

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Rasmus Vananurm 222527IAIB

# **Iseteeninduslik registreerimisrakendus Entigo Platvormi jaoks**

Bakalaureusetöö

Juhendaja: Siim Rebane

BSc

Kaasjuhendaja: Rein Remmel

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: Rasmus Vananurm

04.06.2025

## **Annotatsioon**

Käesoleva bakalaureusetöö eesmärgiks on arendada Entigo OÜ jaoks iseteeninduslik registreerimisrakendus, mis võimaldab potentsiaalsetel klientidel alustada ettevõtte teenuste kasutamist iseteenindsulikult, ilma otsese kontakti või manuaalse sekkumiseta. Töö keskendub probleemile, et iga uue kliendi ükshaaval teenindamine on aeganõudev ning piirab ettevõtte kasvuvõimet.

Nõuete kaardistamisel ja valideerimisel tehti koostööd ettevõttega. Lahenduse väljatöötamisel keskenduti olemasolevate teenuste integreerimisele valdkondades nagu autentimine, arveldusandmete haldus ja maksesüsteemi liidestus. Süsteem kavandati modulaarse ja skaaleeritava arhitektuuriga ning juurutati konteineripõhise infrastruktuuri abil pilveteenusesse.

Töö tulemusena valmis tehniline prototüüp, millega on võimalik tutvuda veebis <https://app.entigo.com/sign-up>. Registreerimisprotsessi käigus loob süsteem uue kasutaja jaoks vajalikud ressursid ja suunab ta kogu protsessi jooksul loogilisse ja turvalisse voogu. Maksehalduse protsessis on kasutajal võimalik iseteeninduslikult muuta makseandmeid ja hallata tellimusi.

Teoreetiline osa kirjeldab prototüübi arendusprotsessi, alustades kliendipoolsete vajaduste ja ootuste väljaselgitamisest. Sellele järgnes lahenduse funktsionaalse ulatuse määratlemine ning töö eesmäärke toetava tehnoloogilise raamistiku leidmine. Arendatud prototüüpi valideeriti lähtudes eelnevalt määratletud nõuetest, keskendudes selle sobivusele püstitatud probleemi lahendamisel ning potentsiaalsetele edasiarendusvõimalustele tulevikus.

Töö käigus püstitatud nõuded said suures osas täidetud, kuid piiratud aja tõttu jäi realiseerimata süsteemi täielik tõrketaluvus, mille edasiarendamine võimaldaks minimeerida potentsiaalseid ärilisi kahjusid. Töö tulemus annab tugeva aluse edasiseks arenduseks ja skaleerimiseks ning on juba kasutusel reaalsete kasutajate juurutamiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 29 leheküljel, 6 peatükki, 12 joonist.

## **Abstract**

### **Self-Service Registration System for the Entigo Platform**

This bachelor's thesis presents the design and development of a self-service registration application for Entigo OÜ, a company offering scalable, cloud-based infrastructure solutions tailored for growth-oriented IT teams. The goal of the project was to eliminate the need for manual onboarding of new clients by automating the account creation process in a secure, scalable, and user-friendly manner. Prior to this development, each customer had to be onboarded manually by the company, involving steps such as creating user accounts, assigning access rights, and configuring billing – processes that hindered the scalability of the platform and introduced operational inefficiencies.

The thesis began with mapping the limitations of the existing solution and the identification of key usability and business-related pain points. The problem was approached in close collaboration with the company, allowing for the co-definition of requirements and iterative validation of the scope. It was understood that the main barrier to scaling was the lack of a streamlined, automated registration flow that could reduce administrative overhead and align with modern user expectations for SaaS platforms.

The implemented solution integrates reliable third-party services in critical areas such as identity verification, billing data collection, and payment gateway management. The registration application uses a modular system architecture, based on principles of domain-driven design and containerized deployment in a Kubernetes cluster hosted on Amazon Web Services (AWS). Authentication is handled through OAuth 2.0 using Google and Zitadel, where social login mechanisms allow for a more seamless user experience while still ensuring email authenticity and data integrity. In order to maintain a stateless backend and reduce system complexity, user data such as email addresses are retrieved directly from the OAuth provider at key stages of the flow rather than being stored throughout the session.



The prototype, deployed at <https://app.entigo.com/sign-up>, guides the user through a logical and secure onboarding sequence. This includes authentication, pre-filled personal data forms, terms acceptance, domain name uniqueness verification, automated environment setup, and billing configuration. Upon successful registration, the system creates the necessary infrastructure resources for the user and ensures that billing options are self-service and modifiable through an integrated portal.

The theoretical part of the thesis details the development process, starting with understanding stakeholder goals and translating them into technical requirements. It further discusses the challenges of integrating external identity providers, ensuring system security, and designing an extensible architecture fit for future scalability. The development approach was deliberately hypothesis-driven and iterative rather than strictly following Agile or Lean methodologies, emphasizing user validation and business value.

Although the resulting solution satisfies most of the defined requirements and is already in use for real user onboarding, some limitations remain. Notably, full fault tolerance and resilience mechanisms were not implemented within the time constraints, leaving room for further development to mitigate potential business risks caused by dependency on third-party systems.

Overall, the project lays a solid foundation for future enhancements of the Entigo Platform's onboarding experience. The resulting prototype not only improves user experience and reduces internal workload but also positions the platform for broader adoption as a SaaS solution.

The thesis is in Estonian and contains 29 pages of text, 6 chapters, 12 figures.

## Lühendite ja mõistete sõnastik

|       |                                                                                                                           |
|-------|---------------------------------------------------------------------------------------------------------------------------|
| SaaS  | Tarkvara kui teenus ( <i>Software-as-a-Service</i> )                                                                      |
| SSO   | Ühene lahendus mitmesse rakendusse sisse logimiseks ( <i>Single Sign-on</i> )                                             |
| CI/CD | Tarkvaraarenduse protsess, mis automatiseerib tarkvara väljalaset ( <i>Continuous Integration/Continuous Deployment</i> ) |
| UI    | Kasutajaliides ( <i>User interface</i> )                                                                                  |
| CSS   | Stiilikeel, millega kujundatakse HTML'i põhine veebileht ( <i>Cascading Style Sheets</i> )                                |
| HTML  | Märgistuskeel, millega luuakse veebilehe struktuur ( <i>HyperText Markup Language</i> )                                   |
| API   | Liides, mis võimaldab tarkvarasüsteemidel omavahel andmeid vahetada ( <i>Application Programming Interface</i> )          |
| CRM   | Kliendisuhete haldus ( <i>Customer relationship management</i> )                                                          |
| DDD   | Domeenipõhine disain ( <i>Domain-Driven Design</i> )                                                                      |
| AWS   | Amazoni pakutav pilveteenus ( <i>Amazon Web Services</i> )                                                                |
| URL   | Aadress, mis näitab, kus mingi ressurss internetis asub ( <i>Uniform Resource Locator</i> )                               |
| REST  | Olekute esitusviisi edastamine ( <i>Representational State Transfer</i> )                                                 |

## Sisukord

|       |                                                           |    |
|-------|-----------------------------------------------------------|----|
| 1     | Sissejuhatus.....                                         | 10 |
| 1.1   | Olemasolev rakendus.....                                  | 11 |
| 1.2   | Valmislahenduste integreerimise vajalikkus.....           | 11 |
| 2     | Nõuded .....                                              | 12 |
| 2.1   | Funktsionaalsed nõuded.....                               | 12 |
| 2.2   | Mittefunktsionaalsed nõuded.....                          | 13 |
| 3     | Metoodika.....                                            | 16 |
| 3.1   | Projekti arhitektuur .....                                | 16 |
| 3.2   | Projekti haldustarkvara .....                             | 18 |
| 3.3   | Eesrakendus .....                                         | 18 |
| 3.4   | Tagarakendus.....                                         | 19 |
| 3.5   | Olulised liidestused .....                                | 20 |
| 3.5.1 | Google'i autentimine .....                                | 20 |
| 3.5.2 | Zitadel identiteedihaldus .....                           | 21 |
| 3.5.3 | Posthog.....                                              | 21 |
| 3.5.4 | Stripe.....                                               | 21 |
| 3.5.5 | Salesforce.....                                           | 22 |
| 4     | Arendus .....                                             | 23 |
| 4.1   | Arenduspõhimõtted .....                                   | 23 |
| 4.2   | Eesrakenduse arendus.....                                 | 24 |
| 4.2.1 | Visuaalne prototüüp.....                                  | 24 |
| 4.2.2 | Struktuur.....                                            | 25 |
| 4.2.3 | Identiteedi terviklikkuse tagamine ja haldus .....        | 27 |
| 4.3   | Tagarakenduse arendus .....                               | 27 |
| 4.4   | Koodi väljastamine.....                                   | 28 |
| 5     | Tulemused, tagasiside ja edasised arendusvõimalused ..... | 29 |
| 5.1   | Vood .....                                                | 29 |

|       |                                                                                                       |    |
|-------|-------------------------------------------------------------------------------------------------------|----|
| 5.1.1 | Registreerimisvoog.....                                                                               | 29 |
| 5.1.2 | Maksehaldusvoog.....                                                                                  | 31 |
| 5.2   | Turvalisus .....                                                                                      | 32 |
| 5.3   | Testimine .....                                                                                       | 33 |
| 5.4   | Ettevõtte tagasiside.....                                                                             | 34 |
| 5.5   | Analüüs .....                                                                                         | 34 |
| 5.5.1 | Projekti arendusprotsess.....                                                                         | 35 |
| 5.5.2 | Nõuete täitmine ja analüüs .....                                                                      | 35 |
| 5.6   | Edasised arendusvõimalused .....                                                                      | 36 |
| 6     | Kokkuvõte.....                                                                                        | 38 |
|       | Kasutatud kirjandus .....                                                                             | 39 |
|       | Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks..... | 41 |
|       | Lisa 2 – Entigo Platvormi kasutajateekond.....                                                        | 42 |
|       | Lisa 3 – Registreerimisvoo vaated.....                                                                | 43 |
|       | Lisa 4 – Maksehaldusvoo vaated .....                                                                  | 45 |

