraafikuid, tulusid ja kulusid. Eriti oluline on automatiseerida õpilastele arvete koostamine, sest sellele kulub kõige rohkem aega.

Kliendi jaoks on kõige olulisem, et kasutusele võetud lahendus oleks paindlik. Näiteks peaks süsteem toetama erinevaid treeningtasude struktuure ja arvete detailid peaksid olema lihtsasti muudetavad.


09.10-15.10 - KEN - [redacted]€

PÜHAPÄEV
20-21 Annika, Seliina, Raul, Seliina sõbranna - väljak mul - [redacted]€ inimene - mina [redacted]€

09.10-15.10 - Matthias - [redacted]€

ESMASPÄEV
20-21 Annika, Keidi, Tuuli, Monica - [redacted]€ - [redacted]€/inimene - mina [redacted]€

KOLMAPÄEV
11.30-12.30 Rünno - väljak mul - [redacted]€ inimene - mina [redacted]€

REEDE
10.30-11.30 Angelina ja Maili - väljak mul - [redacted]€/inimene - mina [redacted]€

Joonis 1. Lõik kliendi treeningute märkimise dokumendist.

2.3 Olemasolevad lahendused

Uue rakenduse loomine on keeruline ja kulukas ettevõtmine, mistõttu tuleks eelistada olemasoleva tarkvara kasutamist. Spordiklubi haldamise lihtsustamiseks on loodud väga palju häid rakendusi, milledel on suur kliendibaas, arendusmeeskond ja abivalmis klienditugi.

Koostöös kliendiga katsetati turul olevaid lahendusi, hinnati nende eeliseid ja puudujääke. Otsiti ka võimalusi avaliku API (*Application Programming Interface*) või avatud lähtekoodi põhjal kohandatud lahenduse loomiseks. Hindamisel mängisid suurimat rolli lahenduse paindlikkus, kvaliteet ja maksumus.

Sobivat varianti siiski ei leitud, üldiselt esinesid järgmised probleemid:

- Kuutasudel põhinev arveldussüsteem, korrapõhine hinnastamine puudub või on keeruline
- Arvel märgitud teenuste kirjeldusi ei saa kohandada
- Arveldamine on tugevalt seotud integreeritud makselahendustega, mille kasutamine on kohati kohustuslik
- Puudub eestikeelne kasutajaliides

Parema ülevaate jaoks võttis autor ühendust kahe teise tennisetreeneriga, et teada saada

nende seisukohad antud probleemi suhtes. Mõlemad tõdesid, et ka nemad pole sarnastel põhjustel sobivat lahendust leidnud ning huvi uue rakenduse vastu on olemas.

Trainero, Tennis Locker App, Rekets, Scult, Sportsengine - need on vaid mõned näited katsetatud lahendustest. Järgnevalt on kirjeldatud kolme potentsiaalikamat olemasolevat rakendust, nende positiivseid ja negatiivseid külgi.

2.3.1 Sportlyzer

Sportlyzer on eestlaste poolt loodud klubihaldustarkvara, mis pakub spordiklubi haldamiseks mitmeid võimalusi. Muuhulgas saab Sportlyzeris hallata klubi liikmeid, arveldust, makseid, treeningutes osalemist, saata kirju ja määrata kodutöid[1]. Lahenduse eeliseks on lai funktsionaalsus - rakendus võimaldab luua treeninguid, märkida osalejaid, treenereid ja asukohti. Samuti saab määrata kuutasusid, luua arveid ja ühendada arveldamine Merit Aktiva raamatupidamisprogrammiga.

Probleemiks osutub asjaolu, et arveldamine põhineb peamiselt kuutasudel ning arvete koostamine pole eriti paindlik. Treeningutega pole võimalik siduda korrapõhist treeningtasu. Arvete genereerimisel pole võimalik kohandada iga õpilase jaoks arve sisu ega määrata õpilase arve saajaks juriidilist isikut. Arveid ühekaupa luues saab küll vabalt määrata arve saaja, sisu, kirjeldused jm detailid, kuid see on aeglane ja ebamugav lahendus.

Sportlyzeril puudub avalik API ning sellel on suletud lähtekood, mistõttu pole olemasolevat lahendust võimalik täiendada.

2.3.2 Playtomic

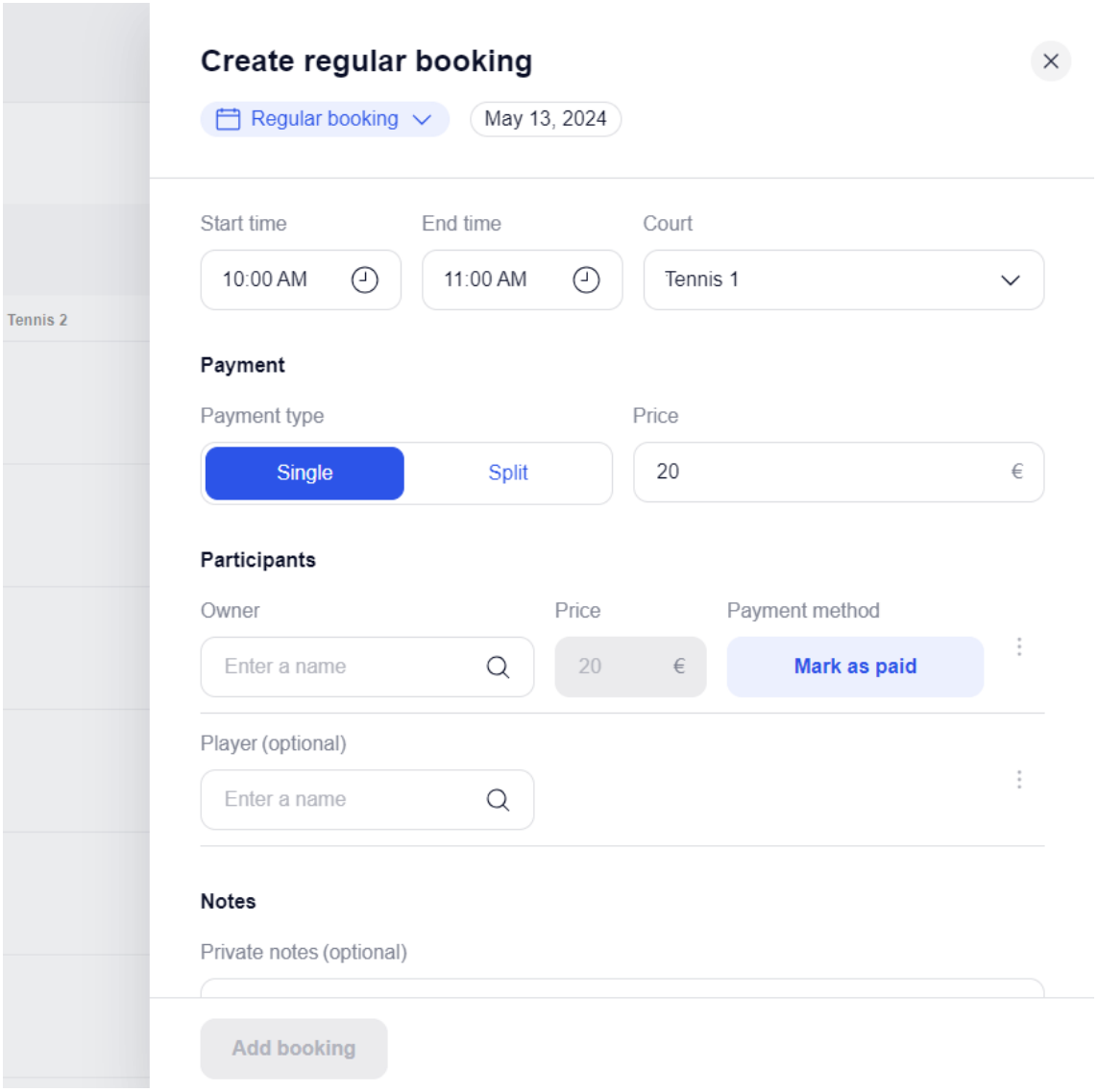
Playtomic on reketiklubi haldamiseks loodud rakendus, mida kasutavad mitmed Eesti padeliväljakute omanikud. Õpilasel on võimalik rakenduses broneerida treeningväljakuid, registreerida treeningutesse, teha makseid, suhelda kogukonnaga. Treener saab lisada nii avalikke kui ka privaatseid treeninguid, seadistada nende hinda ja määrata osalejate arvu [2].

