TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Saamo Kurvits 213064IAIB

Välitööde rakenduse integreerimine automatsiooniplatvormiga

Bakalaureusetöö

Juhendaja: Siim Rebane

BSc

Tallinn 2025

Autorideklaratsioon

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

Autor: Saamo Kurvits

03.06.2025

Annotatsioon

Lõputöö eesmärgiks on luua Syda kalendrirakendusele integratsioon väliste äriprotsessidega, mis võimaldab klientidel paremini hallata rakenduses olevat infot ning töödelda seda vastavalt oma ärilistele vajadustele.

Probleem, mida lõputöö lahendab, on erinevate ettevõtete äriloogika protsesside jaoks lahenduse loomine, mis võimaldab töödelda rakenduse põhilisi andmeid vastavalt ärilistele vajadustele. Tavapärane lahendus on siiani olnud rakenduse lokaalsete formaatide põhine, kus töö raportid ning tööde haldus on vastavalt rakendusesisesele äriloogikale ning ei paku nii palju võimalusi erinevateks töövoogudeks. Tihtipeale on vajadus kliendil võtta platvormil olev info ning kasutada seda edasistes protsessides vastavalt oma vajadustele. Lahendus on integratsioon, mis võimaldab klientidel automaatselt pärida infot kalendrirakendusest ning töödelda andmeid vastavalt oma ärilistele nõuetele ning eesmärkidele.

Lõputöö autori ülesanne on luua vajalikud rakenduse laiendused, et võimaldada selle integreerimist väliste äriprotsessidega edasiste andmeoperatsioonide jaoks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 29 leheküljel, 7 peatükki, 2 joonist, 2 tabelit.

Abstract
Field Service Application Integration with Automation Platform

The primary objective of this thesis is to develop an integration between the Syda calendar application and external business processes, enabling clients to better manage and process the information stored in the application according to their business needs.

The problem addressed by this thesis is the development of a solution for various business processes of different companies, which enables the processing of the primary data of the application in accordance with business needs. The conventional solution has so far been based on local formats of the application, where work reports and work management are in accordance with the internal business logic of the application, and do not offer as many options for different workflows. Often, there is a need for a client to retrieve information from the platform and use it in subsequent processes according to their needs. The solution is an integration, which enables clients to automatically retrieve information from the calendar application and process the data according to their business requirements and objectives.

The author's task in this thesis is to develop the necessary extensions to the application to enable its integration with external business processes for subsequent data operations.

The thesis is in Estonian and contains 29 pages of text, 7 chapters, 2 figures, 2 tables.

Lühendite ja mõistete sõnastik

API	Rakendusliides (*Application Programming Interface*)
CRUD	Loomine, lugemine, uuendamine, kustutamine (*Create, Read, Update, Delete*)
HTTP	Hüperteksti edastamise protokoll (*Hypertext Transfer Protocol*)
IDE	Integreeritud programmeerimiskeskkond (*Integrated Development Environment*)
iPaaS	Integratsiooniplatvorm teenusena (*Integration Platform as a Service*)
JSON	Lihtsustatud andmevahetusvorming (*JavaScript Object Notation*)
JWT	Interneti standard andmete loomiseks koos digitaalallkirja ja/või krüpteerimisega (*JSON Web Token*)
REST	Esitusoleku siire, tarkvaraarhitektuuri laad (*Representational State Transfer*)
URL	Veebis interneti objektide määratlemise standard (*Uniform Resource Locator*)

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Käesolev bakalaureusetöö keskendub veebipõhise välitööde haldamise tarkvaralahenduse integreerimisele väliste äriprotsessidega. Syda on kalendripõhine välitööde halduse rakendus, mis võimaldab tööde planeerijatel ülesandeid delegeerida ning töötajatel neile määratud töid jälgida ja teostada [1]. Lisaks sisaldab platvorm rendimoodulit rendivarustuse ja sellega seotud dokumentatsiooni ning arvete haldamiseks. Süsteemi laiem eesmärk on ettevõtete tööprotsesside optimeerimine kasutajasõbraliku ja integreeritud lahenduse kaudu.

Tänapäeva ettevõtted tuginevad digitaalsetele töövahenditele üha enam, et oma protsesse tõhusamalt juhtida. Kuna Syda rakendus on osa paljude ettevõtete igapäevastest äriprotsessidest, on selle andmed sageli vajalikud ka teistes süsteemides. Praegu peavad mitmed Syda kasutajad sisestama rakenduses leiduvaid andmeid käsitsi erinevatesse välistesse ärisüsteemidesse. See on aeganõudev ja ebaefektiivne, suurendab vigade riski, muudab äriprotsessid jäigaks ega arvesta iga ettevõtte unikaalseid vajadusi ning halvendab kasutajakogemust, kuna töötajad peavad pidevalt erinevate süsteemide vahel liikuma.

Lõputöö eesmärk on arendada Syda välitööde rakendusele integratsioon väliste äriprotsessidega, kasutades selleks levinud automatiseerimisplatvorme nagu Zapier, Make või n8n [2], [3], [4]. Selle integratsiooni peamine siht on vähendada manuaalset andmesisestust, optimeerida töövooge ning parandada kasutajakogemust, võimaldades Syda rakenduse andmete sujuvat sünkroniseerimist teiste süsteemidega. Probleemi lahendamiseks on vajalik luua standardiseeritud integratsioon, mis võimaldab automatiseeritud andmevahetust ja vähendab käsitsi andmete sisestamise vajadust.

Oodatav tulemus on toimiv ja testitud integratsioonilahendus Syda välitööde rakenduse ja väliste äriprotsesside vahel, mis vähendab käsitsi andmesisestuse vajadust ja vigade hulka, muudab ettevõtete tööprotsessid sujuvamaks ja efektiivsemaks ning parandab kasutajakogemust. Lõpptulemusena valmib dokumentatsioon, mis võimaldab lahenduse

edasist arendamist ja laiemat rakendamist.

2 Analüüs

Käesolevas peatükis analüüsitakse Syda rakenduse väliste äriprotsessidega integreerimise vajadust, hinnatakse olemasolevate lahenduste piiranguid ja võimalikke uusi lähenemisviise ning põhjendatakse valitud integratsioonistrateegiat.

2.1 Vajadus paindliku integratsioonilahenduse järele

Käesoleva lõputöö raames arendatava lahenduse vajadus tuleneb tõdemusest, et iga ettevõtte äriprotsessid on unikaalsed. Kuigi teatud protsessid võivad eri ettevõtetes sarnaneda, esineb praktikas alati erinevusi nii eelistustes, töövoogudes kui ka oodatavates tulemustes. See asjaolu on aktuaalne ka Syda tarkvara klientide seas, kes on väljendanud selget soovi saada võimalus eksportida neile olulisi andmeid platvormilt mugavalt ja paindlikult, et neid seejärel sisestada teistesse tarkvarasüsteemidesse või kasutada edasiste äriprotsesside automatiseerimiseks.

2.2 Praeguste lahenduste piirangud

Syda platvormi senised andmeekspordi funktsioonid on valdavalt piirdunud standardiseeritud ning jäiga struktuuriga lahendustega, millest tüüpilisimaks näiteks on ettemääratud vormingus Exceli tabelitena genereeritavad tööde ja töötajate raportid [5]. Kuigi niisugused raportid on pakkunud klientidele teatud lisaväärtust, on nende peamiseks puuduseks osutunud ebapiisav paindlikkus. Klientidel on ilmnenud erisuguseid detailseid nõudmisi ja soove seoses eksporditavate andmetabelite struktuuri ning sisulise koosseisu modifitseerimisega. Need spetsiifilised vajadused on aga sageli sattunud konflikti teiste klientide ootuste või platvormi üldiste standarditega.

Andmete eksportimine rangelt fikseeritud tabelikujul on oma olemuselt jäik ega soodusta andmete automatiseeritud edasitöötlust mitmesugustes välistes süsteemides. Selline formaat tingib märkimisväärse täiendava manuaalse töö vajaduse andmete kogumisel, valideerimisel ja ettevalmistamisel järgnevateks, sageli ettevõttespetsiifilisteks ja variee-

ruvateks protsessideks. Nimetatud piirangutest tulenevalt on selgelt tekkinud vajadus arendada välja uudne lahendus. See lahendus peaks võimaldama klientidel platvormilt ekstraheerida neile vajalikku informatsiooni täies mahus ja detailsuses, tagades seejuures vabaduse töödelda saadud andmeid vastavalt omaenda unikaalsetele eesmärkidele, nõuetele ja integreeritavatele sihtsüsteemidele.

2.3 Integratsioonivõimaluste hindamine

Üheks võimalikuks lahenduseks andmete väljastamiseks oleks üldise rakendusliidese loomine. Selline liides võimaldaks ettevõtetel programmaatiliselt pärida neile vajalikku informatsiooni ning integreerida see oma olemasolevatesse süsteemidesse. Samas ei pakuks pelgalt rakendusliidese olemasolu piisavat väärtust suurele osale Syda klientidest, kellest paljudel puuduvad piisavad tehnilised ressursid või oskusteave, et rakendusliidest iseseisvalt ja efektiivselt kasutada.