## **Jooniste loetelu**

|                                                     |    |
|-----------------------------------------------------|----|
| Joonis 1. Projekti arhitektuur.....                 | 17 |
| Joonis 2. Eesrakenduse struktuur. ....              | 26 |
| Joonis 3. Registreerimisvoog.....                   | 29 |
| Joonis 4. Maksehaldusvoog.....                      | 31 |
| Joonis 5. Autentimise ja valideerimise etapid. .... | 32 |

# 1 Sissejuhatus

Tarkvara kui teenus (SaaS) on kaasaegse tarkvaratööstuse üks juhtivaid mudeleid, pakkudes kasutajatele interneti kaudu juurdepääsu täisfunktsionaalsele rakendustarkvarale. See mudel hõlmab lisaks infrastruktuuri- ja platvormiteenustele ka kasutusvalmis rakendusi, mis on kergesti kättesaadavad ning mille uuendused toimuvad automaatselt, tagades seeläbi kasutajatele mugava ja ajakohase kogemuse [1].

Autentimine on iseteeninduslikes süsteemides keskse tähtsusega tagamaks, et ainult volitatud kasutajad pääsevad ligi tundlikele andmetele ja funktsionaalsusele. SaaS-teenuse jätkusuutlikuks toimimiseks on kriitilise tähtsusega registreerimine lahendada iseteeninduslikult. See võimaldab kasutajatel iseseisvalt konto luua, hallata oma andmeid ning pääseda ligi erinevatele teenustele ilma otsese teenusepakkuja sekkumiseta. Selline lähenemine suurendab kasutajakogemuse sujuvust ja vähendab halduskoormust. Sarnast mudelit kasutavad oma tellimuspõhiste toodete pakkumiseks mitmed suured ettevõtted nagu Netflix, Google, Zoom ja teised [2].

Käesoleva töö peamine eesmärk on arendada Entigo Platvormi jaoks registreerimisrakendus, mis muudab olemasoleva toote kasutusele võtmise mugavamaks eri suurusega klient-ettevõtete jaoks. Teine eesmärk on tuua Entigo (edaspidi ettevõtte) arendusmeeskonda juurde teadmisi hajussüsteemide töökindla integreerimise ja ehitamise kohta. Implementeeritakse algne registreerimisrakendus Entigo Platvormi jaoks, mis hõlmab Google SSO põhist registreerimist, maksevärava lisamist ning kasutajategevuse seiramist. Töö käigus hinnatakse lahenduse sobivust ettevõtte kontekstis.

Lõputöö on jaotatud kuueks peatükiks. Teises peatükis esitatakse ettevõtte poolsed nõuded. Kolmandas peatükis tutvustatakse rakenduse arhitektuuri ning antakse ülevaade tehnoloogilistest valikutest. Neljandas peatükis selgitatakse arendusprotsessi. Viiendas peatükis käsitletakse valminud prototüübi vastavust püstitatud nõuetele ning hinnatakse arendustöö tulemuslikkust. Kuues peatükk on lõputöö kokkuvõte.

## 1.1 Olemasolev rakendus

Entigo Platvormi (edaspidi ka platvorm) eesmärk on kasvule suunatud infotehnoloogia meeskondadele skaleeritavate pilvepõhiste infrastruktuuri lahenduste pakkumine võimendades tehnoloogiaid nagu Kubernetes ja Amazon Web Services [3].

Teenuse praegune lahendus pakub autentimist Zitadel rakenduse kaudu, kuid eeldab iga uue kliendi korral Entigo poolset käsitsi kontode loomist, kasutajaõiguste määramist ning edaspidist maksete haldamist. Olemasolev implementatsioon ei ole ettevõtte vaatest jätkusuutlik, kuna ei toeta platvormi laialdast kasutuselevõttu ja turule toomist SaaS-teenusena. Probleemi keskmes on iseteenindusliku konto ettevalmistamise võimekuse puudumine, mis võimaldaks kiiret ja mugavat teenuse kasutuselevõttu uutele klientidele.

Manuaalsed protsessid ei ole mitte ainult ajamahukad, vaid suurendavad ka vigade tekkimise tõenäosust. Lisaks muudab selline lähenemine süsteemi skaleerimise administratiivselt keeruliseks, sest uute kasutajate arvu kasvades suureneb proportsionaalselt haldustöö maht. Samuti mõjutab automatiseeritud registreerimisvoo puudumine otseselt kasutajakogemust ning võib olla seeläbi takistuseks uute klientide liitumisele.

Need puudused teevad toote esmase kasutuselevõtmise potentsiaalsete klientide jaoks kalliks ja ebamugavaks. Seetõttu on möödapääsmatu luua registreerimisrakendus, mis ühildub süsteemi olemasolevate osadega ja kavandatud kasutaja töövooga (vaata joonist 6). Selline lahendus vähendab halduskoormust, toetab toote edasist skaleerimist ning vastab kaasaegsete kasutajate ootustele sujuva ja iseseisva kasutajakogemuse osas.

## 1.2 Valmislahenduste integreerimise vajalikkus

Kasutajaandmete ja maksete haldamise süsteemide arendamine nullist on keeruline, aeganõudev ja ressursimahukas. Tänapäeva tarkvaraarenduse praktikas on üldlevinud lähenemine kasutada olemasolevaid, usaldusväärseid lahendusi, mis pakuvad standardiseeritud ja turvalisi võimalusi autentimiseks, konto haldamiseks ning maksesüsteemidega integreerimiseks. Alustaval ettevõttel puudub sageli vajalik ressurss, et arendada neid komponente iseseisvalt, seetõttu on otstarbekas kasutada olemasolevaid väliseid teenuseid, mida saab süsteemiga integreerida. Selline lähenemine võimaldab keskenduda toote põhifunktsionaalsuse arendamisele ning tootega võimalikult kiiresti turule tulla [4].

## 2 Nõuded

Lõputöö eesmärk ja nõuded sõnastati koostöös ettevõtte esindajaga, kellega ühiselt määratleti lahenduse peamised ootused. Töö käigus täpsustus ja kujunes lõplik skoop autori jooksva tagasiside põhjal, mis tugines tehnoloogilise prototüüpimise tulemustele. Arendusprotsess ei lähtunud rangelt mõnest kindlast metoodikast, nagu Agile või Lean, vaid keskendus eelkõige kasutajaprobleemide sisulisele mõistmisele ja nende valideerimisele läbi praktilise katsetamise. Seetõttu oli tööprotsess teadlikult hüpoteesipõhine ja iteratiivne: arenduse suunamine toimus nii lõputöö autori kui ettevõtte tagasiside alusel, vältimaks hilisemaid kulukaid muudatusi ning tagamaks toote tegelik väärtus lõppkasutajale [5]. Platvormi kasutajateekond on kujutatud peatükis Lisa 2 joonisel 6, mis visualiseerib kasutaja liikumist reklaamikanalist kuni maksehalduse ja tugiteenuseni. Järgnevad nõuded tulenevad kirjeldatud teekonnast ja on suunatud selle tõhustamisele.

### 2.1 Funktsionaalsed nõuded

#### 1. Kasutajakonto loomine ja registreerimine

- Kasutaja saab alustada konto loomist autentimisvaliku kaudu: autentimisvaliku olemasolu parandab kasutusmugavust ja paindlikkust.
- Autentimise järel täidetakse isikuandmete vaade eeltäidetud nime ja e-posti aadressiga: see vähendab käsitsi sisestamise vajadust, võimaldades kiiremat registreerimist, samas ei teki segadust, milliseid andmeid kasutatakse.
- Kasutaja peab kinnitama teenustingimused ja sisestama unikaalse domeeninime: teenustingimustega nõustumine on juriidiline eeltingimus; domeeninimi on vajalik platvormil loodavate ressursside eristamiseks.
- Domeeninime unikaalsust kontrollitakse reaalselt sisestamise käigus: ennetab registreerimisprotsessi katkestusi hilisemates etappides ja suurendab sujuvust.

#### 2. Automaatne ressursside loomine

- Registreerimise kinnitamisel luuakse kasutajaga seotud teenusobjektid ja keskkonnad: automatiseerimine vähendab manuaalse töö vajadust ning võimaldab



skaleeritavat ja kiiret ressursside ülesseadmist.