Platvormil on arvukalt funktsioone, näiteks mängijate taseme hindamise süsteem, mobiilirakendus ja sisseehitatud maksevõimalused. Kahjuks puudub mugav viis määrata arvete sisu, luua arveid pangaülekandega maksmiseks, ühendada arveldamine Merit Aktiva raamatupidamisprogrammiga.

Playtomicu treeningu lisamise vaade (vt Joonis 2) on heaks eeskujuks, kuidas lahendada treenerite, õpilaste ja treeningtasude kuvamine. Vaatel on loogiline ülesehitus, arvukalt funktsioone ja selle juures on suudetud säilitada andmete kompaktsus.

Suur valik võimalusi muudab rakenduse kasutamise keeruliseks, isegi lihtsa ülevaate saamiseks tuli süvitsi uurida kasutusjuhendeid ja selgitavaid blogipostitusi. Eestikeelse kasutajaliidese puudumine süvendab probleemi veel enam. Autor on varem kasutanud Playtomicut treeningväljakute broneerimiseks, kogemus on olnud negatiivne just rakenduse keerukuse tõttu.

Kirjeldatud probleemide tõttu pole lahendus sobiv, puudub ka avalik API lahenduse kohandamiseks.



Joonis 2. Playtomicu treeningu lisamise vaade.

2.3.3 BookLux

Booklux on broneerimistarkvara, milles saab hallata treeninggraafikut, õpilaste andmeid, makseid, selles on veel näiteks kliendivaade, mis võimaldab õpilastel broneerida sobivaid treeninguid [3].

Rakendus tundub olevat üpris paindlik, lisamoodulite valik on lai. Prooviks loodud kasutajaga õnnestus sisestada õpilaste andmeid, lisada treeninguid, märkida makseid. Küll aga ei õnnestunud leida täpsemaid juhiseid arvete koostamise kohta ning lisafunktsioonide proovimiseks oleks pidanud tellima mitu tasulist paketti. Ettevõtte kodulehel olev kirjeldus on väheütlev, korraga üritatakse rahuldada väga erinevate klientide, näiteks autoteeninduste ja spaakeskuste, vajadusi. Rakendus tundub olevat poolik ja vigane, näiteks kuvati treeneri andmete juures märge, et tegemist on väljakuga ning kasutajaliides sisaldab arvukalt kirjavigu.

Iga valitud lisamoodul suurendab rakenduse kuutasu, mis läheb kokkuvõttes päris kulukaks. Klient on seisukohal, et rakenduse kuutasu ei vasta pakutud lahenduse kvaliteedile. Booklux pakub ka avaliku API kasutamise võimalust, mille minimaalne kuutasu on 39€, kuid selle põhjal täiendava lahenduse loomine ei tundu eelmainitud probleemide tõttu mõistlik.

3. Analüüs

Vajaduste kaardistamiseks analüüsiti kliendi spordiklubide igapäevatoiminguid, treeningtasude arvestamise, arvete koostamise ja treeningute planeerimise protsesse. Selle käigus tekkis väga palju mõtteid, kuidas klubi toiminguid hõlbustada. Prototüübi jaoks otsustati realiseerida olulisemad, ülejäänud võetakse käsile järgmises arendusfaasis. Nõuete seadmisel arvestati ka teiste eelnevalt mainitud treenerite vajadustega. Samuti lähtuti olemasolevate lahenduste puudustest, aga ka nende positiivsetest omadustest.

Täiendavate mõtete saamiseks ning teistele treeneritele töö idee paremaks selgitamiseks loodi sõrestikmudel rakenduse kalendervaatest, mis on nähtaval Lisas 2.

3.1 Funktsionaalsed nõuded

Rakenduse funktsionaalsed nõuded on järgmised:

Isikud

- Rakendusse peab saama lisada õpilaste andmeid (nimi, kontaktandmed, arveldusandmed, märkused)
- Õpilase arvelduskontaktiks peab saama määrata juriidilist isikut
- Rakendusse peab saama lisada treenerite andmeid (nimi, kontaktandmed, märkused)

Treeningud ja muud üritused

- Rakenduses peab olema kalender, kuhu saab lisada treeninguid
- Kalendris olevale treeningule peab saama määrata taustavärvi
- Treeningusse peab saama lisada õpilasi, märkida nendele rakenduvaid treeningtasusid ja treeningus osalemist või mitteosalemist
- Treeningusse peab saama lisada treenereid ning märkida nende töötasu
- Treeningusse peab saama lisada treeningväljakuid, märkida nende rendikulu ja broneeringu staatust
- Kalendris olevaid treeninguid peab saama filtreerida treeninguga seotud õpilaste, treenerite ja väljakute põhjal
- Kalendrisse peab olema võimalik lisada tavaüritusi, et märkida näiteks treenerite vaba graafikut