Selle probleemi leevendamiseks on tänapäeval laialdaselt kasutusel mitmesugused automatiseerimisplatvormid, mis võimaldavad ka vähesema tehnilise taustaga kasutajatel luua keerukaid automatiseeritud töövoogusid erinevate rakenduste vahel. Need platvormid toimivad vahekihina, lihtsustades oluliselt rakendusliidestega suhtlemist ja andmete edastamist paljudesse levinud tarkvaradesse. Seetõttu otsustati Syda platvormi jaoks luua nii rakendusliides kui ka integratsioon vähemalt ühe laialt levinud automatiseerimisplatvormiga.

Allpool analüüsitakse levinumaid automatiseerimisplatvorme, hinnates nende kasutusmugavust, funktsionaalsust, kulutõhusust ja integratsioonivõimalusi. Kulutõhususe hindamisel on üheks indikaatoriks võetud platvormide hinnastuspõhimõtted ja maksumus ligikaudu 10 000 operatsiooni või ekvivalendi kohta kuus. Tuleb ka arvestada, et operatsiooni mõiste võib platvormiti erineda.

2.3.1 Zapier

Zapier on üks tuntumaid ja laialdasemalt kasutatavaid automatiseerimisplatvorme [2]. See paistab silma erakordselt suure, enam kui 8000 rakenduse ja teenuse integratsioonide valikuga ning väga kasutajasõbraliku liidesega, mis teeb selle kättesaadavaks ka algajatele ilma tehniliste eelteadmisteta. Platvormi peamiseks tugevuseks on lihtsus ja kiire seadistamisvõi-

malus. Zapier'i ärimudel põhineb ülesannete arvul, kus iga andmetöötlus samm loetakse eraldi ülesandeks. Kuigi Zapier pakub tasuta paketti, on see funktsionaalsuselt ja ülesannete mahult väga piiratud ning keerukamate, mitmest sammust koosnevate automatiseeringute kasutamine eeldab tasulist paketti [6].

2.3.2 Make

Make, varasemalt tuntud kui Integromat, on võimas automatiseerimisplatvorm, mis on hinnatud oma paindliku ja visuaalselt intuitiivse töövoogude loomise keskkonna poolest [3]. See võimaldab kasutajatel luua keerukaid ja mitmeharulisi stsenaariumeid. Make'i peamiseks eeliseks on selle operatsioonipõhine hinnastusmudel, kus iga mooduli tegevus loetakse üheks operatsiooniks. See mudel pakub tihti Zapieriga võrreldes paremat hinna ja funktsionaalsuse suhet, eriti keskmise ja suurema keerukusega töövoogude puhul, ning võimaldab täpsemat kontrolli kulude üle. Make pakub üle 2000 integratsiooni ning on tugev andmete teisendamisel ja töötlemisel töövoogude sees. Platvormil on väga hea tasuta pakett, mis võimaldab luua ning hallata lihtsaid töövooge. Tasulised paketid on väga mõistliku hinnastusega ning võimaldavad kohati kuni 10 korda odavamalt saavutada samasuguse töövoogude mahu mis Zapieris [7].

2.3.3 Bubble

Bubble on peamiselt tuntud kui koodivaba platvorm veebirakenduste loomiseks, mitte niivõrd spetsialiseerunud automatiseerimisplatvormina [8]. Selle peamine tugevus seisneb võimaluses ehitada keerukaid veebirakendusi ilma programmeerimisoskuseta ning seejärel integreerida need rakendused teiste teenustega läbi kolmandate osapoolte pistikprogrammide. Kuigi Bubble võimaldab luua integratsioone, võib see spetsiifiliselt automatsioonide ning töövoogude loomise ja haldamise kontekstis olla vähem paindlik või kulutõhus kui pühendatud iPaaS lahendused [9]. Selle fookus on rohkem rakenduse enda ehitamisel, kus integratsioonid on üks osa laiemast funktsionaalsusest.

2.3.4 n8n

n8n on paindlik ja võimas automatiseerimisplatvorm, mis kogub kiiresti populaarsust, eriti tehnilisemate kasutajate ja arendajate seas [4]. Selle üks peamisi eristavaid jooni on avatud lähtekoodiga olemus, mis võimaldab platvormi ise majutada. Isemajutus annab

täieliku kontrolli andmete üle, piiramatud töövoogude käivitused ning võib osutuda väga kuluefektiivseks lahenduseks. Lisaks pakub n8n ka pilvepõhist teenust, mille hinnastus põhineb töövoo käivituste arvul, mitte üksikutel operatsioonidel või ülesannetel, mis võib olla erakordselt soodne keerukate ja mahukate töövoogude puhul [10]. Platvormil on üle 1000 integratsiooni ning see toetab täielikult JavaScripti ja Pythoni koodi kasutamist töövoogudes, võimaldades väga põhjalikku kohandamist ja keerukate loogikate implementeerimist. n8n on tuntud ka oma võimekuse poolest tehisintellekti agentide ehitamisel ja andmete töötlemisel. Kuigi n8n-i visuaalne liides on kasutajasõbralik, võib selle täielik potentsiaal ja mõnede täpsemate funktsioonide kasutuselevõtt nõuda tehnilisemat arusaama kui näiteks Zapier.

2.4 Eelistatud integratsiooniplatvormide valik

Eelnevalt analüüsitud platvormide võrdluse põhjal, arvestades Syda tarkvara spetsiifilisi vajadusi, klientide tehnilisi oskusi ning hinnastust ja pakutavaid võimalusi, kerkivad esile kaks peamist kandidaati edasiseks prototüüpimiseks: Make ja n8n. Make'i operatsioonipõhine hinnastus ja visuaalselt rikkalik töövoogude loomise keskkond pakuvad head tasakaalu hinna, paindlikkuse ja kasutusmugavuse vahel, sobides paljudele erineva keerukusega integratsioonistsenaariumidele. n8n omakorda paistab silma oma avatud lähtekoodi ja isemajutamise võimalusega, mis võib pakkuda märkimisväärset kulueelist ja täielikku kontrolli andmete üle. Pilveversiooni töövoopõhine hinnastus on atraktiivne keerukamate töövoogude loomise jaoks. Mõlemad platvormid pakuvad piisavalt laia integratsioonide valikut ja tehnilist võimekust, et rahuldada Syda klientide erinevaid vajadusi väliste äriprotsessidega ühendumisel. Lõplik valik nende kahe vahel tehakse praktilise prototüüpimise käigus, hinnates reaalset sobivust Syda rakendusliidesega ning arendus- ja halduskulusid.

2.5 Kavandatava lahenduse eelised ja nõuded

Sellise kahetasandilise lahenduse implementeerimine võimaldaks klientidel integreerida Syda tarkvara andmed sujuvalt teistesse ärirakendustesse või andmebaasidesse, andes neile täieliku kontrolli andmete struktuuri ja vormingu üle. Kliendid saaksid ise valida, millist informatsiooni nad platvormilt küsivad ning millist osa sellest informatsioonist oma edasistes protsessides kasutavad. See pakub peaaegu piiramatuid võimalusi erinevateks

kombinatsioonideks ja lahendusteks ning suudab tõenäoliselt lahendada ühe integratsiooni abil paljude erinevate klientide spetsiifilised andmevajadused.

Selle lahenduse edukaks arendamiseks on vajalikud selgelt määratletud eesmärgid ja funktsionaalsed nõuded. Esialgses analüüsis on peamisteks funktsionaalseteks nõueteks loodavale rakendusliidesele tuvastatud järgmised:

- Rakendusliidese võtme põhine autentimine turvalise ligipääsu tagamiseks.
- Veebihaakide funktsionaalsus, mis võimaldab reaalajas teavituste saatmist sündmuste toimumisel Sydas.
- Andmete päringu otspunktid, mis võimaldavad ligipääsu peamistele andmeobjektidele.

Veebihaakide funktsionaalsuse kaasamine rakendusliidese nõuetesse on ajendatud vajadusest tagada proaktiivne ja sündmuspõhine andmevahetus Syda platvormi ja väliste süsteemide vahel, et kasutajatel oleks võimalik teostada koheselt sündmuste-järgseid operatsioone.

3 Kasutatavad tehnoloogiad

Käesolev peatükk annab ülevaate lõputöö raames valitud ja kasutatud peamistest tehnoloogiatest ning töövahenditest. Järgnevalt käsitletakse sobiva automatiseerimisplatvormi valikuprotsessi, Syda rakendusliidese arendamiseks kasutatud tagarakenduse tehnoloogiaid, kasutajaliidese lahendusi ning arendusprotsessi toetavaid vahendeid. Valikud on tehtud eesmärgiga tagada lahenduse funktsionaalsus, turvalisus ja jätkusuutlikkus.