### 3. Teavitused ja kasutajavood

- Kasutajale saadetakse e-kiri konto lõplikuks aktiveerimiseks: parandab kasutajamugavust ja arenduskiirust, sest tegemist on standardiseeritud vooga identiteedihalduse pakujate seas.
- Kasutajale kuvatakse teade edukast registreerimisest ning antakse juhised järgnevateks sammudeks: vähendab vajadust klienditoe sekkumiseks.

### 4. Maksehalduse voog

- Kasutajal on platvormilt ligipääs maksehaldusele: maksehaldus peab olema kasutajale igal hetkel ligipääsetav, et tagada usaldus platvormi vastu.
- Vajadusel palutakse täita arveldusandmed (nimi, aadress, telefon): vajalikud arvelduseks ning vastavuses maksude ja raamatupidamise nõuetega.
- Kasutaja suunatakse maksehalduse portaalile, kus on võimalik hallata pakette, maksemeetodeid ja vaadata arvelduse ajalugu: annab kasutajale kontrolli oma paketi üle ning vähendab vajadust manuaalseks kliendihalduseks.

### 5. Ootenimekiri ja kontaktandmete salvestamine

- Platvormi tootmisvalmiduse eelses faasis on võimalik kasutaja suunata lihtsustatud registreerimisvoogu: võimaldab koguda huviliste kontaktid enne platvormi lanseerimist, mis toetab turule sisenemise strateegiat.
- Sellisel juhul salvestatakse kasutaja andmed ettevõtte Salesforce'i süsteemi müügivihjena: võimaldab müügimeeskonnal hallata potentsiaalseid kliente läbi olemasoleva CRM-lahenduse.

## 2.2 Mittefunktsionaalsed nõuded

### 1. Skaleeritavus ja hooldatavus

- Rakendus järgib modulaarset arhitektuuri ja domeenipõhise disaini põhimõtteid: need võimaldavad süsteemi komponentide sõltumatut arendamist, testimist ja laiendamist.
- Rakendus on jagatud ees- ja tagarakenduseks, millel on selgelt eristatud vastutusvaldkonnad: arhitektuurne eristatus pakub suuremat turvalisust.
- Loodav rakendus peab sobituma ettevõtte olemasoleva infrastruktuuriga: see kindlustab, et ettevõttel on võimekus rakenduse haldamiseks ja edasiarendamiseks.

miseks.

## 2. Turvalisus

- Kasutajaandmete edastamisel välditakse lihtteksti; autentimise ja autoriseerimise korral kasutatakse turvalisi pääsutunnuseid (tokeneid): tagab vastavuse tänapäevaste infoturbe nõuetega, vältimaks isikuandmete lekkimist või kuritarvitamist.
- Eesrakenduse ja tagarakenduse vahelise suhtluse turvalisus on tagatud valideeritud andmestruktuuride ja serveripoolsete kontrollide abil: vähendab ründevektoreid ja tagab, et sisestatud andmed on oodatud kujul.

## 3. Keskkond ja ettevalmistamine

- Demo- ja toodanguvalmis rakendus töötab konteineripõhises infrastruktuuris ning on juurutatud Kubernetese klastrisse AWS pilveteenuse kaudu: tagab ühetaolisuse platvormiga.
- Juurutamiseks kasutatakse GitHub Actions automatiseeritud töövooge (*CI/CD pipeline*): tagab, et koodimuudatused jõuavad kiiresti ja usaldusväärselt toodangukeskkonda, vähendades käsitsi sekkumise vajadust ja inimlikke vigu.
- Rakenduse konteineri pildid salvestatakse DockerHub'i ning sealtkaudu juurutatakse need sobivasse keskkonda: võimaldab keskset ja hallatavat kujutiste hoiustamist, mis lihtsustab juurutusprotsessi.

## 4. Jälgitavus ja testitavus

- Kasutajakogemuse ja funktsionaalsuse testimiseks rakendatakse dünaamilisi funktsioonilippe (*feature flags*): võimaldab uusi funktsionaalsusi kasutajatele järk-järgult kättesaadavaks teha.
- Süsteemi käitumist on võimalik jälgida reaajas: toetab tõrgete kiiret tuvastamist ja aitab analüüsida kui kaugele kasutaja eeldatud voos jõuab.

## 5. Kasutajakogemus

- Kasutajaliides järgib eelnevalt valideeritud visuaalset prototüüpi: tagab visuaalse ja funktsionaalse järjepidevuse ning ettevõtte visiooni täitmise.
- Liidestatud süsteemide tõrke puhul tekkinud probleemide tõttu ei tohiks kasutaja konto ettevalmistamine katkeda: parandab süsteemi vastupidavust; tagab, et kliente ei kaotata mõne välise teenuse ajutise rikke tõttu.
- Disainis lähtutakse kasutajasõbralikkusest ning järgitakse visuaalsete ja funktsionaalsete reeglitega.

sionaalsete komponentide korduvkasutatavuse põhimõtteid: korduvkasutatavad komponendid kiirendavad arendust, vähendavad visuaalset ebahühtlust ja toetavad paremat hooldatavust.

- Jälgitakse kohanduva disaini põhimõtteid: kogu rakenduse funktsionaalsus peab olema mugavalt kättesaadav nii horisontaalses kui vertikaalses vaates.

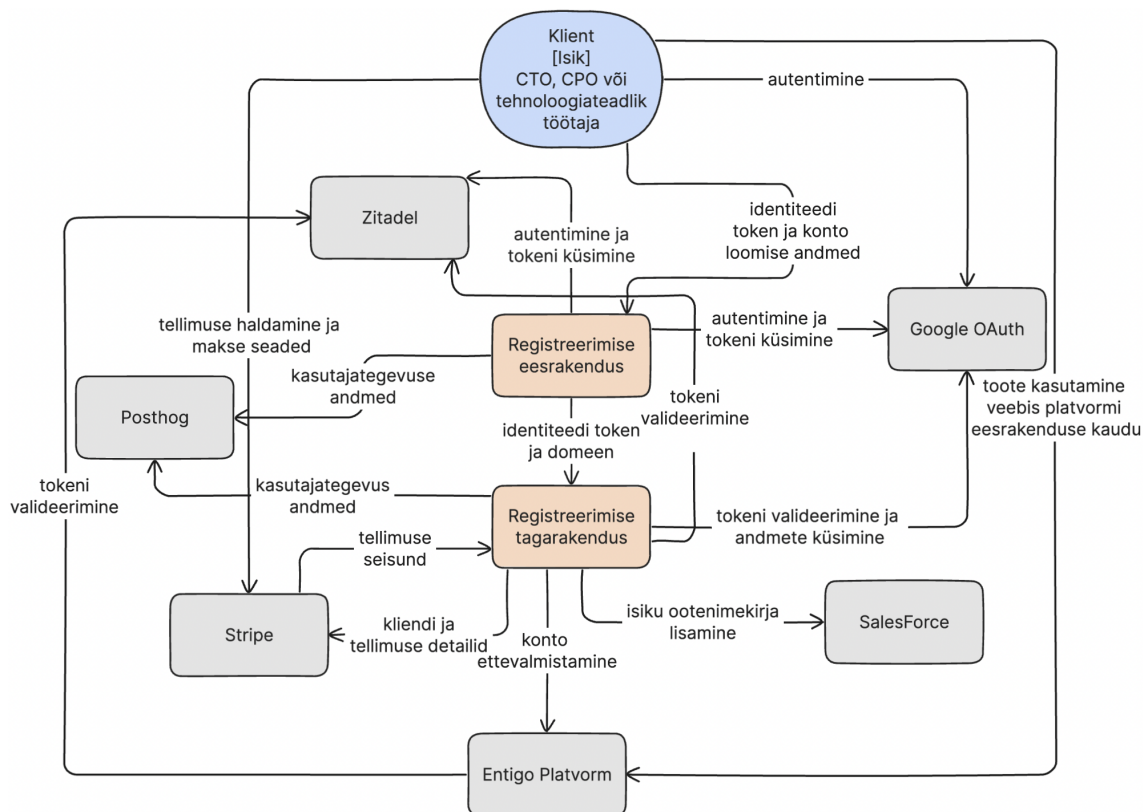
## **3 Metoodika**

Nõuete alusel valis töö autor tehnoloogiad, mis võimaldaksid luua vastava veebirakenduse. Rakenduse üleseks programmeerimiskeeleks valiti TypeScript. Tagarakendus loodi Node.js raamistikuga ja eesrakenduse loomiseks kasutati React raamistikku. Projekti halduseks kasutati ettevõtte GitHub'i keskkonda. Tehnoloogiliste valikute kõige olulisemaks põhjuseks oli ettevõttesisene kompetents vastavaid tehnoloogiaid kasutavat rakendust tulevikus edasi arendada ning hallata. Järgnevad peatükid selgitavad valikuid täpsemalt.

### **3.1 Projekti arhitektuur**

Arhitektuuri eesmärk oli saavutada modulaarne, hästi hooldatav struktuur, mis toetab edasist arendust ning pakub kasutajatele kiiret ja sujuvat kogemust. Selleks oli tarvis, et registreerimisrakendus pakuks sujuvat ühilduvust platvormiga, olles samas Entigo Platvormi ülejäänud koodibaasist eraldiseisev rakendus. Lõputöö skoobis oli kaks eraldiseisvat voogu, mille käsitlemine kuulub registreerimisrakenduse vastutusalade hulka: registreerimise voog ja maksehalduse voog.