Arveldamine

- Õpilastele peab olema võimalik määrata korrapõhiseid treeningtasusid, kuutasusid, turniirimakse jm klubi teenustasusid
- Igale õpilasele peab saama valitud perioodi kohta koostada arve, mis koosneb õpilasele määratud tasudest. Arve kirjeldust peab saama muuta
- Arveid peab saama koostada vabalt valitud perioodi kohta. See tähendab, et õpilastele määratud tasusid on võimalik koguda kuu alguses ettemaksena, kuu lõpus või ka iga treeningu kohta eraldi
- Korraga peab saama genereerida arved mitme õpilase jaoks
- Arvetele peab saama lisada käibemaksu
- Rakendusse peab saama sisestada laekunud makseid
- Koostatud arveid peab olema võimalik kopeerida Merit Aktiva raamatupidamisprogrammi
- Arveid peab saama alla laadida PDF-vormingus
- Rakenduses peab olema alternatiivne (arveväline) arvestussüsteem, kuhu saab valikuliselt lisada õpilastele määratud tasusid, treenerite töötasusid ja makseid. Antud arvestuses olevaid objekte ei tohi saada märkida arvetele ega kopeerida raamatupidamisprogrammi
- Rakenduses peab olema võimalik näha kokkuvõtet valitud perioodil loodud arvetest, toimunud treeningutest, väljakute rendikulust, treenerite töötundidest ja -tasudest

3.2 Mittefunktsionaalsed nõuded

Rakenduse mittefunktsionaalsed nõuded on järgmised:

- Rakendust peab saama kasutada veebilehe kaudu, mis ei nõua lisatarkvara paigaldamist
- Kasutajaliides peab olema modernne ja intuitiivne
- Rakendus ei tohi olla häirivalt aeglane, veebilehe laadimiseks ei tohiks kuluda üle 8 sekundi
- Rakenduse andmetele ei tohi ligi pääseda isikud, kellel puudub selleks voli

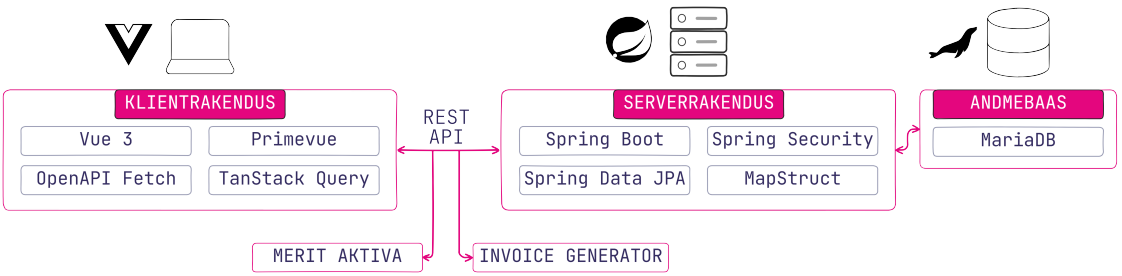
4. Tehnoloogiline ülevaade

Selles peatükis kirjeldatakse loodava rakenduse arhitektuuri, arendusmetoodikat ning peamiste tehnoloogiate valikut.

4.1 Üldine arhitektuur

Töö tulemiks on veebirakendus, mida on võimalik korraga kasutada mitme spordiklubi haldamiseks. Süsteemi põhilisteks osadeks on klientrakendus, serverrakendus ja andmebaasisüsteem. Kasutaja saab veebibrauseris käivitada klientrakenduse, mis pakub kasutajaliidest andmete vaatamiseks ja muutmiseks. Klientrakendus teeb andmete saamiseks ja muutmiseks päringuid serverrakendusele. Serverrakendus saab korraga teenindada mitut kasutajat, selles on rakendatud äriloogika, näiteks kasutaja sisestatud andmete valideerimine, ning see teeb päringuid andmebaasisüsteemile. Andmebaasisüsteemi vastutab andmebaasi kirjutamise ja lugemise eest.

Serverrakendusega suhtlemiseks on defineeritud REST (*Represational State Transfer*) API, andmete edastamiseks kasutatakse peamiselt JSON (*JavaScript Object Notation*) andmevahetusvormingut. Joonisel 3 on esitatud rakenduse arhitektuur koos peamiste tehnoloogiatega.



Joonis 3. Rakenduse arhitektuur.

4.2 Arendusmetoodika

Kuna arendusaeg oli piiratud, aga töö maht suur, lepiti kliendiga kokku, et käesoleva töö raames arendatakse üks terviklik prototüüp. Valmis prototüübi põhjal lepitakse kokku vajalikud muudatused ning edasine arendusplaan. Töö autor ega klient ei pidanud vajalikuks arendusprotsessi keskel poolikut prototüüpi analüüsida, nõuete täpsustamiseks suheldi vajaduspõhiselt.

Arenduskäigu planeerimisel võeti aluseks kliendi nõuded ja arvestati, et täpsemad detailid selguvad töö käigus. Üldiseks tööplaaniks sai projekti seadistamine, põhiliste andmestruktuuride loomine, seejärel nõuetest tulenevate funktsioonide järkjärguline arendamine ning lõpuks loodud lahenduse testserverisse paigaldamine. Töö käigus logiti arendusele kulunud aega ning iga suurema arenduskäigu lõppedes kirjutati tehtud töö kohta kommentaar.

Klient- ja serverrakendust arendati paralleelselt funktsioonide kaupa. Kõigepealt implementeeriti serverrakenduses vajalik loogika, seejärel loodi klientrakenduses vastavad vaated. Sel viisil oli lihtne arvestada mõlema süsteemi osade vajadustega ning loodud funktsiooni sai terviklikult katsetada kohe peale selle implementeerimist.

Töö käigus üritati järgida üldtuntuid kvaliteetse koodi printsiipe. Samas oli oluline, et arendustegevus oleks kiire. Autor üritas leida kesktee koodi kvaliteedi ja koodi kirjutamise kiiruse vahel. Tuleb tõdeda, et töö lõpupoole keskenduti pigem kiire tulemuse saavutamisele ning järgmises arendusfaasis tuleb omajagu koodi ümber teha. Esialgu oli plaan kirjutada loodud lahendusele automaattestid, kuid selleks ei jäänud antud töö raames piisavalt aega. Rakenduse funktsioone testiti käsitsi, automaattestid kirjutatakse arenduse järgmises etapis.

4.3 Peamised tehnoloogiad ja tööriistad

Tehnoloogiate valikul võeti arvesse rakendusele seatud nõudeid, autori kogemust, tehnoloogiate dokumentatsiooni põhjalikkust ja populaarsust. Järgmistes alapeatükkides kirjeldatakse süsteemi olulisemaid tehnoloogiaid ning põhjendatakse tehtud valikuid.