3.1 Integreeritav automatiseerimisplatvorm

Käesolevas analüüsis keskendutakse kolmele peamisele automatiseerimisplatvormile – Zapier, Make ja n8n – mis valiti välja eelnevas peatükis käsitletud laiema valiku hulgast. Ehkki turul eksisteerib arvukalt teisigi automatiseerimislahendusi, on nimetatud kolmik pälvinud laialdaseima kasutuse ja tunnustuse turuliidritena. Integratsioonilahenduse realiseerimiseks sobivaima platvormi valik tugineb neljale kesksele hindamiskriteeriumile: taskukohasus, kasutusmugavus, integreerimisprotsessi lihtsus ning platvormi funktsionaalsus ja paindlikkus. Nende kriteeriumite eesmärk on tagada lahendus, mis arvestab optimaalselt nii Syda klientide vajadusi kui ka integratsiooni arendusprotsessi efektiivsust ja loodavat väärtust.

Tabel 1. Automatiseerimisplatvormide võrdluse tabel. Hinnastuse aluseks 10000 operatsiooni.

Platvorm	**Hind**	**Kasutuslihtsus**	**Integreerimine**	**Paindlikus**
Zapier	116,66€	Väga lihtne ja intuitiivne	Hästi dokumenteeritud, tasuline	Skriptide võimekus, suurim integratsioonide valik
Make	16,66€	Visuaalne lohistamise redaktor	Hästi dokumenteeritud, tasuta	Tehisintellekti agendid, põhjalik integratsioonide valik
n8n	50€	Tehniline	Koodi põhine, tehniline	Tehisintellekti agendid, isemajutus, skriptide võimekus

Tabelist 1 ilmneb platvormide hinnastuse märkimisväärne varieeruvus. Oluline on seejuures arvestada, et iga platvorm defineerib operatsiooni, mis on sageli hinnastamise aluseks, erinevalt. Make platvormi puhul loetakse operatsiooniks iga mooduli käivitus, sealhulgas andmepäringud nagu pollimine ja trigerid. Zapier rakendab sarnast arvestust, ent ei hõlma trigerite ega andmepäringute käivitamisi operatsioonide hulka. n8n platvormi operatsioonikäsitlus on kõige laiahaardelisem, defineerides operatsioonina ühe töövoo tsükli käivituse, sõltumata selles sisalduvate moodulite arvust. Sellest tulenevalt võib n8n puhul kaaluda ka baaspaketti (2500 operatsiooni kuus hinnaga 20€) kui võrreldavat alternatiivi, mis asetab Make'i ja n8n-i hinnataseme sarnaseks, eriti arvestades, et mahukate töövoogude korral võib n8n osutuda märgatavalt kuluefektiivsemaks.

3.1.1 Valiku põhjendus

Syda tarkvara ja väliste süsteemide vahelise integratsiooni realiseerimiseks valiti Make platvorm. See otsus tugines põhjalikule kaalutlusele lähtuvalt eelnevalt defineeritud hindamiskriteeriumidest: taskukohasus, kasutusmugavus, integreerimisprotsessi lihtsus ning platvormi funktsionaalsus ja paindlikkus. Make'i operatsioonipõhine hinnastusmudel osutus käesolevas kontekstis optimaalseks, pakkudes parimat tasakaalu kulude ja funktsionaalsuse vahel, eriti kui võrrelda Zapieri potentsiaalselt kõrge maksumusega mahukate töövoogude implementeerimisel. Ehkki n8n platvorm võis teatud stsenaariumite, sealhulgas isemajutuse korral, pakkuda sarnast või isegi soodsamat hinda, langetati valik Make'i kasuks tänu selle intuitiivsele visuaalsele töövoogude konfigureerimise keskkonnale ja põhjalikule dokumentatsioonile. Need aspektid muutsid Make'i kasutajasõbralikumaks alternatiiviks, mis tagab kiirema arendusprotsessi ja lihtsama halduse, tegemata kompromisse platvormi funktsionaalsuses või paindlikkuses. Lisaks toetasid Make'i valikut selle ulatuslik toetatud integratsioonide hulk ning selgelt defineeritud võimalused API-põhiste ühenduste loomiseks, mis on Syda integratsioonilahenduse seisukohast kriitilise tähtsusega.

3.2 Tagarakenduse rakendusliides

Syda tarkvaralahenduse tagarakendus on realiseeritud Java programmeerimiskeeles, tuginedes Spring Boot raamistikule. Platvormil on kasutusel kaks eraldiseisvat tagarakendust, ent

käesoleva lõputöö fookuses on neist uuem, mis on määratud edasiste arenduste, sealhulgas käesolevas töös loodava integratsioonilahenduse implementeerimise keskkonnaks. Selles tagarakenduses on kasutusel Java versioon 17 ning Spring Boot raamistiku versioon 3.4.3. Nimetatud tehnoloogiate valik on põhjendatud meeskonna varasema põhjaliku kogemusega, mis tagab arendusprotsessi sujuvuse ja efektiivsuse. Lõputöö raames ei looda uut tagarakendust nullist, vaid täiustatakse ja laiendatakse olemasolevat rakendusliidest, et võimaldada vajalik andmevahetus väliste süsteemidega vastavalt püstitatud eesmärkidele. See hõlmab olemasoleva API edasiarendamist, et see suudaks pakkuda vajalikke otspunkte ja funktsionaalsust loodavale integratsioonile.

3.3 Kasutajaliides

Käesoleva lõputöö kontekstis on kasutajaliidese arendusel sekundaarne roll, kuna peamine fookus on suunatud tagarakenduse rakendusliidese arendamisele ja integratsioonimehhanismide loomisele, mis võimaldavad Syda andmete sünkroniseerimist väliste äriprotsessidega. Minimaalsete ja tööks hädavajalike kasutajaliidese komponentide loomiseks, näiteks integratsiooni haldamiseks või konfigureerimiseks Syda platvormi sees, kasutatakse Angular raamistiku versiooni 11. See tagab ühilduvuse olemasoleva kasutajaliidese tehnoloogiapinuga ning võimaldab vajadusel uusi funktsionaalsusi sujuvalt integreerida. Arendatavad kasutajaliidese elemendid piirduvad vaid nende osadega, mis on otseselt seotud loodava integratsioonilahenduse administreerimise või monitoorimisega.

3.4 Koodihoidla

Tarkvaraarenduse projektifailide, sealhulgas käesoleva lõputöö raames loodava lähtekoodi keskseks hoidlaks on Atlassian Bitbucket [11]. Selles keskkonnas hallatakse kogu Syda tarkvaralahenduse koodibaasi, tagades versioonikontrolli ja meeskonnatöö võimalused. Koodihoidlasse on integreeritud automatiseeritud tarkvaraarenduse pipeline'id, mis võimaldavad uute rakenduse versioonide kontrollitud ja automaatset reliisimist. See protsess hõlmab esmalt versioonide paigaldamist testkeskkonda ning pärast valideerimist liigutamist produktsioonikeskkonda. Lisaks on peamiste repositooriumitega integreeritud SonarQube koodianalüüsi tööriist. SonarQube aitab tagada arendatava koodi, sealhulgas integratsiooniga seotud komponentide kvaliteeti, tuvastades potentsiaalseid probleeme, turvanõrkusi ja

koodivigu, ning on seega oluline osa kvaliteetse ja jätkusuutliku lahenduse loomisel [12].

4 Integratsiooni arendus

Antud lõputöö praktilise osa keskmes on Syda tarkvarale loodava integratsioonivõimekuse tehniline teostus. See peatükk kirjeldab üksikasjalikult nimetatud API komponentide – otspunktide, autentimismehhanismi ja veebihaakide süsteemi - arendamist alates funktsionaalsetest nõuetest kuni tehnilise implementatsiooni ja kvaliteedi tagamiseni, tuginedes eelnevates peatükkides esitatud analüüsile ja tehnoloogiavalikutele.

4.1 Rakendusliidese laiendamine

Antud alapeatükis on välja toodud Syda tarkvara rakendusliidese laiendamise protsessi etapid, alustades funktsionaalsetest nõuetest ja lõpetades uute otspunktide tehnilise implementatsiooniga. Eesmärgiks oli luua API, mis on nii funktsionaalne kui ka turvaline ning mugav kasutada välistel integratsioonidel.

4.1.1 Funktsionaalsed nõuded

Rakendusliidese funktsionaalsed nõuded kujunesid välja tihedas koostöös Syda tarkvara tooteomanikuga, lähtudes platvormi olemasolevatest põhiandmetest, äriloogikast ning klientidelt saadud tagasisidest integratsioonivajaduste kohta. Põhiliseks eesmärgiks seati võimaldada kasutajatel läbi rakendusliidese teostada kõiki peamisi andmeoperatsioone Syda platvormi kesksete andmeobjektidega. See tähendab, et platvormi tuumikobjektid – eelkõige tööd ja töö raportid – pidid muutuma hallatavaks rakendusliidese kaudu. Lisaks tuli tagada ligipääs ja haldusvõimekus nendega seotud olulistele andmeobjektidele, nagu objektid (tööde teostamise asukohad), kliendid, töötajad, töö tüübid, töödega seotud kontrollnimekirjad ning kontaktisikud.