Arendusprotsessis juhinduti domeenipõhise disaini (DDD) põhimõtetest ehk keskse tähtsusega oli äri loogika mõtestamine ning olulist rolli mängis konkreetse domeeni mõistete ja kontekstuaalse sõnavara kasutamine, mis tagas parema kooskõla tehnilise ja ärilise tähenduse vahel [6].



Joonis 1. Projekti arhitektuur.

Joonisel 1 on kujutatud registreerimisrakenduse suhtluse struktuur nii ettevõttesiseste kui ka väliste rakendustega. Lõputöö käigus arendatavad komponendid on märgitud oranži ja kasutaja sinise värviga, liidestatud süsteemid on hallid. Ettevõtte sisesteks liidestusteks on Entigo Platvormi ees- ja tagarakendus. Ülejäänud integreeritud süsteemid on välised teenused, millega suhtlus toimub enamikul juhtudest teekide abil, mis on lisatud Node Package Manageri kaudu.

Platvormi eesrakendus suhtleb registreerimisrakendusega seetõttu, et kogu platvormi makselahenduste loogika on koondatud just registreerimisrakendusse. Selline arhitektuuriline otsus tuleneb vajadusest võimaldada platvormi paigaldamist suuremate klientide kohalikesse süsteemidesse. Seetõttu peab süsteem olema üles ehitatud viisil, mis võimaldab registreerimise ja makselahendustega seotud funktsionaalsuste eraldamist platvormi põhifunktsionaalsusest, tagades paindlikkuse erinevate kasutusstsenaariumite puhul. Liidestus platvormi tagarakendusega on eelkõige vajalik ettevõtte siseste ressursside loomiseks konto tekitamise ja ettevalmistamise protsessis. Väliste liidestuste kohta on võimalik lugeda peatükis 3.5 Olulised liidestused.

## 3.2 Projekti haldustarkvara

Käesolevas projektis kasutati projekti halduseks GitHub'i keskkonda, mis on arendajate seas laialdaselt levinud tööriist tänu oma võimekale versioonihaldusele ja integreeritud töövoogudele. GitHub võimaldab hallata koodi muudatusi Git-versioonihalduse kaudu. Pakutavatest võimalustest olid lõputöö kontekstis esmatähtsad *pull request*'ide ja koodi ülevaatuste võimalus ning harude (*branches*) loomine [7]. Kuigi lõputöö raames valmiva koodi kirjutamisega tegeles töö autor üksi, siis toimusid regulaarsed ettevõttepoolsed koodi ülevaated, et tagada koodi kvaliteet. Samuti kasutati lõputöö raames GitHubi Actions funktsionaalsust ehk CI/CD tööriistu, mis võimaldavad automaatselt ehitada ja juurutada koodi igal muudatusel, vähendades käsitööd ja vigade tekkimise võimalust [8].

Ülesannete haldamiseks kasutati lõputöö raames Jira projekti halduskeskkonda, mis on ettevõttesisene standard. Jira võimaldas ülesandeid selgelt struktureerida, jagada need etapi viisil teostatavateks üksusteks ning jälgida nende edenemist kogu arendusprotsessi vältel [9]. Tänu sellele oli võimalik hoida töövoog hästi organiseerituna ning tagada läbipaistev ja süsteemne projekti juhtimine.

## 3.3 Eesrakendus

Eesrakendus on tarkvararakenduse osa, millega lõppkasutaja vahetult suhtleb. See hõlmab kasutajaliidest (UI) ning kõiki funktsioone, mis töötavad kasutaja brauseris või seadmes, võimaldades andmete sisestamist, navigeerimist ning visuaalset tagasisidet süsteemiga suhtlemisel.

Arendustööle eelnevalt loodi kasutajaliidese visuaalne prototüüp, et kavandada süsteemi struktuur ja kasutusloogika. Prototüüp valmis Figma keskkonnas, mis on laialdaselt kasutatav tööriist veebirakenduste kujundamiseks. Selle etapi mõtestatud läbiviimine aitas lõputöö autoril ennetada võimalikke kasutajakogemuse probleeme ja andis eesrakenduse arendusele selge visuaalse suuna.

Käesolevas projektis otsustas autor eesrakenduse arendamiseks kasutada TypeScripti, mis on JavaScripti laiendus, kuhu on lisatud staatilised andmetüübid. TypeScripti eeliseks standardse JavaScripti ees on paremad arendusaegsed veatuvastused, koodi loetavus ja hooldatavus, mis muudavad arendusprotsessi mugavamaks ja koodi töökindlamaks,

vähendades veaohklikkust. [10].

Eesrakenduse struktuur loodi React raamistiku abil, mis on hetkel kõige populaarsem JavaScripti raamistik veebirakenduste arendamiseks [11]. React võimaldab luua korduvkasutatavaid komponente ja hallata rakenduse seisundit tõhusalt, mis teeb selle sobivaks ka keerukamate rakenduste ülesehitamiseks [12]. Oluline aspekt Reacti valimiseks oli asjaolu, et see on juba ettevõttes kasutusel ja samade tehnoloogiate kasutamine aitab ettevõttel piirata tehnoloogilise keerukuse kasvu.

Selleks, et luua kasutajaliidese komponente nagu nupud, sisestusväljad, ikoonid ja teated kasutati peamiselt valmis UI-komponentide teeki Ant Design, mis pakub laia valikut viimistletud ja funktsionaalselt paindlikke elemente. Väline komponentide teek valiti, et kiirendada arendust ja tagada kasutajaliidese visuaalne järjepidevus [13]. Sarnaselt Reactile oli Ant Design juba muudes ettevõtte rakendustes kasutusel ning seetõttu aitas selle kasutamine tagada koodiliselt ühtset stiili. Ant Design teegi väliste stiilide ja kujunduse määramiseks kasutati CSSi.

Tagarakendusega suhtlemiseks loodi ühendus OpenAPI spetsifikatsiooni abil, mille põhjal genereeriti kliendi API liidesed. See võimaldas mugavalt tekitada päringute tüübid ja andmestruktuurid, vähendades käsitsi kodeerimise vajadust [14]. Andmepäringute tegemiseks ja vahemällu salvestamiseks kasutati Tanstack Query teeki, mis võimaldab hallata asünkroonseid andmeid ning hoida kasutajaliides sujuvalt sünkroonis serveri andmestikuga [15].

### **3.4 Tagarakendus**

Võrreldes eesrakendusega oli tagarakenduse kood vähem mahukas. Tagarakenduse arendamisel kasutati mitmeid tänapäevaseid ja laialdaselt levinud tehnoloogiaid, mille valik lähtus süsteemi lihtsuse, laiendatavuse ja kogukonna toe kriteeriumidest. Sarnaselt eesrakendusele valiti programmeerimiskeeleks TypeScript, et vähendada muude keelte õppimiseks kuluvat aega ning hoida stiil kogu projekti üleselt võimalikult ühtne. Põhiliseks arendusraamistikuks oli Express, mis töötab Node.js keskkonnas. Express võimaldab luua kergekaalulisi ja paindlikke veebirakendusi ning REST-teenuseid, olles samas hästi dokumenteeritud ja toetatud arendajate kogukonna poolt [16]. Node.js tagab serveripoolse JavaScripti käivitamise,

võimaldades TypeScripti kasutada nii ees- kui tagarakenduse arenduses, mis lihtsustab arendustööde haldamist ning suurendab koodi taaskasutatavust [17].

Andmevahetuseks väliste teenuste ja süsteemide vahel kasutati Axios teeki, mis võimaldab mugavalt teostada HTTP-päringuid ja hallata nende vastuseid. Axiose kasutamine lihtsustab API-päringute struktuuri ning pakub tuge veahalduseks ja päringute konfigureerimiseks, muutes tagarakenduse töökindlamaks ja hooldatavamaks [18].

Kokkuvõttes võimaldas valitud tehnoloogiate kooslus luua tagarakenduse, mille arendamine oli sujuv ja kiire, kuna autor sai suures osas tugineda varasematele teadmistele ega pidanud omandama uut programmeerimiskeelt või selle ökosüsteemi. Tehnoloogiate omavaheline sobivus aitas kaasa efektiivsele prototüüpimisele ning lihtsustas integratsiooni teiste süsteemi komponentidega.

### **3.5 Olulised liidestused**

Süsteemi arenduse käigus olid kriitilise tähtsusega mitmed välised teenused, mille liidestamine võimaldas võtta kasutusele olulist funktsionaalsust ilma selle nullist arendamiseta. Selline lähenemine vähendas märgatavalt arendusaega ning võimaldas minimaalse sobiva lahenduseni jõuda kiiremini. Peamised olulised liidestused olid Google'i autentimine, Zitadel identiteedihaldus, Posthog, Stripe ja Salesforce.