4.3.1 Spring Boot

Serverrakendus arendati Java keeles kasutades Spring raamistikku, täpsemalt selle laiendust nimega Spring Boot, mille abil saab vähese seadistusega luua toimiva lahenduse [4]. Spring Boot võimaldab arendada keerulist ent töökindlat tarkvara, olles paindlik, turvaline, kiire ning omades suurt kommuuni probleemide lahendamiseks [5].

Alternatiiviks on näiteks Node.js JavaScripti käituskeskkond, mis suudab korraga teenindada väga palju samaaegseid päringuid, mille kiirus on oluline [6]. Käesoleva töö raames loodav rakendus aga ei oma kasutusjuhte, kus samaaegsete päringute jõudlus oleks kriitiliselt tähtis. Autor on varasemalt Spring Booti kasutanud nii tööalastes kui ka isiklikes projektides. Spring Boot on maailmas üks populaarsemaid tehnoloogiaid [7]. Eeltoodut arvestades on valitud raamistik sobiv eesmärgi täitmiseks.

Koos Spring Bootiga võeti kasutusele Spring Data JPA[1], mis võimaldab Java koodis defineerida andmebaasi mudelit ja teostada andmebaasioperatsioone. Tulevikus peaks siiski mudeli defineerima mõnel muul viisil, näiteks Liquibase[2] abil, sest Java koodis on lihtne teha vigu. Prototüübi loomiseks on aga kasutusse võetud lahendus ideaalne, sest andmebaasi mudelit saab kiiresti ja vähese vaevaga muuta.

4.3.2 Vue.js

Klientrakendus realiseeriti Vue.js raamistikul JavaScript ja TypeScript keeles. Vue abil on võimalik arendada SPA (*Single-page Application*) põhimõttel töötavat veebirakenduse kasutajaliidest, mis pärib serverilt andmeid vastavalt vajadusele ning kuvab need dünaamiliselt veebilehele, ilma et peaks kogu lehekülge uuesti laadima [8]. Kõnealune raamistik on samuti maailmas üks populaarsemaid [7]. Autor kaalus ka teisi variante, näiteks React või Angular, kuid kiirema arenduse huvides sai otsustavaks teguriks autori varasem kogemus Vuega.

Lisaks võeti kasutusele PrimeVue komponentide teek, milles on arvukalt valmiskomponente kasutajaliidese loomiseks. Disainiprotsessi kiirendamiseks soetati Primevue Apollo mall, mis sisaldas näidisdisaine erinevatele lehtedele. Alternatiivina kaaluti Vuetify kasutamist, millega on autoril väga positiivseid kogemusi, kuid PrimeVue stiil näis modernsem ja värskendavam.

4.3.3 MariaDB

Andmebaas põhineb MariaDB tehnoloogial. See on relatsioonilise andmebaasisüsteemi MySQL täiustatud väljaanne, millel on näiteks suurem operatsioonide täitmise kiirus ning parem laiendatavus kui MySQLil. Mõlemad andmebaasitehnoloogiad on avatud lähtekoodiga [9, 10, 11]. Andmebaasitehnoloogiaks oleks võinud valida ka PostgreSQL, mille eeliseks on näiteks JSON andmetüübi võimekus[12]. Loodava rakenduse andmestruktuur ja andmebaasipäringud pole ülemäära keerulised ega vaja erilahendusi, millest tulenevalt otsustati kasutada autorile tuttavamat MariaDB tehnoloogiat.

4.3.4 Tehisintellekt

OpenAI keelerobotit ChatGPT[3] kasutati arenduse käigus tekkinud küsimuste lahendamiseks, tehnoloogiate kasutusviisi selgitamiseks, näidete genereerimiseks ja koodist

[1] https://spring.io/projects/spring-data-jpa
[2] https://www.liquibase.com/
[3] https://chat.openai.com/chat

vigade leidmiseks. Keelerobotist saadud vastustesse suhtuti skeptiliselt ning neid valideeriti teistest allikatest saadud info põhjal.

Github Copilotit[4] kasutati koodi genereerimiseks. See analüüsib olemasolevat koodi ning genereerib kirjutamise ajal koodijuppe, mida saab enda kirjutatud koodi juurde lisada. Tööriist aitas kiiremini kirjutada lihtsamat koodi, kuid rakenduse põhiline ja keerulisem loogika tuli siiski iseseisvalt implementeerida.

[4]https://github.com/features/copilot

5. Arenduskäik

Siin peatükis kirjeldatakse arenduskäigu olulisemaid etappe, arendamisel tekkinud probleeme ning tähtsamaid lahenduskäike. Olulisemad loodud klientrakenduse vaated on nähtavad Lisas 3 oleval lingil. Valminud andmebaasi diagramm on esitatud Lisas 4.

5.1 Seadistamine ja baasstruktuuri loomine

Koodi varunduse ja projektihalduse eesmärgil loodi Githubis eraldi repositooriumid *frontend*i ja *backend*i jaoks. Varasema kogemuse põhjal eelistas autor, et need süsteemi osad oleks üksteisest eraldatud, sest nii on lähtekoodist parem ülevaade. Serverrakenduse jaoks seadistati Spring Boot ja andmebaas. Seejärel loodi esialgsed andmestruktuurid spordiklubide, õpilaste, treenerite, treeningväljakute ja treenerite andmete hoidmiseks ning implementeeriti äriloogika andmete salvestamiseks ja klientrakenduse päringute töötlemiseks.

Järgmisena otsustati leida lahendus korduvate treeningute loomiseks. Selleks uuriti olemasolevaid lahendusi, kulutati tunde foorumeid lugedes ning joonistati paberile kümneid erinevaid ideid. Ülesande muutis keerukaks asjaolu, et treeninguga on seotud õpilased, treenerid, treeningtasud, väljakud ja muud andmed. Peale pikka katsetamist jõuti järelduseni, et tegemist on hetkel vähetähtsa probleemiga, mille lahendamine ei mängi rolli peamiste eesmärkide saavutamisel. Alternatiivina lisati võimalus kopeerida olemasolev treening uutele kuupäevadele.

Seejärel seadistati klientrakenduse jaoks Vue.js ning PrimeVue komponentide teek. Kasutusele võetud mall oli kirjutatud Javascriptis, mistõttu lisati projekti TypeScript, et oleks võimalus tüübikindlamat koodi kirjutada. Omajagu aega kulus kalendri disainimisele, sest see oli PrimeVuest eraldiseisev komponent.