Kuigi Syda rakenduses eksisteerisid sisemised otspunktid paljude nimetatud andmete pärimiseks, ei sobinud nende olemasolev implementatsioon avaliku rakendusliidese eesmärkidega. Need olid peamiselt mõeldud platvormi enda kasutajaliidese toetamiseks ning nende andmevastused sisaldasid sageli spetsiifilist lisainformatsiooni ja struktuure, mis olid

relevantsed vaid sisemise äriloogika jaoks. Avaliku rakendusliidese otspunktide andmevastusobjektid disainiti aga põhimõttel, et need edastaksid ainult seda informatsiooni, mis on integratsiooni loovale kliendile tegelikult vajalik või kasulik. Selline lähenemine tagab esiteks mugavama kasutuskogemuse integratsiooni arendajatele ning teiseks optimeerib andmeedastusmahtusid, vähendades tarbetut müra ja koormust.

Funktsionaalsed nõuded ei piirdunud ainult andmete lugemisega. Mitmete andmeobjektide, näiteks tööde ja kontaktide puhul, oli vajalik implementeerida ka loomise funktsionaalsus. Samuti oli tarvis uuendamise meetodeid tööde, töö raportite ja kontaktide andmete muutmiseks. Kustutamise operatsioone esialgses skoobis ei implementeeritud, et vältida andmete juhuslikku kadu ning jätta see võimalus tulevikuarendusteks rangelt kontrollitud tingimustel.

Tabel 2. Rakendusliidese andmeobjektide ja CRUD operatsioonide ülevaade.

Andmeobjekt	**Create**	**Read**	**Update**	**Delete**
Töö	✓	✓	✓	-
Töö raport	-	✓	✓	-
Kontakt	✓	✓	✓	-
Töötaja	-	✓	-	-
Töö tüüp	-	✓	-	-
Kontrollnimekiri	-	✓	-	-
Objekt	-	✓	-	-
Klient	-	✓	-	-

4.1.2 Otspunktide implementatsioon

Pärast funktsionaalsete nõuete detailset dokumenteerimist ning nende kinnitamist ettevõtte tooteomanikuga, alustati uute rakendusliidese otspunktide tehnilise implementatsiooniga Syda Java ja Spring Boot põhises tagarakenduses. Arendusprotsess järgis iteratiivset lähenemist, alustades kõige kriitilisemate andmeobjektide, nagu tööde haldamise funktsionaalsuse, realiseerimisest.

Kõik uued API otspunktid realiseeriti REST arhitektuuristiili põhimõtteid järgides, kasutades standardseid HTTP meetodeid ning JSON formaati andmevahetuseks. Spring Boot raamistiku võimalusi, nagu @RestController, @Service ja @Repository annotatsioonid, kasutati koodi struktureerimiseks vastavalt tunnustatud n-kihilisele arhitektuurimustrile.

Iga andmeobjekti jaoks loodi spetsiifilised andmeedastusobjektid, mis defineerisid täpselt, millised andmed API kaudu liiguvad, tagades sellega andmete selgepiirilisuse ja vältides sisemiste andmemudelite paljastamist. Näiteks tööde otspunktide arendamisel loodi JobApiResponse, mis sisaldas vaid integratsioonide jaoks relevantset infot.

Päringute valideerimine implementeeriti nii andmeedastusobjektide tasemel, kasutades Java Bean Validation annotatsioone, nagu @NotNull või @Size, kui ka äriloogika kihis, et tagada andmete korrektsus enne nende salvestamist andmebaasi. Spring Security raamistikku kasutati uue API võtmete põhise autentimismehhanismi integreerimiseks, mida kirjeldatakse detailsemalt järgmises peatükis. API versioonihaldust esialgses etapis ei implementeeritud, kuid otspunktide URL-struktuur kujundati viisil */api/v1/*, mis võimaldab tulevikus mugavalt uute versioonide lisamist ilma olemasolevaid integratsioone lõhkumata. Arenduse käigus pöörati tähelepanu ka päringute optimeerimisele, et tagada API kiire ja efektiivne toimimine, eriti suuremate andmemahtude korral.

4.2 Turvalisuse tagamine

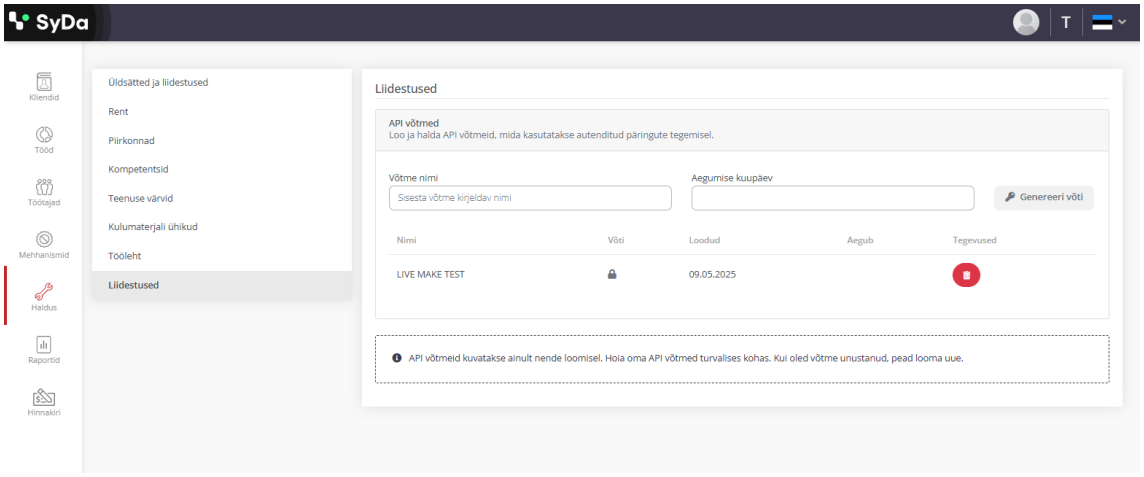
Loodava rakendusliidese puhul tekkis selge vajadus uue, spetsiaalselt välistele integratsioonidele mõeldud autentimismeetodi järele, kuna platvormi olemasolev JWT põhine autentimine ei pakkunud piisavat paindlikkust ega haldusvõimekust API klientide jaoks. Peamiseks probleemiks osutus asjaolu, et JWT-põhise autentimise korral oleks ligipääsu tühistamine konkreetsele integratsioonile osutunud keerukaks. Aktiivse ja valiidse JWT ligipääsu eemaldamiseks oleks vaja olnud roteerida JWT allkirjastamise privaatset võtit terves rakenduses, mis oleks invalideerinud kõik olemasolevad kasutajate JWT'id ning tekitanud seeläbi laialdast ebamugavust või segadust. Seetõttu otsustati implementeerida API võtmete põhine autentimissüsteem, mis võimaldab igale integratsioonikliendile genereerida unikaalse võtme ning vajadusel selle ligipääsu individuaalselt hallata ja tühistada.

Selle lahenduse väljatöötamine nõudis mitmeid kaalutletud otsuseid. Lihtne lahendus, kus API võti oleks alati selgel kujul kättesaadav ja andmebaasis krüpteerimata hoitud, vähendaks oluliselt rakenduse turvalisust, eriti arvestades potentsiaalset ohtu, et kasutajad võivad võtmeid ebaturvaliselt jagada või et andmebaasi kompromiteerimisel satuksid võtmed kolmandate osapoolte kätte. Selle probleemi esmaseks kaalutud lahenduseks oli võtmepaari - identifikaator ja saladus, sarnaselt kasutajanime ja parooliga - kasutuselevõtt.

See aga oleks tekitanud kliendile vajaduse hallata kahte eraldiseisvat mandaati ning nende edastamine autentimispäringus poleks olnud kõige intuitiivsem.

Lõplikuks lahenduseks valiti ühekordselt nähtava API võtme genereerimise põhimõte. Süsteem genereerib uue võtme loomisel selle identifitseeriva osa ning saladuse osa. Need kaks komponenti kombineeritakse ja edastatakse Base64 kodeeringus kliendile ühtse võtmena. Klient peab selle võtme koheselt turvaliselt salvestama, kuna seda ei kuvata talle enam kunagi uuesti. Andmebaasi salvestatakse võtme identifitseeriv osa ning saladuse osa turvaline räsiväärtus, kasutades selleks tugevat räsialgoritmi Bcrypt [13]. Selline lähenemine lahendab kaks peamist probleemi: esiteks on kliendil hallata vaid ühte API võtit ning teiseks on võtme salajane osa andmebaasis kaitstud ka lekke korral.

API võtmete haldamiseks oli vaja luua tarkvara haldusmoodulisse kasutajaliides, kust oleks võimalik hallata ettevõtte võtmeid. Selle jaoks loodi uus alamkategooria *Liidestused* rakenduse üldsätete kategooriasse.



Joonis 1. API võtmete haldamiseks loodud kasutajaliides.