#### **3.5.1 Google'i autentimine**

Mugava ja turvalise kasutajakogemuse huvides lahendati registreerimisel autentimine lõputöö skoobis Google'i konto kaudu. See lähenemine võimaldas kasutajate isiksust tõestada konto loomise protsessis ja lõi aluse täielikult paroolivaba sisselogimise toetamiseks tulevikus. Kuigi kasutajakontod luuakse hetkel siiski koos parooliga, suunab valitud lahendus arendust pikaajaliselt selle eesmärgi poole. Kasutatud react-oauth/google teek järgib Google OAuth 2.0 standardeid ning tagab, et autentimisprotsess oleks kooskõlas tänapäevaste turvanõuetega [19].



### **3.5.2 Zitadel identiteedihaldus**

Ettevõtte toodete autentimine oli juba varasemalt tsentraalselt lahendatud Zitadel identiteedihaldustarkvara kaudu, mis tagab ühtse ja turvalise sisselogimiskogemuse kogu süsteemis. Sama lähenemist rakendatakse ka registreerimisrakenduses, kus enne maksehaldussüsteemiga seotud toimingute tegemist peab kasutaja läbima autentimise Zitadel'i kaudu. See võimaldab kasutada ühtset sisselogimismehhanismi, lihtsustab kasutajahaldust ning tagab parema kontrolli kasutaja õiguste ja ligipääsude üle [20].

### **3.5.3 Posthog**

Kasutajategevuse analüüsimiseks ja funktsioonlippude kasutamiseks integreeriti süsteemi Posthog. See rakendus võimaldab koguda teavet näiteks selle kohta, milliseid nuppe klikitakse, milliseid lehti külastatakse ning kui kaua seal viibitakse [21]. Mainitud andmed aitavad tuvastada võimalikke kitsaskohti kasutajaliideses ning toetavad informeeritud otsuste tegemist edasise arenduse kavandamisel. Posthogi kasuks rääkisid kasutusmugavus ja põhjalik dokumentatsioon.

### **3.5.4 Stripe**

Maksehalduse jaoks kasutati rakenduses Stripe lahendust. Seda rakendati lõppkasutaja aadressiandmete esmaseks küsimiseks ning prooviperioodi ja edasise tellimuse haldamiseks. Stripe'i API võimaldas kiiresti üles seada maksehaldusega seotud vood, sealhulgas kasutajakontode automaatse loomise ning seosed tellimuste ja maksetega [22]. Stripe'i valiku kasuks rääkisid selle suur valik isikupärastamise võimalusi ja suhteliselt lihtne integreeritavus.

Kuna maksehalduse lahendus on ettevõtte jaoks ärikriitilise tähtsusega, ei tehtud Stripe'i valikut lõputöö autori poolt üksi. Autor uuris ja võrdles veel mitmeid alternatiive, sealhulgas Chargebee'd ja Revolut'i. Revolut oleks sobinud vaid maksevärava rolli, kuid sel juhul tulnuks kogu tellimuste haldus ja arveldusloogika ettevõttes ise välja arendada [23]. Chargebee pakkus küll valmislahendust tellimuste haldamiseks, kuid kasutas taustal samuti Stripe'i ning tõi kaasa täiendavat keerukust ja kulu [24].

Pärast alternatiivide analüüsi ning arutelu ettevõtte esindajaga otsustati valida Stripe, kuna

see pakub kogu vajalikku funktsionaalsust seoses maksete ning tellimuste haldusega. Ettevõtte tulevane ärimudel näeb ette täpse arvelduse virtuaalse protsessori tuumade kasutuse alusel tunnipõhiselt, mida Stripe suudab samuti tehniliselt ja administratiivselt toetada.

### **3.5.5 Salesforce**

Viimase integratsioonina lisati Salesforce, mis on üks kõige populaarsemaid kliendisuhete halduse (CRM) süsteeme [25]. Salesforce oli ettevõtte siseselt juba varem kasutuses ning selle integreerimine oli vajalik, et võimaldada registreerimisrakenduse kasutamist huviliste ootenimekirja lisamiseks, kuni platvorm jõuab faasi, kus see on küps lõppkasutajatele avaldamiseks.

## 4 Arendus

Arendustöö toimus käsikäes tehnoloogiliste valikute tegemisega, kuna projekti ärinõuded kujunesid välja paralleelselt arendusprotsessiga. Skoop oli seetõttu küllalt dünaamiline ja töö autor osales aktiivselt nõuete täpsustamises, andes tagasisidet tehniliste prototüüpide arendustulemuste põhjal. Iteratiivne lähenemine võimaldas ärinõutel ja tehnilistel lahendustel kujuneda üksteist arvestades ning tagas, et lõplik rakendus vastaks reaalsele vajadusele. Ettevõttepoolne ärianalüüs toimus samaaegselt lõputööga, kuid jäi lõputöö koosseisust välja.

### 4.1 Arenduspõhimõtted

Arendus toimus agiilse arenduse põhimõtteid järgides ühenädalaste sprintide kaupa, mille lõpus osales lõputöö autor koosolekutel ettevõtte ülejäänud tarkvaraarenduse meeskonnaga.

Kogu registreerimissüsteemi kohta loodi ettevõtte projekti haldustarkvaras üks *epic*-tüüpi ülesanne, mille alla loodi vastavalt olukorrale *Task* või *Story* tüüpi alamüksuseid. Ühe ülesande kestus varieerus mõnest tunnist paari nädalani.

Kuigi registreerimisrakenduse arendamisega tegeles lõputöö autor üksi, viidi suurem osa arendusülesandeid läbi eraldi Giti harudes. Enne kui need liideti projekti põhiharusse, viidi läbi koodi ülevaatus mõne teise ettevõttes töötava arendaja poolt, et tagada koodi kvaliteet ja vastavus nõuetele.

Kui oli aeg rakenduse uue versiooni avaldamiseks, siis viidi uued muudatused põhiharusse ning kasutati *CI/CD pipeline*'e, et ehitada valmis uued Dockerhubi konteineri pildid. Kuigi süsteemi ees- ja tagarakendus loodi ühise GitHubi projektina, siis olid need eraldatud *CI/CD pipeline*'i tasandist alates.

## **4.2 Eesrakenduse arendus**

Eesrakenduse arendus keskendus kasutajaliidese loomisele ning selle loogika rakendamisele viisil, mis toetaks sujuvat ja intuitiivset kasutajakogemust. Arenduse käigus pöörati suurt tähelepanu kasutajamugavusele ja turvalisusele. Funktsionaalsuse järk-järgulise täiustamise käigus lisati rakendusele uusi funktsioone ja parandati olemasolevaid. See protsess toimus tihedas arutelus arendusmeeskonna ja kaasjuhendajaga, kes andsid väärtuslikku tagasisidet ja suuniseid. Eesrakenduse loomine oli kõige mahukam osa registreerimissüsteemi arendusprotsessist.

### **4.2.1 Visuaalne prototüüp**

Enne arendustöö alustamist loodi rakenduse visuaalne prototüüp, mille eesmärk oli kavandada süsteemi kasutajaliidese loogika ning välimus. Prototüübi loomine võimaldas enne programmeerimisega alustamist selgelt defineerida, millised funktsionaalsused ja kasutajavood on antud skoobis vajalikud, et rakendus vastaks nii lõputöö eesmärkidele kui ettevõtte ootustele.

Visuaalse prototüübi loomine oli oluline suhtluses ettevõtte esindajaga, kuna see aitas tagada, et töö autor oli ärinõuetest õigesti aru saanud. Selle põhjal sai ettevõtte esindaja anda varakult tagasisidet ning teha vajadusel täpsustusi või muudatusi enne, kui hakati süsteemi arendama. See vähendas riski, et hilisem arendus ei vasta ettevõtte ootustele, ja aitas kiirendada kogu tööprotsessi.

Prototüübi loomisel oli oluline, et see oleks loodud kasutades Ant Design teegi visuaalseid komponente ja kujunduspõhimõtteid, et tagada ühtivus valitud tehnoloogiatega ning järjepidevus ettevõtte toodete kasutajaliideste disainis. Lisaks võimaldas selline lähenemine arvestada kohe kasutatava teegi piirangute ja võimalustega.

Tänu prototüübile oli võimalik ennetada mitmeid kasutusloogikaga seotud probleeme ning tagada, et loodav rakendus oleks intuitiivne, funktsionaalne ja visuaalselt kooskõlas ettevõtte ootuste ning olemasoleva kasutajaliidese stiiliga.

## 4.2.2 Struktuur

Eesrakenduse kaustastruktuur on kujutatud joonisel 2. Selle selge ülesehitus aitab hoida koodi korrastatuna ja arendustööd paremini hallata. Struktuur on jagatud loogilisteks üksusteks vastavalt funktsionaalsusele ja vastutusalale.

Üks olulisemaid kaustu on “providers”, kuhu on koondatud erinevad konteksti pakkujad. Need võimaldavad jagada valitud loogikat või andmeid rakenduse erinevate komponentide vahel, pakkudes tsentraalseid lahendusi nagu autentimisloogika või seadistused. Konteksti kasutamine aitab vältida liigset andmete edasiandmist komponentide kaudu ning toetab puhtamat ja taaskasutatavat koodi.

Kaustas “layouts” asuvad rakenduse üldised kujundusmallid ja korduvkasutatavad struktuurielemendid. Need komponendid määravad ära vaadete üldise ülesehituse ning tagavad visuaalse ja funktsionaalse järjepidevuse kogu rakenduses.