5.2 Arveldamise struktuur

Tasude määramiseks ja arvete loomiseks tuli arvestada, et enamasti koostatakse arved alles peale treeningperioodi lõppu. See tähendab, et tasud oleks vaja kirja panna enne arvete loomist. Kasutusele võeti mudel, mis koosneb artiklitest, teenustest, arvetest, maksetest ja maksekontodest. Süsteem põhineb suuresti raamatupidamisprogrammides kasutataval struktuuril. Tänu sellele on lihtsam loodav rakendus nendega integreerida ning kasutajale

on arveldamise struktuur kergemini hoovatav.

Järgmisena kirjeldatakse täpsemalt arveldamise mudeli andmeobjekte.

Artikkel: Artikli abil määratakse õpilasele rakendatud tasu liik. Artiklil on kood ja valikuliselt ka kirjeldus. Artikkel on vajalik arvete raamatupidamisprogrammi kopeerimiseks ning määrab selles kontod, millel tehingut kajastatakse. Rakenduses oleva artikli kood peab kattuma raamatupidamisprogrammis oleva artikli koodiga, et arvete kopeerimine toimiks.

Maksekonto: Maksekontoks võib olla näiteks sularahakassa või pangakonto. Selle abil saab grupeerida õpilaste tehtud makseid. Maksekontol on nimetus, tüüp ja valikuliselt kirjeldus. Maksekonto tüübiks saab määrata "arveväline", mis võimaldab kirja panna makseid, mille jaoks kasutatakse eraldi arvestust.

Makse: Makse on seotud maksekontoga ning arve või teenusega. Sellel on summa, kuupäev ja valikuliselt märkmed.

Arve: Arve koosneb teenustest, sellel on klubisiseselt unikaalne number, kuupäev, maksetähtaeg, käibemaksumäär, viivis ning valikuliselt täpsustavad märkmed. Arve on seotud ühe õpilasega ning sellele saab lisada makseid.

Teenus: Teenuseks võib olla näiteks treeningtasu, liikmemaks või reketi renditasu. Sisuliselt on tegemist arve reaga, mida on võimalik määrata ka enne arve koostamist.

Teenus on seotud ühe artikli ja õpilasega, valikuliselt ka treeningu, makse või arvega. Sellel on kirjeldus, mis kuvatakse arvel, hind ning arvestusaeg, et oleks selge, mis ajaperioodi jaoks see on loodud. Teenusele saab lisada makse, et märkida enne arve koostamist tasutud summasid.

Teenusel on kolm staatust: tava, tühistatud, arveväline. Tühistatud teenust ei kajastata arvetel ega aruannetes. Arvevälist teenust pole võimalik lisada arvetele, aruannetes kajastatakse seda muudest teenustest eraldi.

5.3 Serverrakendusele päringute tegemine

Esialgu kasutati serverrakendusele päringute tegemiseks Axiost[1] ning korduvate päringute vähendamiseks ja komponentide vahel andmete jagamise lihtsustamiseks võeti kasutusele Pinia[2]. Andmevahetuse struktuuri loomisel lähtuti uurimise käigus leitud artiklist, milles kasutatud koodi ülesehitus oli loogiline ja lihtsasti loetav [13]. Päringutest saadud andmetele kirjutati TypeScript tüübid käsitsi.

Arenduse käigus tekkis üha rohkem andmestruktuure, andmete uuendamine Piniaga ning TypeScript tüüpide haldamine muutus tülikaks. Olukorra parandamiseks võeti kasutusele OpenAPI TypeScript[3] ning OpenAPI Fetch[4] teegid. OpenAPI Typescript võimaldab OpenAPI spetsifikatsiooni põhjal genereerida TypeScript tüübid API-ga suhtlemisel kasutatavatele andmetele. OpenAPI Fetch muudab serverrakendusele päringute tegemise tüübikindlaks.

OpenAPI spetsifikatsioon genereeriti springdoc-openapi[5] Java teegi abil, mis võimaldab spetsifikatsiooni luua serverrakenduse koodi põhjal. Selgema dokumentatsiooni loomiseks lisati kirjeldusi API otspunktidele ja andmestruktuuridele. Tulemust sai näha ka läbi eraldi kasutajaliidese (vt näidist Joonis 4), mis lihtsustas edasist arendusprotsessi.

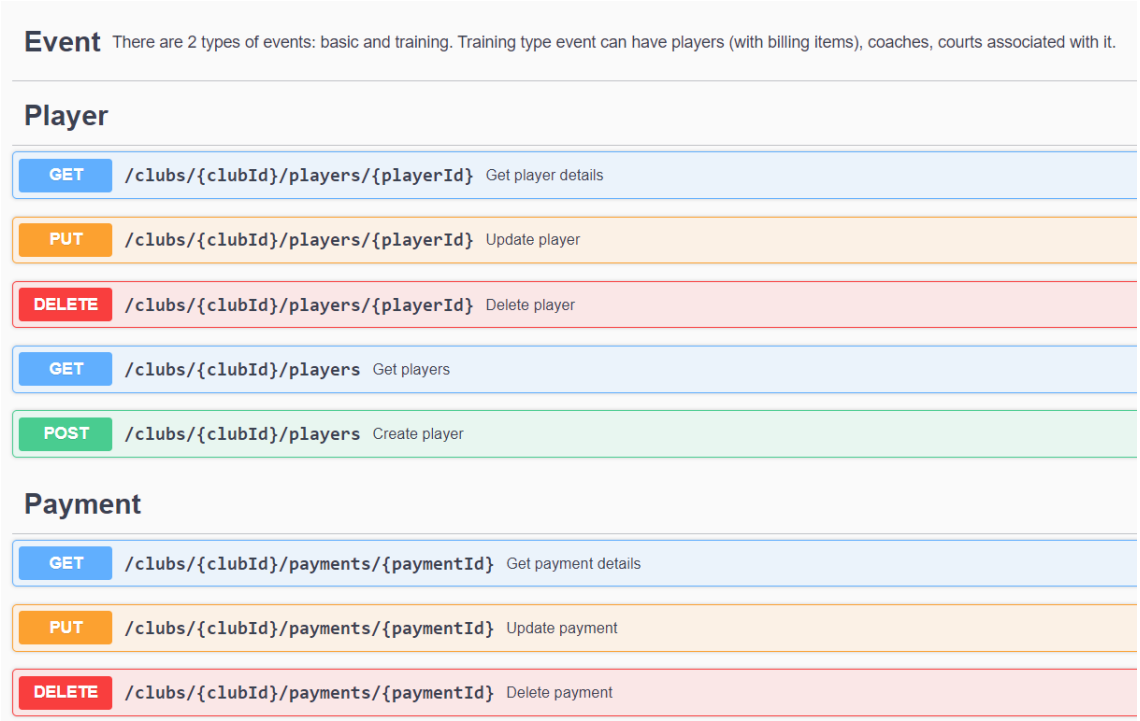
Pinia asendati järk-järgult Tanstack Queryga[6], mis teeb serverist saadud andmete vahemällu salvestamise ning salvestatud andmete uuendamise üpris mugavaks.