Autentimisprotsess ise implementeeriti Spring Security raamistiku abil loodud kohandatud filtrina. See filter kontrollib igat sissetulevat päringut, mis on suunatud integratsiooni rakendusliidese otspunktidele, täpsemalt */api/*. Filter otsib päringu päisest, milleks määrati *syda-api-key*, edastatud API võtit, eraldab sellest identifitseeriva osa ja kliendi poolt edastatud salajase osa ning seejärel võrdleb saladust andmebaasis hoitava räsi vastu. Eduka autentimise korral lubatakse päring edasi rakendusliidese vastavasse kontrollerisse.

4.3 Veebihaakide implementeerimine

Lisaks traditsioonilistele rakendusliidese otspunktidele, kus klient peab andmete saamiseks ise aktiivselt päringuid tegema, tekkis äriline vajadus reaalajas teavituste järele teatud sündmuste toimumisel Syda platvormil. Selleks otsustati implementeerida veebihaakide süsteem, esialgses etapis keskendudes platvormi peamisele andmeobjektile – töödele. Veebihaagid võimaldavad klientrakendustel registreerida URL-i, kuhu Syda saadab automaatselt HTTP POST päringu koos relevantse infoga kohe, kui defineeritud sündmus toimub, näiteks kui töö luuakse, selle andmeid muudetakse, alustatakse, kustutatakse või lõpetatakse.

Veebihaakide funktsionaalsuse implementeerimiseks loodi esmalt uus andmebaasitabel veebihaakide registreeringute hoidmiseks. Selles tabelis säilitatakse iga registreeritud veebihaagi kohta selle unikaalne identifikaator, kliendi poolt määratud sihtkoha URL, sündmuse tüüp, mille kohta teavitust soovitakse, ning seos ettevõttega, kellele veebihaak kuulub. Rakendusliidesesse lisati uued otspunktid veebihaakide haldamiseks:

- `POST`: Uue veebihaagi registreerimiseks, kus päringu kehas edastatakse sihtkoha URL ja sündmuse tüüp.
- `GET`: Kasutajaga seotud aktiivsete veebihaakide nimekirja pärimiseks.
- `DELETE`: Konkreetse veebihaagi eemaldamiseks selle identifikaatori alusel.

Kõik veebihaakide haldamise otspunktid on kaitstud sama API võtmete põhise autentimismehhanismiga, mis loodi ülejäänud rakendusliidese jaoks.

Sündmuste käsitlemise loogika integreeriti Syda tagarakenduse äriloogikasse. Töödega seotud oluliste operatsioonide järel luuakse sündmus Spring ApplicationEventPublisher liidese abil. Nende sündmuste käsitlemiseks olid loodud kuulaja meetodid, annoteeritud @EventListener annotatsiooniga. See võimaldas loogiliselt hallata sündmuste käsitlust vastavalt sündmuse tüübile. Seejärel otsitakse andmebaasist kõik antud sündmuse tüübiga seotud aktiivsed veebihaakide registreeringud ning saadetakse igaühele neist asünkroonselt HTTP POST päring. Lisaks sündmuse tüübi filtreerimisele rakendub ka üldine ettevõtte filter, mis tagab, et töö teavitus saadetakse vaid vastava ettevõtte registreeritud URL-idele. Päringu keha sisaldab JSON formaadis informatsiooni terve töö kohta. Veebihaakide saatmisel implementeeriti logimine, et märgata potentsiaalseid tõrkeid registreerunud

veebihaakide teavitamisel ning vajadusel neid analüüsida.

4.4 Tugitegevused ja kvaliteedi tagamine

Rakendusliidese arendusprotsessi vältel rakendati mitmeid praktikaid, et tagada koodi kõrge kvaliteet, töökindlus ja jätkusuutlikkus. Kesksteks aspektideks olid pidev koodianalüüs, automatiseeritud testimine ning meeskonnasisene koostöö.

- **Versioonihaldus:** Kogu Syda rakendusliidese laienduste lähtekood hallati Atlassian Bitbucket versioonihaldussüsteemis, kasutades Git'i. See võimaldas jälgida muudatusi, teha koostööd ning vajadusel taastada varasemaid versioone.
- **Koodikvaliteedi analüüs:** Projekti Java koodibaasi jaoks oli seadistatud SonarQube staatilise koodianalüüsi tööriist. Iga uue koodimuudatuse puhul analüüsis SonarQube koodi, tuvastades potentsiaalseid vigu, turvanõrkusi ja stiiliprobleeme. See aitas hoida koodi kvaliteeti ühtsena ning järgida parimaid praktikaid.
- **Automatiseeritud testimine:** Rakendusliidese arendamisel pandi suurt rõhku integratsioonitestide loomisele. Kõikide uute API otspunktide ja olulisemate äriloogika komponentide jaoks kirjutati Spring Boot testraamistikku kasutades integratsioonitestid. Need testid kontrollisid otspunktide korrektset toimimist, sealhulgas päringute valideerimist, autentimist, andmete töötlemist ja oodatud vastuste genereerimist. Testid käivitati automaatselt iga koodimuudatuse järel pideva integratsiooni protsessi osana, mis aitas tagada, et uued muudatused ei lõhu olemasolevat funktsionaalsust ning vastavad kvaliteedinõuetele enne koodi liitmist edasistesse arenduskeskkondadesse.
- **Koodi ülevaatused:** Enne uue koodi liitmist peamisesse arendusharusse vaatas selle üle ja kinnitas teine arendaja. Koodiülevaatuste eesmärk oli tuvastada võimalikke loogikavigu, parandada koodi loetavust, jagada teadmisi ning tagada ühtse koodistiili järgimise.

Need tegevused aitasid minimeerida vigu, parandada lahenduse üldist robustsust ning tagada, et loodud rakendusliides on usaldusväärne ja hallatav.

4.5 Arendustöö väljakutsed ja lahendused

Rakendusliidese arendusprotsess sisaldas mitmeid tehnilisi väljakutseid, mis nõudsid hoolikat analüüsi ja läbimõeldud lahendusi. Järgnevalt kirjeldatakse peamisi esile kerkinud probleeme ja nende lahenduskäike.

Esimene märkimisväärne väljakutse oli seotud API võtmete põhise autentimismehhanismi kujunduse ja implementeerimisega. Nagu eelnevalt mainitud, oli vaja leida tasakaal turvalisuse ja kasutusmugavuse vahel. Mitmete iteratsioonide käigus kaaluti erinevaid lähenemisi alates lihtsatest staatilistest võtmetest kuni keerukamate võtmepaarideni. Lõplikuks valitud ühekordselt nähtava, räsitud saladusega API võtme lahendus nõudis hoolikat andmebaasi struktuuri planeerimist ning turvalise genereerimis- ja valideerimisloogika väljatöötamist. Otsustuspunktid hõlmasid võtme struktuuri defineerimist ning kliendile edastatava formaadi valikut.

Teine oluline tehniline ülesanne oli autentimisfiltri korrektne implementeerimine Spring Security raamistikus nii, et see ei satuks konflikti Syda rakenduse olemasoleva JWT-põhise autentimismehhanismiga, mida kasutatakse platvormi kasutajaliidese sessioonide haldamiseks. Väljakutse seisnes selles, et rakenduses pidi paralleelselt toimima kaks erinevat autentimisstrateegiat erinevate otspunktide gruppide jaoks. See lahendati Spring Security konfiguratsioonis filtrite järjestuse täpse määramisega ning uue API võtme filtri rakendamisega ainult spetsiifilistele URL-mustritele (/api/**), jättes muud rakenduse osad senise autentimisloogika alla. See tagas, et uus API autentimine ei mõjutaks olemasolevate kasutajate tööd.

Kolmandaks väljakutseks võib pidada veebihaakide süsteemi asünkroonset ja töökindlat implementeerimist. Oli oluline tagada, et veebihaakide saatmine ei blokeeriks põhiprotsesse. See lahendati asünkroonsete meetodite kujul, mis võimaldas viia teavitamise loogika eraldi transaktsiooni, mis ei hoidnud sündmust tekitanud operatsiooni kinni. Samuti implementeeriti logimine veebihaagi teavitamise kohta, et võimaldada hilisemat analüüsi tõrgete korral.

Nende ja teiste väiksemate väljakutsete lahendamine nõudis põhjalikku süsteemi tundmist, tehnoloogiate valdamist ning kohati ka loomingulist lähenemist, et saavutada püstitatud eesmärgid nii funktsionaalsuse kui ka turvalisuse osas.

5 Automatsioonirakenduse loomine

Eelmises peatükis kirjeldatud Syda platvormi rakendusliides, sealhulgas selle autentimismehhanism ja veebihaakide võimekus, loob aluse välistele integratsioonidele. Käesoleva peatüki eesmärk on kirjeldada selle API tarbeks loodud kohandatud automatsioonirakenduse arendusprotsessi Make platvormil. Sellise ühenduse loomine on oluline samm Syda integreerimisel väliste äriprotsessidega, kuna see lihtsustab oluliselt lõppkasutajatel automatiseeritud töövoogude koostamist, kapseldades Syda API spetsiifilised tehnilised detailid ning pakkudes Make keskkonnas intuitiivset kasutajaliidest Syda andmetega opereerimiseks. Peatükk katab Make'i rakenduse loomise põhiseaded, ühenduse konfigureerimise, autentimise implementeerimise, API otspunktidele vastavate moodulite loomise ning veebihaakide toe realiseerimise.