“Views” kaust sisaldab spetsiifilisi vaateid ehk lehti, kus on ühendatud kujunduselemendid, unikaalsed kasutajaliidese komponendid ning vaatepõhine andmepäringute loogika. See struktuur võimaldab iga vaate loomisel keskenduda oma funktsionaalsusele, hoides samal ajal üldise arhitektuuri puhtana.

Kaustas “Routes” on kirjeldatud vaadetele vastavad marsruudid ja aadressipõhine loogika. Siin määratletakse, milline vaade kuvatakse konkreetse URL’i korral ning kuidas toimub navigeerimine rakenduses. “Routes” kaust töötab tihedalt koos “RouterProvider” failiga, kus pannakse kokku rakenduse ülene aadressipuu.

- frontend
    - dist
    - node\_modules
    - openapi
    - public
    - scripts
    - src
      - assets
      - commons
      - components
      - icons
      - layouts
      - providers
        - AntdConfigProvider
        - ApisProvider
        - AuthProvider
        - ConfigProvider
        - GlobalContextProvider
        - OidcProvider
        - QueryClientProvider
        - TrackingProvider
        - ZitadelIdentityProvider
      - RouterProvider.tsx
    - routes
    - views
      - AccountInformationView
      - CustomerFormView
      - EverythingIsSetView
      - SignUpLandingView
    - LoginView.tsx
    - index.tsx
  - TS vite-env.d.ts
  - .env.local
  - Dockerfile
  - eslint.config.js
  - index.html
  - nginx.conf
  - package.json
  - README.md
  - tsconfig.json
  - tsconfig.node.json
  - vite.config.ts

Joonis 2. Eesrakenduse struktuur.

### **4.2.3 Identiteedi terviklikkuse tagamine ja haldus**

Kuna registreerimisrakenduses töödeldakse isikuandmeid ning hallatakse maksehalduse loogikat, oli turvalisuse tagamine kriitilise tähtsusega. Veel oli oluline pakkuda kasutajatele sujuvat ja intuitiivset registreerimise kogemust, järgides samaaegselt ettevõttepoolseid nõudeid identiteedihalduse osas. Selleks on autentimisprotsess üles ehitatud kahe teenusepakkuja teenustele: esmalt autendib kasutaja end Google'i konto kaudu, mille järel, registreerimisprotsessi lõpus, luuakse talle automaatselt Zitadel'i konto. Viimane on vajalik, et tagada kasutajale juurdepääs kõigile ettevõtte ressurssidele läbi ühtse sisselogimissüsteemi.

Mõlema teenusepakkuja puhul on rakenduses järgitud kõrgeid turvastandardeid ehk kasutajaandmeid ei saadeta kunagi lihttekstina eesrakendusest tagarakendusse. Selle asemel vahendab eesrakendus ainult turvalisi ligipääsu tokeneid, mille alusel saab tagarakendus otse teenusepakkujalt pärida vajalikke andmeid kasutaja kohta. Selline lahendus vähendab andmete lekkimise riski ning tagab, et kasutajainfo käsitlemine oleks kooskõlas kaasaegsete turvapraktikatega.

## **4.3 Tagarakenduse arendus**

Tagarakenduse arendus toimus paralleelselt eesrakenduse loomisega, sest mõlema komponendi omavaheline koostöö oli süsteemi toimimise seisukohalt kriitilise tähtsusega. Arendusprotsessis oli oluline, et mõlemad pooled areneksid samaaegselt ja kooskõlas, võimaldades testida ning kinnistada andmevahetust ja funktsionaalsust juba varajases staadiumis. Tagarakenduse roll ei olnud üksnes eesrakenduse teenindamine, vaid see täitis ka kesket rolli kogu süsteemi turvalisuse ja andmete usaldusväärsuse tagamisel.

Peamiseks eesmärgiks tagarakenduse arendamisel oli ehitada kindel ja turvaline sild eesrakenduse ning erinevate liidestatud teenuste vahel, sealhulgas autentimisteenused, makselahendused ja ettevõttesisesed süsteemid. Arvestades, et eesrakendus töötab kliendi seadmes (näiteks veebibrauseris), kus turvalisuse tagamine on keerulisem ning kasutaja saab muuta saatmisele minevat andmestikku, oli oluline viia tundlike andmete käsitlemine serveripoolsesse keskkonda.

Turvalisuse tagamiseks kujundati süsteem selliselt, et tagarakendusest ei oleks võimalik

kasutajaandmeid pärida lihttekstilise parameetri, näiteks e-posti aadressi alusel. Selle asemel kasutati turvalisi autentimismehhanisme, kus edastati üksnes ligipääsutokenid, näiteks OAuth 2.0 standardi kohaseid autoriseerimistokenid. Need võimaldasid tagarakendusel tuvastada kasutaja isikusamasuse ja seejärel ise turvaliselt vajalikke andmeid kolmandate osapoolte nagu identiteedi- või maksehaldusteenustest teenustest pärida, töödelda ning vajadusele vastavalt teisele liidestusele edastada.

See lähenemine toob mitmeid eeliseid. Esiteks väheneb oht, et pahatahtlik kasutaja saadab süsteemi võltsitud või muul moel manipuleeritud andmeid, kuna kõik tundlikud päringud toimuvad serveri poolsest kontrollitud keskkonnas. Teiseks aitab see säilitada süsteemi loogilise ülesehituse: eesrakendus keskendub kasutajaliidesele ja kasutajakogemusele, samas kui tagarakendus vastutab äriloomika ja liidestuvate süsteemide autentsuse ning andmete tervikluse valideerimise eest.

Üldjoontes ei suurenda kirjeldatud jaotus mitte ainult süsteemi töökindlust ja turvalisust, vaid toetab ka selle hilisemat skaleerimist ja hooldamist. Kuna autentimise ja andmepäringute loogika on koondatud ühte keskpunkti, on muudatuste tegemine lihtsam ja vähem hajutatud.

## **4.4 Koodi väljastamine**

Pärast koodi uue versiooni valmimist integreeriti see projekti põhiharusse, mis käivitas GitHub Actions'is eelnevalt seadistatud CI/CD töövoog. See töövoog lõi uue DockerHub'i konteineri pildi, mille silt (*tag*) tähistas konkreetset versiooni ning pakkus võimalust hallata erinevaid rakenduse seisundeid usaldusväärselt ja korduvkasutatavalt.

Seejärel kasutas lõputöö autor loodud DockerHub'i silti koos vastavate keskkonnamuutujatega, et paigaldada rakendus Kubernetes'e klastrisse, mis asus Amazon Web Services pilvekeskkonnas. Selline lahendus võimaldas rakenduse kättesaadavaks teha testimiseks mõeldud demo-keskkonnas.

Peale demo-keskkonnas funktsionaalsuse kontrollimist, viidi sarnast töövoogu järgides rakenduse uus versioon ka tootmiskeskonda. See tagas, et kõik keskkonnad olid hallatud ühtse ja turvalise protsessi abil, kasutades kaasaegseid arenduse põhimõtteid ning tööriistu.



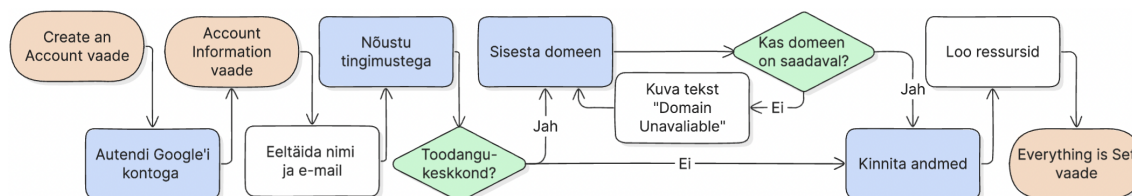
## 5 Tulemused, tagasiside ja edasised arendusvõimalused

Lõputöö arendustöö tulemusel valmis Entigo Platvormi jaoks registreerimisrakenduse esimene iteratsioon [26], mis on loodud vastama nii ettevõtte vajadustele kui lõppkasutaja mugavusele. Rakendus koosneb kahest põhilisest komponendist: eesrakendusest ehk kasutajaliidesest ning tagarakendusest ehk serveripoolsest loogikast. Lahendus sisaldab kahte eraldi voogu, esimene keskendub uue kasutaja registreerimisele ning teine maksehalduse protsessidele. Selline jaotus võimaldab funktsionaalsust paremini hallata, tagades samas sujuva kasutajakogemuse ning süsteemi loogilise ülesehituse. Arendusprotsessi käigus pöörati erilist tähelepanu arhitektuurilisele selgusele, turvalisusele ning tuleviku kaalutlustele, et loodud rakendus oleks jätkusuutlik platvormi arenedes.

### 5.1 Vood

Registreerimissüsteem koosneb kahest voost, mis ei ole mõeldud vahetult üksteisele järgnema, kuid mida on võimalik selliselt läbida. Esimese voo tulemusena loob kasutaja omale konto ning alustab tasuta katseperioodi tootega. Teise voo tulemusena sisestab kasutaja puuduvad makseandmed ning alustab soovi korral toote tasulise kasutamisega. Sama voogu on võimalik kasutada ka hilisemaks makseandmete haldamiseks. Järgnevates alapeatükkides on kirjeldatud mõlemat voogu põhjalikumalt.