5.4 Autentimine

Sisselogimiseks saadetakse serverrakendusele kasutajatunnus (e-posti aadress) ja parool. Paroolist arvutatakse räsi ning kasutajatunnuse ja parooli räsi kombinatsiooni võrreldakse andmebaasis olevate andmetega. Kui kasutajatunnus ja parool on õiged, siis genereeritakse kaks JWT (*JSON Web Token*) tõendit - lühiajaline, mida kasutatakse päringute autentimiseks ning pikaajaline, mida kasutatakse lühiajalise tõendi aegumisel uue saamiseks.

Serverrakenduselt saadud JWT tõend sisaldab ka autoriseerimiseks vajalikku nimekirja spordiklubidest, mille andmetele on kasutajal õigus ligi pääseda.

[1] https://axios-http.com/docs/intro
[2] https://pinia.vuejs.org/introduction.html
[3] https://openapi-ts.pages.dev/6.x/introduction
[4] https://openapi-ts.pages.dev/openapi-fetch/
[5] https://springdoc.org/
[6] https://tanstack.com/query/latest/docs/framework/vue/overview



Joonis 4. Kasutajaliides OpenAPI spetsifikatsiooni jaoks.

JWT tõendid saadetakse küpsistena, tänu millele ei pea klientrakenduses looma lahendust nende salvestamiseks ja päringutele lisamiseks. Küpsiseid haldab veebibrauser, seeläbi on süsteem ka turvalisem.

5.5 Spordiklubide andmete eraldamine

Rakendus on mõeldud kasutamiseks mitmele spordiklubile. Andmete eraldamiseks kaaluti järgmisi meetodeid:

1. Andmebaas iga klubi jaoks
2. Üks andmebaas, erinev andmebaasiskeem igale klubile
3. Üks andmebaas ja andmebaasiskeem, seos klubiga määratakse olemites

Esimesed kaks valikut nõuavad keerulisemat ja kallimat infrastruktuuri, samas ei pea igas andmebaasipäringus klubi põhjal andmeid filtreerima. Siiski otsustati kolmanda variandi kasuks, et arendusprotsess oleks kiirem ja arusaadavam.

5.6 Ajahetke ja rahasumma andmetüübid

Ajahetkede töötlemiseks otsustati kasutada Java Instant tüüpi, andmebaasis DATETIME tüüpi ning klientrakendusega suhtlemisel String tüüpi ISO 8601 vormingus[7]. Aega hoitakse UTC-s, ainult klientrakenduses kuvatakse ajahetki Eestis kehtivas ajavööndis. Mugavamaks kasutamiseks konverteeritakse klientrakeduses ajahetked Date tüüpi.

Rahasummade jaoks kasutab andmebaas Decimal, serverrakendus BigDecimal ja klientrakendus Number tüüpi. Klientrakenduses ei tehta rahasummadega arvutusi, seega Number tüübi täpsus ei mängi antud hetkel rolli. Vajadusel saab kasutada näiteks Big.js[8] teeki, millega välditakse Number tüübi arvutustest tulenevaid ebatäpsusi.

Esialgu otsustati, et rahasummasid arvestatakse kahe murdosa täpsusega, kuid peagi selgus, et käibemaksu arvestamine ja kuutasu jagamine treeningute arvuga nõuab suuremat täpsust. Näiteks jagades 100€ kolme treeningu vahel, on kahekohalise murdosa täpsusega tulemuseks 33,33€. Korrutades seda kolmega, saame 99.99€. Kliendile esitatud arvel oleks 100€ asemel 99,99€. Õnneks on see väheoluline probleem, aga siiski suurendati rahasummad nelja komakoha täpsuseks.

Kõik tasud salvestatakse andmebaasi ilma maksudeta, käibemaks arvutatakse arvel kuvamiseks eraldi.

5.7 Arve summad

Arvel tuleb näidata erinevaid summasid ja jääke. Esialgu arvutas serverrakendus iga arvel tehtud muudatuse korral kõik summad uuesti ning salvestas need andmebaasi. Üsna pea selgus, et sellise meetodiga kaugele ei jõua, sest iga muudatust on väga raske jälgida ning lahendus kubiseb vigadest.

Otsustati, et summad arvutatakse iga päringu korral uuesti ning summeerimiseks kasutatakse ka andmebaasisüsteemis sisalduvaid funktsioone. Andmete hulk on väike ning operatsioon pole ülemäära keeruline, seega ei tekita korduv arvutamine praeguses arendusfaasis probleeme. Joonis 5 sisaldab serverrakenduse koodi arve vahesumma, tasutud summa ja kogusumma arvutamiseks.

[7]https://www.iso.org/iso-8601-date-and-time-format.html
[8]https://github.com/MikeMcl/big.js

```
@Formula("(SELECT COALESCE(SUM(bi.unit_price), 0) FROM
    billing_item bi WHERE bi.invoice_id = id)")
private BigDecimal totalExcTax = BigDecimal.ZERO;

@Formula("(SELECT COALESCE(SUM(p.amount), 0) FROM
    payment p WHERE p.invoice_id = id)")
private BigDecimal totalPaid = BigDecimal.ZERO;

@Transient
public BigDecimal getTotalTax() {
    return totalExcTax.multiply(tax.getRate())
            .setScale(2, RoundingMode.HALF_UP);
}
```

Joonis 5. Serverrakenduse kood arve vahesummade arvutamiseks.

5.8 Integratsioon Merit Aktiva tarkvaraga

Merit Aktiva programmi arvete lisamiseks tutvuti selle API dokumentatsiooniga, tehti testpäringuid ning implementeeriti vajalikud funktsioonid klient- ja serverrakendustes. Merit Aktiva API dokumentatsioon oli väheste selgitustega, päringu ebaõnnestumisel ei täpsustatud, mis läks valesti. Katsetati erinevaid päringuid, kuni õnnestus aru saada, milliseid andmeid API server nõuab.

5.9 Arved PDF-vormingus

Prototüübis puudub õpilastele arvete saatmise võimalus, seda tehakse läbi Merit Aktiva raamatupidamisprogrammi, milles on ka funktsioon arvete PDF-vormingus allalaadimiseks. Siiski lisati ka loodavasse rakendusse võimalus PDF arveid alla laadida, et rakendust oleks võimalik kasutada ilma Merit Aktivata.

Selleks leiti vabalt saadaval Invoice Generator API[9]. Antud lahenduse peale ei saa küll pikaajaliselt loota, sest generaatori tasuta kasutamine võidakse ühepoolselt lõpetada.