5.1 Make platvormi rakenduse arenduskeskkond

Make on laialdaselt kasutatav integratsiooniplatvorm teenusena, mis võimaldab erinevate rakenduste ja teenuste vahel andmeid sünkroniseerida ja töövooge automatiseerida ilma ulatusliku programmeerimiseta. Kuigi Make pakub universaalseid HTTP mooduleid suvaliste rakendusliidestega suhtlemiseks, võimaldab platvormi arenduskeskkond luua ka kohandatud rakendusi. Kohandatud rakenduse loomine Syda jaoks valiti seetõttu, et see pakub mitmeid eeliseid:

- **Kasutusmugavus:** Lõppkasutajad saavad Syda funktsionaalsust kasutada Make'i stsenaariumites spetsiaalsete, selgelt nimetatud moodulite kaudu, ilma et peaksid ise HTTP päringuid konfigureerima.
- **Abstraktsioon:** API tehnilised detailid, nagu päiste haldus ja spetsiifilised autentimisnõuded, on peidetud rakenduse sisse.
- **Järjepidevus ja hooldatavus:** Keskne rakendus tagab, et kõik Syda integratsioonid kasutavad API-t ühtsel ja korrektsel viisil. API muutuste korral tuleb uuendada vaid keskset rakendust, mitte iga individuaalset stsenaariumit.

Make'i rakenduse loomine hõlmab rakenduse defineerimist, ühenduse tüübi määramist, autentimismeetodite seadistamist ning erinevate moodulite, nagu trigerid, tegevused ja otsingud, implementeerimist JSON-põhiste konfiguratsioonide kaudu.

5.2 Ühenduse konfigureerimine ja autentimine

Make'i rakenduse esmane samm on ühenduse defineerimine, mis määrab, kuidas Make suhtleb Syda API-ga. Syda jaoks loodud Make'i rakenduses konfigureeriti ühendus kasutama eelnevalt arendatud API-võtme põhista autentimismehhanismi. Make'i rakenduse ühenduse seadistamisel defineeriti järgmised aspektid:

- **Autentimise tüüp:** Valiti API-võtme tüüpi autentimine. See võimaldab Make'i kasutajal, kes soovib Syda rakendust oma stsenaariumis kasutada, sisestada oma unikaalse Syda API-võtme turvaliselt Make'i ühenduse loomisel.
- **API-võtme edastamine:** Konfigureeriti, et API-võti lisatakse igale Syda API-le tehtavale päringule HTTP päisesse, näiteks `SYDA-API-KEY` päisena, vastavalt Syda API implementatsioonile.
- **Baas-URL:** Määrati Syda API keskne baas-URL, näiteks `https://jobs.syda.io/api`, millele kõik moodulite poolt tehtavate päringute teekonnad liidetakse.

See seadistus tagab, et kõik Make'i rakenduse kaudu tehtavad kutsed Syda API-le on korrektselt vormistatud ning autenditud, kasutades individuaalseid ja turvaliselt hallatavaid API-võtmeid.

5.3 Moodulite arendamine

Moodulid on Make'i rakenduse funktsionaalsed plokid, mida kasutajad saavad oma automatiseerimisstsenaariumitesse lisada. Syda Make'i rakenduse jaoks loodi moodulid, mis vastavad peamistele Syda API otspunktidele ja operatsioonidele, nagu on kirjeldatud eelmises peatükis, vaata tabel 2.

5.3.1 Tegevusmoodulid ja Otsingumoodulid

Loodi mitmeid tegevus- ja otsingumooduleid, mis võimaldavad andmeid lugeda, luua ja uuendada. Iga sellise mooduli arendamisel defineeriti Make'i arenduskeskkonnas selle

parameetrid, API kutsungi loogika, sealhulgas URL, meetod ja päringu keha koostamine sisendparameetritest, ning vastuse struktuur ehk kuidas API vastusest saadud JSON andmed kaardistatakse mooduli väljundväljadeks, mida saab kasutada stsenaariumi järgmistes sammudes.

5.3.2 Veebihaagi triger

Et võimaldada reaalajas sündmuspõhist integratsiooni, loodi Syda Make'i rakendusele veebihaagi triger-moodul. See moodul toimib järgmiselt:

- **Veebihaagi registreerimine Make'is:** Mooduli seadistamisel genereerib Make unikaalse URL-i, millele Syda saab teavitusi saata.
- **Struktuuri defineerimine:** Määrati oodatava veebihaagi JSON-andmestruktuur, et Make saaks sissetulevaid andmeid korrektselt parsida. See struktuur peab vastama Syda veebihaakide poolt saadetavale andmeformaadile.
- **Kasutamine stsenaariumis:** Kasutaja saab lisada selle trigeri oma Make'i stsenaariumi algusesse. See Make'i poolt genereeritud URL tuleb registreerida Syda platvormil vastava sündmuse, näiteks "Töö uuendatud", veebihaagina, kasutades Syda API veebihaakide haldamise otspunkte. See käsitletakse Make'is automaatselt.

Kui Sydas toimub vastav sündmus, saadab Syda POST päringu registreeritud Make'i URL-ile. Make'i rakenduse veebihaagi moodul võtab need andmed vastu ja käivitab seotud stsenaariumi, edastades sündmuse andmed stsenaariumi järgmistele moodulitele töötlemiseks.

5.4 Rakenduse testimine

Pärast Syda Make'i rakenduse ühenduse, autentimise ja peamiste moodulite esialgset implementeerimist viidi läbi testimine. See hõlmas individuaalsete moodulite funktsionaalsuse kontrollimist, näiteks kas töö andmete pärimine tagastab oodatud tulemused või kas uue kontakti loomine õnnestub. Samuti loodi terviklikke test-stsenaariumeid Make'is, et valideerida moodulite omavahelist koostööd ja andmevoogude korrektsust. Testimise käigus kontrolliti ka veatöötlust ja API-võtme autentimise korrektset toimimist erinevates situatsioonides. Leitud puudused parandati iteratiivselt.

6 Tulemuste analüüs

Antud peatükis on dokumenteeritud lõputöö arenduse käigus valminud lahendus ning selle valideerimine Syda rakenduse klientidega. Seejärel on tagasiside ning tulemuste põhine analüüs ning ideed edasiste arenduste jaoks seoses lõputöös valminud lahendusega.

6.1 Tulemused

Selles alapeatükis kirjeldatakse detailselt lõputöö käigus valminud lahenduse peamisi komponente ja nende loodud väärtust nii Syda platvormile kui ka selle klientidele. Eesmärk on anda konkreetne ülevaade praktilise töö käegakatsutavatest väljunditest.

6.1.1 Valminud lahenduse komponentide ülevaade

Lõputöö praktilise osana valmis mitmekihiline integratsioonilahendus Syda tarkvarale. Selle keskmes on kaks peamist komponenti. Esiteks arendati Syda platvormi jaoks uus rakendusliides. See REST rakendusliides baseerub API võtmete põhisel autentimisel ning sisaldab olulisi otspunkte platvormi andmeobjektide, nagu tööde ja töö raportite, haldamiseks. Lisaks implementeeriti veebihaakide süsteem, mis võimaldab reaalajas teavitusi saada näiteks tööde olekumuutuste korral. Terve rakendusliidese raames sai loodud 21 uut otspunkti. Nimetatud rakendusliidese jaoks implementeeriti ka avalik Swagger dokumentatsioon, mis pakub vajaliku info API otspunktide kohta, et mugavalt integratsioone luua [14].

Teiseks loodi Make'i platvormile kohandatud rakendus. See Make'i rakendus lihtsustab oluliselt Syda API kasutamist automatiseeritud töövoogudes, pakkudes eeldefineeritud mooduleid API otspunktidega suhtlemiseks ning veebihaakide vastuvõtmiseks. Märkimisväärne tulemus selle komponendi arendamisel on Syda rakenduse edukas avaldamine Make'i platvormi avalikus integratsioonide kataloogis [15]. Need kaks komponenti koos moodustavad tervikliku ja funktsionaalse lahenduse Syda integreerimiseks väliste süsteemidega, võimaldades paindlikku andmevahetust ja protsesside automatiseerimist.

6.1.2 Loodud lahenduse väärtus Syda platvormile ja selle klientidele

Valminud integratsioonilahendus pakub märkimisväärset väärtust nii Syda platvormi haldajatele kui ka selle lõppkasutajatele. Syda platvormi jaoks tähendab see eelkõige tehnoloogilise võimekuse ja konkurentsivõime kasvu. Kaasaegne ja hästi dokumenteeritud API ning tugi laialt levinud automatiseerimisplatvormile nagu Make, eriti nüüd, kui Syda rakendus on Make'is avalikult kättesaadav, avab uksed uutele kliendisegmentidele ja vähendab vajadust kulukate erilahenduste järele. Samuti loob see tugeva aluse tulevasteks API-põhisteks arendusteks ja platvormi funktsionaalsuse laiendamiseks.