#### 5.1.1 Registreerimisvoog



Joonis 3. Registreerimisvoog.

Joonisel 3 on esitatud registreerimisvoo põhimõte. Oranžiga on märgitud erinevad vaated, rohelisega otsustuskohad, kust voog hargneb, sinisega kasutaja tehtavad tegevused ja

valgega süsteemi tehtavad tegevused.

Järgnevalt on kirjeldatud registreerimisvoo tööpõhimõte:

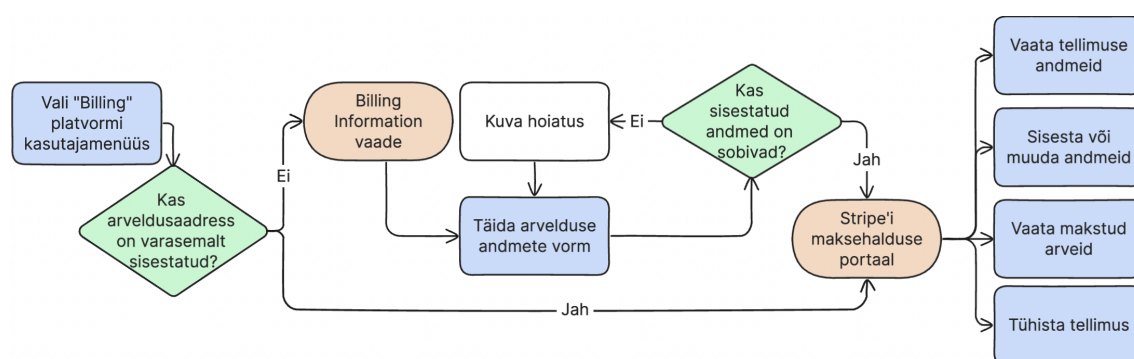
1. Registreerimisprotsess algab lehega „*Create an Account*“ (joonis 7), kus kasutajal palutakse valida endale sobiv autentimisviis. Esialgses versioonis on kasutusel ainult Google SSO-põhine sisselogimine.
2. Google SSO valimisel avaneb hüpikaken, mille kaudu toimub autentimine kasutaja Google'i kontoga.
3. Kui autentimine õnnestub, suunatakse kasutaja järgmisele lehele pealkirjaga „*Account information*“ (joonis 8). Seal kuvatakse kasutajale eeltäidetud kujul tema nimi ja e-posti aadress, mis on saadud autentimisteenusest. Nende andmete muutmine pole võimalik, kuna tegemist on süsteemis juba verifitseeritud informatsiooniga.
4. Samas vaates palutakse kasutajal täita domeeni väli, mis on unikaalne identifikaator, mida kasutatakse hiljem ressursside loomisel. Süsteem kontrollib automaatselt domeeni saadavust reaalajas, iga sisestatud sümboli lisamisel või eemaldamisel. Lisaks tuleb kasutajal nõustuda platvormi kasutustingimustega.
5. Kui kasutaja vajutab nuppu „*Confirm*“, algatatakse mitmete seotud ressursside loomine. Tema jaoks luuakse Stripe'i kliendi objekt, millega kaasneb automaatselt tasuta prooviperiood. Samuti genereeritakse Kuberneteses tööruum (*workspace*) ning Zitadel identiteedihalduses organisatsioon, millede nimetuseks kasutatakse sisestatud domeeni. Lisaks saadetakse kasutajale e-kiri, mille kaudu ta saab lõpule viia oma Zitadel'i konto loomise.
6. Protsessi lõpetuseks kuvatakse kasutajale kinnitav informatiivne vaade „*Everything is set*“ (joonis 9), mis teavitab, et ressursid on edukalt loodud. Lehel antakse kasutajale juhis kontrollida oma elektroonset postkasti, et lõpule viia konto seadistamine ja alustada platvormi kasutamist.

Registreerimissüsteemi valmimise ajaks ei olnud Entigo Platvorm veel piisavalt küps, et võimaldada lõppkasutajatel teenust realselt kasutama hakata. Seetõttu suunati rakenduse funktsionaalsus esialgu huviliste ülesmärkimisele ning nende lisamisele ootenimekirja. Selleks lisati registreerimisrakendusse keskkonnamuutuja, mis määras, millist registreerimisvoo versiooni parasjagu kuvada. Lisaks rakendati Posthog liidestuse funktsioonilippude funktsionaalsust, mis võimaldas üksikutele kasutajatele lisada lipukesi ning kontrollida,

kellele anda juurdepääs registreerimisrakenduse toodangukeskkonnale. Seeläbi oli võimalik teenuse ligipääs esialgu tagada vaid valitud usaldusväärsetele test-kasutajatele.

Toodanguelse voo eripäraks oli see, et domeeni sisestamise samm jäi vahele ning tehniliste ressursside loomise asemel lisati registreerimisprotsessi edukalt läbinud kasutaja ettevõtte Salesforce'i andmebaasi müügivihjena (*lead*). See võimaldas ettevõttel jätkata huvilistega suhtlust kuni hetkeni, mil platvormi arendustase võimaldas neile teenuse reaalselt kättesaadavaks teha.

### 5.1.2 Maksehaldusvoog



Joonis 4. Maksehaldusvoog.

Joonisel 4 on esitatud maksehaldusvoo põhimõte. Oranžiga on märgitud erinevad vaated, rohelisega otsustuskohad, kust voog hargneb, sinisega kasutaja tehtavad tegevused ja valgega süsteemi tehtavad tegevused.

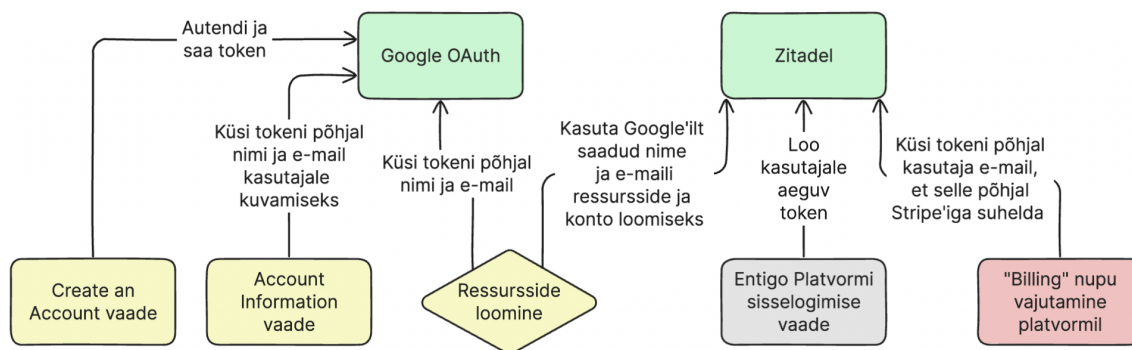
Alljärgnevalt on esitatud maksehalduse töövoo põhimõte:

1. Maksehaldusprotsess käivitub, kui kasutaja klikib Entigo Platvormi sisselogituna kasutajamenüüs valikule „Billing“ (joonis 10).
2. Kui kasutaja kasutab maksehaldust esimest korda või pole varasemalt sisestanud oma arveldusega seotud aadressiandmeid, suunatakse ta esmalt vastavaid andmeid küsivale vormile „Billing information“ (joonis 11). Seal tuleb sisestada järgmised andmed:
  - täisnimi
  - arveldusaadress
  - telefoninumber koos vastava riigikoodiga

Andmete korrektseks täitmiseks suunatakse kasutaja edasi Stripe'i maksehalduse portaali.

3. Kui kasutaja on varasemalt aadressiandmed juba salvestanud, siis suunatakse ta „Billing” nupule vajutades otse Stripe'i portaali.
4. Stripe'i maksehalduse keskkonnas (joonis 12) on kasutajal võimalik hallata kõiki maksetega seotud andmeid. Portaali funktsionaalsus võimaldab:
  - lisada ja eemaldada maksemeetodeid
  - hallata teenuse paketti
  - uuendada isikuandmeid
  - sirvida varasemate maksete ajalugu

## 5.2 Turvalisus



Joonis 5. Autentimise ja valideerimise etapid.

Autentimise ja valideerimise etappide joonisel on registreerimisvoo etapid tähistatud kollasega, maksehaldusega seotud tegevused punasega ning töö skoobist väljapoole jäävad protsessid hallina. Autentimisteenuste pakkujad on märgitud rohelisega. Joonisel kuvatud tegevused ja vaated järjestuvad kasutaja töövoos vasakult paremale.

Lõputöö raames rakendati kahte autentimismeetodit: Google'it ja Zitadel'i. Kuna ettevõtte eesmärk oli vältida käsitsi e-maili valideerimist ja liikuda sujuvama kasutuskogemuse poole, oli vajalik lahendus, kus süsteem suudaks tagada Google'ilt saadud e-posti aadressi autentsuse ja tervikluse ehk kasutajal ei tohi olla võimalust seda protsessi käigus muuta. Seetõttu küsis tagarakendus otse Google'ilt emaili aadressi ligipääsutokeni alusel, mitte ei tuginenud eesrakendusest edastatud andmetele, mis võivad olla muudetud või ebausaldusväärsed.