5.10 Testserver

Rakenduse demonstreerimiseks ning kliendile katsetamiseks laeti loodud süsteem test-serverisse. Selleks kasutati Dockerit ning AWS EC2 virtuaalserverit. *Frontend*'i jaoks loodi konteiner, milles töötav NGINX veebiserver edastab klientrakenduse faile ning suunab

[9]https://invoice-generator.com/developers

päringuid serverrakendusele. *Backend*'i konteineris töötab Spring Boot serverrakendus, eraldi konteiner on loodud ka andmebaasile.

Virtuaalserveri tüübiks on t3.micro, millest piisab prototüübi näitamiseks, kuid 1GB muutmälust jääb kohati juba praegu väheks.

6. Validatsioon

Töö valideerimiseks vaadeldi analüüsi käigus seatud nõuete täitmist, klient katsetas loodud lahendust ning hindas selle kasulikkust. Koguti tagasisidet, et saada ülevaade ka rakenduse puudustest.

6.1 Nõuete täitmine

Kliendile tehti demo rakenduse funktsioonidest. Selle käigus käidi ükshaaval läbi rakendusele seatud nõuded ning veenduti, et need said täidetud.

6.2 Kliendi hinnang

Klient lõi rakenduses tenniseklubi õpilaste ja treenerite nimekirjad, määras artiklid ja maksekontod. Seejärel sisestas möödunud kuul toimunud treeningute andmed ning genereeris nende põhjal õpilastele arved. Veendumaks tulemuse õigsuses, võrreldi rakenduses genereeritud andmeid käsitsi arvutatud summadega. Need olid võrdsed, millest võib järeldada, et lahendus toimib. Valideeriti ka treenerite töötundide arvestus ning katsetati alternatiivse arvepidamise lahendust.

Kõige selgemini saab rakenduse kasulikkust hinnata säästetud aja põhjal. Kliendil kulus käsitsi iga õpilase treeningtasude arvutamiseks ja arvete koostamiseks üle 20 tunni. Rakendusse nimekirjade sisestamine, möödunud kuu treeningute märkimine ja arvete genereerimine võttis vaid 1,5 tundi. Edaspidi pole vaja õpilaste ega treenerite nimekirjasid enam luua ning treeninguid märgitakse kuu vältel jooksvalt, seega on arveldamisele kuluv aeg veelgi väiksem. Selle põhjal saab öelda, et loodud lahendus täidab seatud eesmärke.

6.3 Tagasiside

Prototüüpi näidati ning anti katsetada veel kolmele treenerile. Tagasiside oli positiivne, arutati edasiste plaanide üle ning lepiti kokku, et eesoleva suve jooksul testitakse rakendust põhjalikumalt ning lisatakse uusi võimalusi, mida sügisest kasutusele võtta. Kohati tekitas segadust arvete ja teenuste olemus, mille tõenäoliselt saab lahendada täiendavate selgituste ja juhistega kasutajaliideses. Murekohana toodi välja ka treeningu detailvaadet, mida oli ebamugav kasutada. Treeningtasude kiiremaks sisestamiseks paluti luua funktsioon, mis võimaldaks ühe korraga kõikide treeningus osalejate vahel jagada fikseeritud treeningtasu.

7. Edasiarendused

Järgmises arendusfaasis on plaan kirjutada olemasolev kood kvaliteetsemaks, täiustada kasutajaliidest ning leida jätkusuutlik lahendus rakenduse pilves jooksutamiseks. Rõhk pannakse ka automaattestide loomisele.

Mõned olulisemad funktsioonid, mida plaanitakse lisada:

- Automaatne andmebaasi varundus
- Merit Aktiva programmiga maksete sünkroniseerimine
- Arvete saatmine õpilase e-posti aadressile
- Treeningusse registreerimise vorm, mida õpilased saavad kasutada
- Treeneri roll, milles on piiratud funktsioonid
- Mobiilisõbralik kasutajaliides
- CI/CD kontrollitumateks väljalasketeks
- Detailne logimine ja süsteemi staatuse automaatne kontroll, et tekkinud vigu saaks kiiresti lahendada

Laiema ülevaate saamiseks võetakse ühendust rohkemate treeneritega, kellele demonstreeritakse täiendatud prototüüpi. Kogutakse tagasisidet ning kaardistatakse uued nõuded. Sügiseks on plaan valmis saada rakenduse esimene laiemale publikule suunatud versioon, milles on ka vaated õpilaste jaoks.

Pikema perspektiivi unistusteks on integreeritud makseviisid, lehekülg rakenduse müümiseks ja turundamiseks, Google Calendar integratsioon, SMS teavitused, SimplBooks jm raamatupidamistarkvarade integratsioon.

8. Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli luua veebirakenduse prototüüp spordiklubi õpilaste, treenerite, treeningute andmete ja õpilastega seotud arveldamise mugavaks haldamiseks, et säästa spordiklubi juhtimisele kuluvat aega. Töö skoobi piiramiseks keskenduti reketiklubidele ja klubisiseste vaadete loomisele ning nõuded seati peamiselt ühe konkreetse padeli- ja tenniseklubi (kliendi) vajaduste põhjal, kuid suheldi ka teiste spordiklubide treeneritega.

Kõigepealt uuriti kliendi praegust süsteemi, selle murekohti ning peamisi vajadusi olukorra parendamiseks. Kliendi kasutatav süsteem oli aeganõudev ja vigaderohke, soov oli leida paindlik lahendus. Tehti taustauuring olemasolevatele lahendustele, kuid sobivat ei leitud. Probleemideks olid näiteks piiratud arveldamise struktuur, eestikeelse kasutajaliidese puudumine ja liiga keeruline kasutajaliides. Seejärel seati nõuded loodavale rakendusele, pandi paika töö tehnoloogilise osa alus ning arendati valmis prototüüp.

Tulemuse valideerimiseks katsetas klient loodud lahendust, veenduti, et nõuded said täidetud ning rakenduse funktsioonid toimivad. Katsetamiseks loodi rakendusega ühe kuu arved ning selle käigus selgus, et loodud lahendus vähendab ajakulu märkimisväärselt. Lõpetuseks pandi paika edasine arendusplaan.

Töö eesmärk sai täidetud ning autor omandas uusi teadmisi projekti planeerimisest, analüüsist ja programmeerimisest. Töö käigus loodud prototüübiga seati alus suuremale projektile, mille käigus soovitakse luua täislahendusena rakendus spordiklubi haldamiseks.

Kasutatud kirjandus

[1] Sportlyzer. *Sportlaste arengu monitoorimise ja tiimi haldamise tarkvara | Sportlyzer*. [Kasutatud: 07-05-2024]. URL: https://www.sportlyzer.com/et/.