Syda klientide jaoks on peamine väärtus suurem paindlikkus ja kontroll oma andmete üle. Nagu analüüsifaasis tuvastatud, võimaldab lahendus klientidel ise valida, millist informatsiooni nad platvormilt küsivad ning kuidas seda oma äriprotsessides kasutavad. See lahendab varasema probleemi seoses jäikade ekspordivormingutega. Võimalus automatiseerida andmesisestust ja sünkroniseerimist väliste süsteemidega vähendab oluliselt käsitsitööd, minimeerib inimlikke vigu ning tõstab töövoogude üldist efektiivsust. Syda rakenduse avalik kättesaadavus Make'i platvormil lihtsustab veelgi nende integratsioonide loomist. Lõppkokkuvõttes saavad ettevõtted luua just enda spetsiifilistele vajadustele vastavaid integratsioone, optimeerides ressursikasutust ja parandades otsustusprotsesse.

6.2 Lahenduse valideerimine

Valminud lahenduse toimivuse ja ärilise väärtuse hindamiseks viidi läbi praktiline valideerimine. Selles jaotises kirjeldatakse valideerimismetoodikat ning konkreetset stsenaariumit, mida kasutati lahenduse testimiseks, ning esitatakse oluline tulemus seoses lahenduse avalikustamisega.

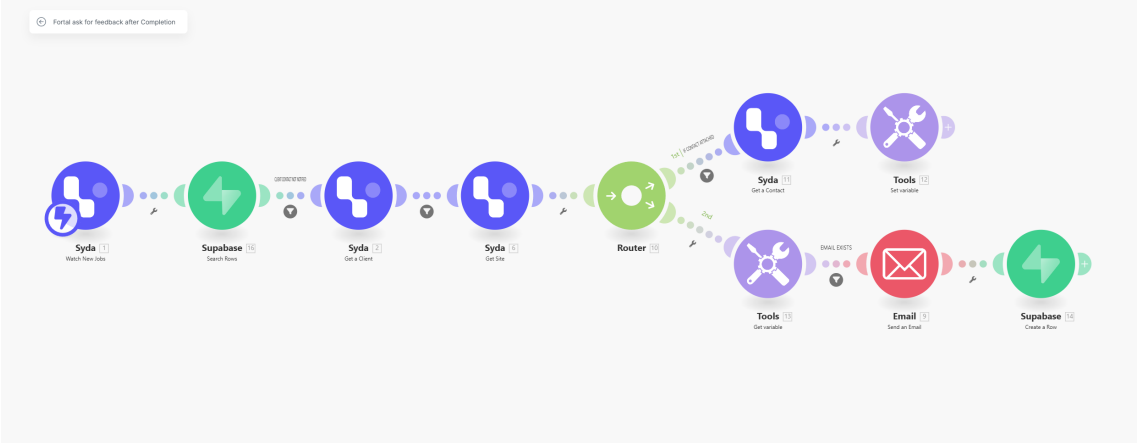
6.2.1 Valideerimismeetodi kirjeldus

Lahenduse valideerimiseks valiti mitmetahuline lähenemine. Esiteks keskenduti praktilise kasutusjuhtumi realiseerimisele Make platvormil, et demonstreerida tehnilist toimivust ja potentsiaalset ärilist kasu. Teiseks oluliseks valideerimise osaks oli Syda jaoks loodud Make'i rakenduse esitamine Make'i platvormi meeskonnale ülevaatuseks eesmärgiga see avalikult kättesaadavaks teha. Valideerimisprotsessi kaasati Syda tooteomanik ning simulatsiooni käigus testiti stsenaariumi sisemiste testkasutajatega. Hinnati lahenduse

võimekust lahendada konkreetne ärivajadus, API vastuste korrektsust ning Make'i rakenduse intuitiivsust ja vastavust platvormi standarditele.

6.2.2 Praktiline valideerimisstsenaarium Make platvormil

Valideerimiseks loodi Make'is automatiseeritud töövoog, mille eesmärk oli kliendisuhtluse parandamine ja tagasiside kogumine pärast töö lõpetamist Sydas. Stsenaarium käivitus Syda platvormilt saabuva veebihaagi peale, mis teavitas töö märkimisest lõpetatuks. Seejärel kasutas Make'i stsenaarium loodud kohandatud rakenduse mooduleid, et pärida Syda rakendusliidesest vastava töö detailid ning töö tellinud kliendi kontaktandmed. Saadud informatsiooni põhjal koostas stsenaarium personaliseeritud e-kirja, mis tänas klienti koostöö eest ning sisaldas linki tagasisideküsitlusele. Küsitluse eesmärk oli koguda sisendit teenuse kvaliteedi edasiseks parandamiseks. Lõpetuseks saatis Make'i stsenaarium koostatud e-kirja kliendi kontaktisikule. See stsenaarium demonstreeris edukalt nii Syda API veebihaakide ja otspunktide funktsionaalsust kui ka Make'i rakenduse võimekust neid komponente orkestreerida.



Joonis 2. Praktilise valideerimise jaoks loodud Make stsenaarium.

6.2.3 Valideerimise tulemused

Valideerimisstsenaariumi tehniline teostus Make platvormil õnnestus ootuspäraselt. Süsteem käivitus korrektselt Syda veebihaagi saamisel, andmepäringud Syda rakendusliidesele läbi Make'i rakenduse moodulite toimisid tõrgeteta ning e-kiri koos dünaamilise sisuga saadeti edukalt välja. See kinnitas loodud integratsioonikomponentide tehnilist valmidust.

Üks selle töö kõige kaalukamaid valideerimistulemusi on Syda jaoks loodud Make'i

rakenduse edukas avalikustamine Make'i platvormi ametlikus integratsioonide kataloogis. Valideerimisprotsessi algfaasis esitati rakendus Make'i meeskonnale ülevaatuseks. Positiivne vastus ja rakenduse lisamine avalike integratsioonide hulka saabus 12. mail 2025. aastal. Protsess oli kiire ning nõudis vaid ühte täiendusvooru, mis keskendus peamiselt moodulite nimetuste ja kirjelduste selguse parandamisele ning standardiseerimisele vastavalt Make'i platvormi nõuetele. See tulemus ei kinnita mitte ainult loodud Make'i rakenduse tehnilist kvaliteeti ja vastavust tööstusstandarditele, vaid annab ka sõltumatu kinnituse selle kasutuskõlblikkusele laiemale publikule.

6.3 Tulemuste analüüs ja edasiarendusettepanekud

Käesolev jaotis keskendub töö käigus saavutatud tulemuste kriitilisele hindamisele, püstitatud eesmärkide täitmise analüüsile ning võimalikele edasiarendussuundadele.

6.3.1 Saavutatud eesmärkide hindamine

Lõputöö peamiseks eesmärgiks oli luua toimiv integratsioonilahendus Syda rakenduse ja väliste äriprotsesside vahel, mis võimaldaks automatiseeritud andmevahetust. Arendatud Syda API ning Make'i kohandatud rakendus, mis on nüüdseks ka Make platvormil avalikult kättesaadav, täidavad selle eesmärgi, pakkudes tehnilised vahendid andmete liigutamiseks ja töövoogude automatiseerimiseks. Töö tulemusena väheneb vajadus käsitsi andmesisestuseks, mis oli üks peamisi probleemipüstitusi, ning sellega seotud vigade oht. Loodud lahendus parandab ettevõtete töövoogude efektiivsust, muutes andmete kasutamise paindlikumaks ja kiiremaks, mis vastab lõputöö sissejuhatuses seatud oodatavatele tulemustele. Syda rakenduse edukas avaldamine Make'i platvormil on täiendav kinnitus lahenduse kvaliteedile ja eesmärkide saavutamisele. Võib öelda, et püstitatud põhieesmärgid on valminud lahenduse näol saavutatud.

6.3.2 Kogutud tagasiside analüüs

Pärast valideerimisfaasi koguti tagasisidet lahenduse kohta nii Syda tooteomanikult kui ka valideerimisstsenaariumis osalenud kliendilt. Kuigi tagasisideandjate ring jäi piiratuks, andsid saadud kommentaarid sisukat teavet lahenduse praktilise väärtuse ja võimalike parenduskohtade kohta.

Tooteomaniku üldine hinnang loodud lahendusele oli väga positiivne, mis kinnitab arendustöö kontseptuaalset õnnestumist ja loodud integratsioonivõimekuse olulisust platvormi jaoks. Eraldi toodi välja heakskiit lahenduse üldisele funktsionaalsusele ja selle potentsiaalile Syda tarkvara laiendamisel. Valideerimisstsenaariumis osalenud klient tõstis esile asjaolu, et loodud integratsioonilahenduse haldamine ja korrigeerimine on täielikult nende endi kätes. Seejuures rõhutati positiivsena võimalust teha muudatusi iseseisvalt, ilma vajaduseta oodata arendustsükleid. Selline autonoomia vastab otseselt töö üheks eesmärgiks olnud paindlikkuse ja kontrolli suurendamisele kliendi jaoks oma äriprotsesside üle.

Tooteomanik tõi konstruktiivse märkuse ja edasiarendusettepanekuna esile piirangu seoses tööde loomisega. Nimelt eeldab praegune lahendus uue töö sisestamisel rangelt selle sidumist olemasoleva kliendi ja objektiga. See tähendab, et täiesti uute, süsteemis varasemalt defineerimata klientide või objektide puhul, ei ole tööde loomine vahetult võimalik enne nende eelnevat süsteemi sisestamist. Eeltoodud asjaolu võib teatud kiireloomulistel või esmakordsetel kasutusjuhtudel töö loomise protsessi pikendada. Nimetatud piirang tuleneb tarkvaralahenduse varasematest arhitektuuriotsustest ning selle kõrvaldamist ei käsitletud käesoleva töö raames, kuna see jäi välja töö eesmärkidest.

Kokkuvõtvalt võib öelda, et saadud tagasiside oli valdavalt positiivne, kinnitades lahenduse kasulikkust ja tehnilist teostust nii platvormi haldaja kui ka lõppkasutaja perspektiivist. Esitatud tähelepanek tööde loomise funktsionaalsuse kohta pakub aga selge sisendi võimalikeks edasiarendusteks, et lahenduse paindlikkust ja kasutusmugavust veelgi suurendada.

6.3.3 Lõputöö piirangud

Käesoleva lõputöö raames valminud lahendusel on teatud piirangud, mis tulenevad peamiselt töö ajalisest ja ressursilisest mahust. Esiteks, kuigi loodud Syda API katab olulisemad andmeobjektid nagu tööd ja töö raportid, ei hõlma see veel kogu Syda platvormi funktsionaalsust, millest peamine on rendihalduse moodul. Seega on API laiendamise potentsiaal märkimisväärne. Teiseks, süsteem ei taga veebihaagi teavituste robustset edastamist, kuna puuduvad sõnumijärjekorra ja ühendusprobleemide tõrkekäsitluse implementatsioonid. Järelikult ei tehta korduskatseid ning teavitus kaotatakse esimese edastusvea korral. Kolmandaks, lahenduse valideerimine toimus piiratud hulga stsenaariumite ja testkasutajatega, seega laiaulatuslikku kasutajakogemuse testimist ja jõudlustestimist suure koormuse all ei

teostatud. Samuti keskendus töö integratsioonivõimekuse loomisele, mitte niivõrd valmis töövoogude kataloogi pakkumisele.

6.3.4 Edasiarendusettepanekud

Tuginedes valminud lahendusele, selle valideerimise esmastele tulemustele ja tuvastatud piirangutele, saab välja tuua mitmeid edasiarendusettepanekuid. Syda API osas võiks kaaluda täiendavate otspunktide loomist teistele andmeobjektidele, näiteks rendimooduli spetsiifilistele andmetele, ning olemasolevate andmeobjektide funktsionaalsuse laiendamist. Süsteemi töökindluse ja veebihaagi teavituste edastamise robustsuse suurendamiseks tuleks kaaluda sõnumijärjekorra ja korduskatsetega tõrkekäsitluse lisamist. Samuti oleks kasulik laiendada veebihaakide poolt toetatud sündmuste hulka, et pakkuda veel rohkem paindlikkust sündmuste töötlemisel.

Make'i kohandatud rakenduse puhul oleks loogiline samm uute moodulite lisamine vastavalt API laienemisele ning kasutajate tagasisidele. Võiks arendada ka keerukamaid komposiitmooduleid, mis teostaksid mitu seotud operatsiooni ühe korraga. Kuna Syda rakendus on nüüdseks Make'i platvormil avalikult kättesaadav, on edasiseks oluline samm selle aktiivne haldamine, uuendamine vastavalt API muutustele ning kasutajatelt saadava tagasiside põhjal täiendamine.

Üldisemas plaanis oleks väärtuslik luua põhjalikum tehniline dokumentatsioon ja kasutusjuhendid nii API kui ka Make'i rakenduse kasutajatele, sealhulgas peamiselt praktilised näidisstsenaariumid levinumate integratsioonivajaduste katmiseks. Samuti on oluline jätkata API kasutuse monitoorimist, et tagada selle stabiilne ja turvaline toimimine ning koguda sisendit edasisteks parandusteks.

7 Kokkuvõte

Käesoleva töö eesmärk oli luua Syda kalendripõhisele välitööde rakendusele paindlik integratsioonivõimekus väliste äriprotsessidega, lahendades sellega probleemi, mis seisnes andmete käsitsi sisestamises erinevatesse süsteemidesse ning platvormi jäikades andmeekspordi võimalustes. Analüüs näitas selget vajadust standardiseeritud ja kasutajasõbraliku lahenduse järele, mis võimaldaks Syda klientidel platvormil olevaid andmeid tõhusalt kasutada ja töödelda vastavalt nende unikaalsetele ärilistele vajadustele. Integratsioonivõimaluste hindamise tulemusena valiti sobivaimaks platvormiks Make, mis pakub head tasakaalu kasutusmugavuse, paindlikkuse ja kuluefektiivsuse vahel.

Töö praktilise osana realiseeriti Syda olemasoleva tagarakenduse laiendusena kahetasandiline integratsioonilahendus. Esmalt arendati välja dokumenteeritud rakendusliides, mis võimaldab välistel süsteemidel programmaatiliselt ligi pääseda Syda peamistele andmeobjektidele, nagu tööd ja töö raportid. Selleks implementeeriti uued otspunktid ning turvalisuse tagamiseks API võtmete põhine autentimismehhanism. Lisaks loodi veebihaakide süsteem, mis võimaldab reaalajas teavituste saatmist välistesse süsteemidesse oluliste sündmuste toimumisel Syda platvormil.

Teise peamise komponendina loodi Syda API tarbeks kohandatud rakendus Make platvormil. See rakendus lihtsustab oluliselt Syda andmetega opereerimist Make töövoogudes, abstraheerides API tehnilised detailid ning pakkudes intuitiivseid mooduleid andmete lugemiseks, loomiseks ja uuendamiseks.

Valminud lahenduse väärtus seisneb eelkõige Syda platvormi konkurentsivõime tõusus ja arendusressursi säästus, kuna väheneb vajadus spetsiifiliste integratsioonide järele. Klientidele pakub lahendus suuremat paindlikkust andmete kasutamisel, võimaldades äriprotsesside automatiseerimist, mis suurendab efektiivsust.

Lahenduse valideerimine toimus praktilise stsenaariumi abil Make platvormil, kus demonstreeriti automatiseeritud klienditagasiside kogumise töövoogu. Valideerimine kinnitas API,

veebihaakide ja Make rakenduse tehnilist toimivust ning lahenduse ärilist väärtust. Saadud tagasiside oli valdavalt positiivne, tuues esile lahenduse selguse ja kasutusmugavuse.

Lõputöö käigus püstitatud eesmärgid said suures osas täidetud – valmis toimiv ja testitud integratsioonilahendus, mis võimaldab Syda andmete automatiseeritud andmevahetust ja vähendab käsitsi töö vajadust. Töö tulemused loovad tugeva aluse Syda platvormi edasisteks integratsioonialasteks arendusteks.

Kasutatud kirjandus

[1] Syda Solutions OÜ. „Syda“. *https://www.syda.io/* (2025).

[2] Zapier Inc. „Zapier“. *https://zapier.com/* (2025).

[3] Inc. Celonis. „Make“. *https://www.make.com/en* (2025).

[4] n8n GmbH. „n8n“. *https://n8n.io/* (2025).

[5] Microsoft Corporation. „Microsoft Excel“. *https://www.microsoft.com/en-us/microsoft-365/excel* (2025).

[6] Zapier Inc. „Zapier hinnastus“. *https://zapier.com/pricing* (2025).

[7] Inc. Celonis. „Make hinnastus“. *https://www.make.com/en/pricing* (2025).

[8] Inc. Bubble Group. „Bubble“. *https://bubble.io/* (2025).

[9] Inc. Bubble Group. „Bubble hinnastus“. *https://bubble.io/pricing* (2025).

[10] n8n GmbH. „n8n hinnastus“. *https://n8n.io/pricing/* (2025).

[11] Atlassian Corporation. „Atlassian Bitbucket“. *https://bitbucket.org/product/* (2025).

[12] SonarSource SA. „SonarQube“. *https://www.sonarsource.com/products/sonarqube/* (2025).

[13] Niels Provos David Mazières. „A Future-Adaptable Password Scheme“. Teoses: *Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference*. 1999.

[14] Syda Solutions OÜ. „Syda avaliku API dokumentatsioon“. *https://jobs.syda.io/docs* (2025).

[15] Inc. Celonis. „Syda integratsioon Make platvormil“. *https://www.make.com/en/integrations/syda* (2025).

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

Mina, Saamo Kurvits

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Välitööde rakenduse integreerimine automatsiooniplatvormiga", mille juhendaja on Siim Rebane
    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.

03.06.2025

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