[2] Playtomic Manager. *New Private Classes section in Playtomic Manager*. [Kasutatud: 12-11-2023]. URL: https://help.playtomic.com/en/articles/6235231-new-private-classes-section-in-playtomic-manager.

[3] Booklux. *Parim veebipõhine broneerimissüsteem: tarkvara ja tasuta rakendus*. [Kasutatud: 10-05-2024]. URL: https://www.booklux.com/et.

[4] Inc VMware. *Spring boot*. [Kasutatud: 28-10-2023]. URL: https://spring.io/projects/spring-boot.

[5] Inc VMware. *Why Spring?* [Kasutatud: 12-11-2023]. URL: https://spring.io/why-spring.

[6] OpenJS Foundation. *Introduction to Node.js*. [Kasutatud: 12-11-2023]. URL: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs.

[7] *Stack Overflow Developer Survey 2023*. URL: https://survey.stackoverflow.co/2023.

[8] Vue.js. *Introduction*. [Kasutatud: 28-10-2023]. URL: https://vuejs.org/guide/introduction.html.

[9] Inc Amazon Web Services. *What's the Difference Between MariaDB and MySQL?* [Kasutatud: 28-10-2023]. URL: https://aws.amazon.com/compare/the-difference-between-mariadb-vs-mysql/.

[10] MariaDB. *MariaDB vs MySQL*. [Kasutatud: 12-11-2023]. URL: https://mariadb.com/database-topics/mariadb-vs-mysql/.

[11] Oracle. *MySQL 8.0 Reference Manual. What is MySQL?* [Kasutatud: 12-11-2023]. URL: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html.

[12] Inc Amazon Web Services. *What's the Difference Between MariaDB and PostgreSQL?* [Kasutatud: 10-05-2024]. URL: https://aws.amazon.com/compare/the-difference-between-mariadb-and-postgresql/.

[13] Israel Diaz Zapata. *Axios + Vue.js 3 + Pinia, a 'comfy' configuration you can consider for an API REST*. [Kasutatud: 09-05-2024]. URL: https://medium.com/@bugintheconsole/axios-vue-js-3-pinia-a-comfy-configuration-you-can-consider-for-an-api-rest-a6005c356dcd.

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

Mina Paul-Bryan Tanilsoo

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Spordiklubi treeningute haldamise veebirakenduse arendus", mille juhendaja on Tarvo Treier
   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.

27.05.2024

[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.

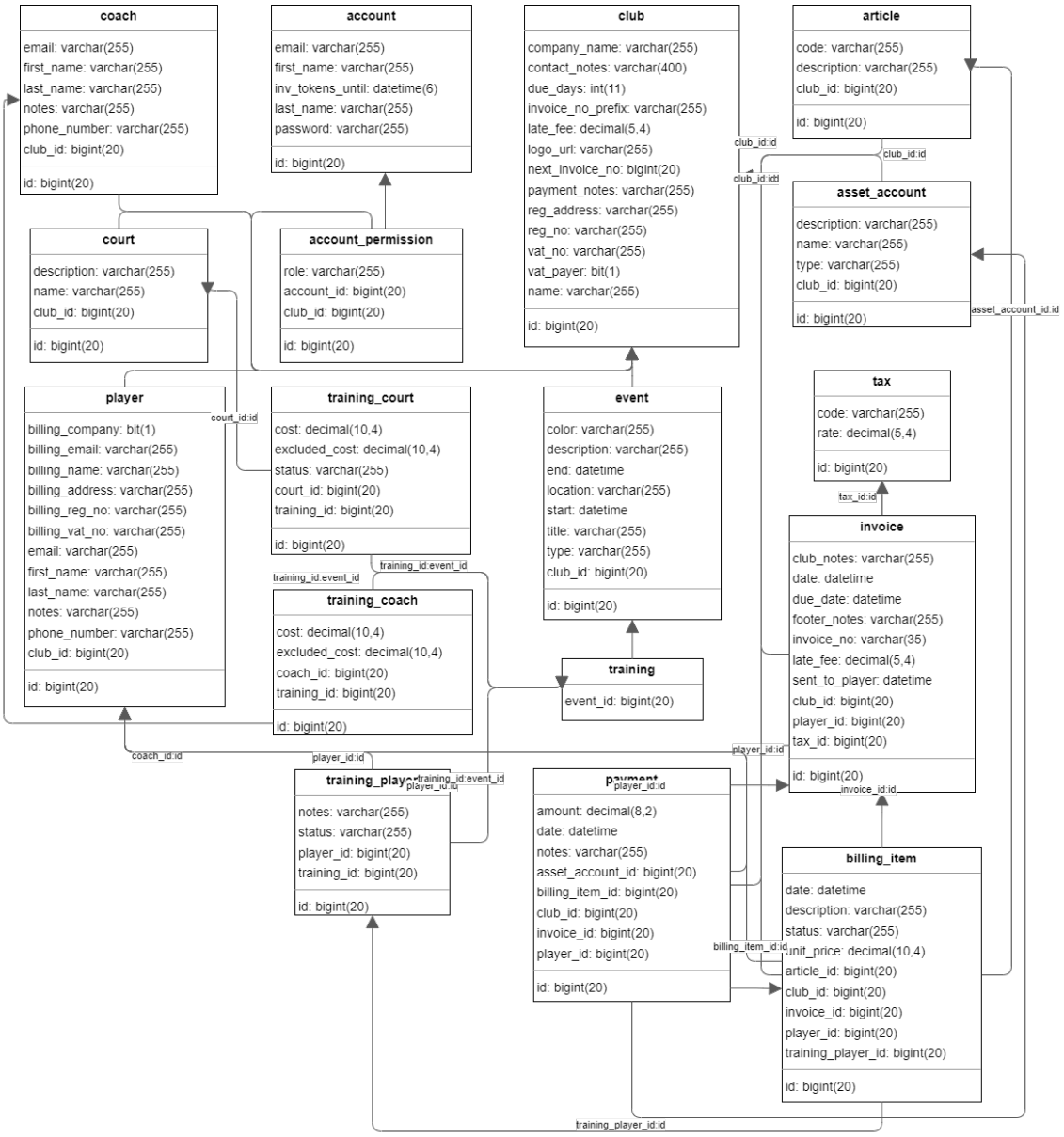
Lisa 2 – Sõrestikmudel kalendervaatest

Joonis 6. Figmas loodud kalendervaate sõrestikmudel.

Lisa 3 – Link kuvatõmmistele kasutajaliidese peamistest vaadetest

https://drive.google.com/drive/folders/1g-VD7rmHJvcENZPjA1qGHMu4mBOdq65f?usp=sharing

Lisa 4 – Andmebaasi diagramm



Joonis 7. Andmebaasi diagramm.