Google OAuth'iga suhtleb süsteem „*Create an Account*“ vaates kasutaja autentimiseks ja tokeni saamiseks. Saadud token salvestatakse eesrakendusse. Enne „*Account Information*“ vaate kuvamist küsib tagarakendus Google'i käest tokeni põhjal kasutaja nime ja e-maili selleks, et kuvada Zitadel'i konto loomisel kasutatavad andmed. Registreerimisvoo lõpus, kui luuakse seotud ressursid, tehakse Google'ile samasisuline päring uuesti. See võimaldab tagarakendusel jääda olekuta ehk mitte talletada kasutajaandmeid sessiooni vältel. Antud lähenemine oli väikese kasutajamahu juures lihtsaim ja kuluefektiivseim olekuvabaduse eesmärgi täitmiseks.

Kui kasutaja on Zitadel'i konto loomisega lõpuni jõudnud ja logib Entigo Platvormile sisse, siis väljastatakse talle aeguv Zitadel'i token, mis salvestatakse platvormi eesrakendusse. Sisselogimise loogika ei kuulunud lõputöö skoopi. Kui kasutaja soovib ligi pääseda Stripe'i maksehalduse portaalile, siis saadetakse token registreerimisrakenduse tagarakendusse ja tokeni põhjal küsitakse Zitadel'i käest kasutaja e-maili aadress, mille põhjal on võimalik koos salajaste Stripe'i võtmetega sealse API käest kasutaja andmeid küsida ja kasutajale maksehalduse portaali sessiooni URL genereerida.

### **5.3 Testimine**

Lõputöö fookus oli kasutajaliidese arendamisel ja väliste teekide liidestamisel, mistõttu keskendus testimine peamiselt funktsionaalsuse ja kasutajakogemuse valideerimisele koostöös ettevõttega. Arendusprotsessi vältel ei rakendatud automaatsete kirjutamist. Selle otsuse tegemisel lähtuti töö piiratud ajaraamist, arendusprojekti ulatusest ning asjaolust, et töö eesmärk oli prototüüpse rakenduse loomine, mille lõplik tootestamine toimub hilisemas etapis ettevõtte sisemiste arendusprotsesside raames.

Peamise testimismeetodina kasutati manuaalset testimist arenduse käigus ning töö lõppfaasis viidi ettevõtte poolt läbi kasutajatestimine. Testimisel keskenduti eelkõige kriitiliste kasutusvoogude kontrollimisele ja funktsionaalsuse vastavusele ettevõtte ootustega. Testimise korraldus ja ulatus olid kooskõlas prototüübi olemuse ja töö eesmärkidega ning moodustasid osa praktilisest koostööst ettevõttega.

## 5.4 Ettevõtte tagasiside

Lõputöö koostamise ja arendusprotsessi vältel tehti tihedat koostööd ettevõtte esindajaga, kes andis jooksvalt sisulist ja tehnilist tagasisidet arenduse edenemise kohta. Regulaarne suhtlus võimaldas vajadusel lahendust kohandada ning tagas selle, et valmiv rakendus vastaks ettevõtte ootustele.

Pärast töö valmimist viis ettevõtte läbi kasutajatestimise, mille käigus ilmnes üks varrem arvestamata olukord: võimalus alustada sama kasutaja poolt registreerimisprotsessi mitu korda viis süsteemi tõrkeni. Ülejäänud testimisel ei ilmnunud olulisi tehnilisi ega kasutajakogemusega seotud puudujääke, mis viitab süsteemi üldisele töökindlusele ja kasutajasõbralikkusele.

Registreerimisvoos tuvastatud probleem lahendati, rakendades kontrollmehhanismi, mis tuvastab, kas autenditud e-posti aadressiga on Zitadel'i süsteemis juba olemas kasutajakonto. Selle teabe alusel suunatakse kasutaja kas registreerimisprotsessiga edasi või teavitatakse teda olukorrast hoiatusteate kaudu.

Lahenduse esialgsel kavandamisel tõstatus murekoht, et kontode olemasolu kontrollimise funktsionaalsus võiks teoreetiliselt võimaldada kasutajakontode kaardistamist. Antud risk ei realiseerunud, kuna sarnaselt ülejäänud tundlikule funktsionaalsusele on konto olemasolu võimalik kontrollida üksnes kontole vastava kehtiva ligipääsutokeni alusel: autentimise järel tehakse päring tagarakendusse, mille päises on Google'i ligipääsutoken. Selle põhjal tehakse päring Google'ile e-maili saamiseks. Saadud e-maili abil tehakse päring platvormi tagarakenduse API otspunktile, mis on rakenduse väliselt ligipääsmatu. Selline piirang tagab, et säilib süsteemi turvalisus ja kasutajate privaatsus.

Ettevõtte tagasiside põhjal võib järeldada, et arendusprotsess oli tulemuslik, ning valitud tehnoloogiate ja tööviiside kombinatsioon võimaldas luua hästi toimiva ja ootustele vastava lahenduse.

## 5.5 Analüüs

Järgnevates alapeatükkides on kirjeldatud projekti arendusprotsessi ja analüüsitud täidetud ning täitmata jäänud nõudeid.

### 5.5.1 Projekti arendusprotsess

Projekti tehnilise osa arendusprotsess toimus 5 kuu jooksul. Selle perioodi jooksul osales lõputöö autor aktiivselt Entigo tootearendusmeeskonna koosolekutel ning suhtles ettevõtte meeskonnaliikmetega, et õppida sisemisi süsteeme paremini tundma registreerimisrakenduse nendega integreerimise eesmärgil.

Lõputöö ülikooli poolseks juhendajaks oli Siim Rebane, kes pakkus nõu peamiselt lõputöö kirjutamise osas. Tehniliste küsimuste ja nõuete täpsustamise jaoks tehti koostööd ettevõttepoolse kaasjuhendaja Rein Remmeliga. Arenduse vältel viidi regulaarselt läbi töö edenemist kajastavaid demosid nii ülikooli- kui ka ettevõttepoolsele juhendajale.

### 5.5.2 Nõuete täitmine ja analüüs

Lõputöö käigus väljatöötatud lahendus vastas suures osas esialgselt määratletud funktsionaalsetele ja mittefunktsionaalsetele nõuetele. Registreerimis- ja maksehaldusvoogude toimimine sai terviklikult realiseeritud: kasutaja saab iseteeninduslikult luua konto ja selle kasutamist alustada ning hiljem maksehaldusega tegeleda.

Funktsionaalsete nõuete seast osutus üheks keerukamaks lahendada kasutaja kõigi kontaktandmete esitamise kohustuslikuks muutmine enne maksemeetodi sisestamist. Arenduse käigus katsetati mitut erinevat Stripe'i poolt pakutavat maksehaldust toetavat voogu, kuid lõplikus lahenduses kasutusele võetu osutus ainsaks, mis võimaldas korraga täita nii kontaktandmete kogumise nõude kui toetada ettevõtte poolt planeeritavat kasutuspõhist teenuse hinnastamist.

Mittefunktsionaalsete nõuete osas pöörati erilist tähelepanu turvalisusele ja süsteemi hooldatavusele. Kasutajaandmete edastamisel on täielikult välditud lihtteksti saatmine ees- ja tagarakenduse vahelises suhtluses. Rakenduse juurutusprotsess on automatiseeritud GitHub Actions *CI/CD pipeline*'i kaudu ning rakendus on paigaldatud konteineripõhiselt Kubernetese klastrisse Amazon Web Services infrastruktuuris. Kasutatakse ka jälgitavuse ja funktsioonide testimise tööriistu nagu funktsioonilipud, mis võimaldavad kontrollitult uusi funktsionaalsusi kasutajatele kättesaadavaks teha.

Vaatamata laialdasele nõuete täitmisele jäi lahenduses ühe olulise aspektina puudu tõrke-

taluvuse täielik realiseerimine. Nimelt ei suuda süsteem hetkel automaatselt toime tulla liidestatud teenuste ajutiste tõrgetega. Kui mingi süsteemi osa ei vasta või esineb tehniline katkestus, siis peatub kasutaja konto ettevalmistamise voog ja kasutajal ei võimaldata järgmise sammu juurde liikuda. Praegune versioon ei proovi ebaõnnestunud päringuid taustal automaatselt uuesti, mis võib viia olukorrani, kus potentsiaalne klient ei saa registreerimisprotsessi lõpuni viia. See tähendab, et näiteks reklaamikampaania kaudu huvi üles näidanud kasutaja ei jõua teenuse kasutamiseni. Sellest tulenevalt kujutab antud piirang endast ettevõtte jaoks ka reaalset äririskiga seotud puudujääki, mida tuleks esimeses järjekorras lahendada edasises arendustsükklis.

Kokkuvõttes on loodud registreerimis- ja maksesüsteemi prototüüp, millel on tugev vundament ja mis vastab suurele osale ärilistest ja tehnilistest ootustest. Edasised täiustused peaksid keskenduma süsteemi vastupidavuse ja kasutusmugavuse suurendamisele ebatavalistes olukordades, et tagada toote usaldusväärne ja skaleeritav kasutuselevõtt. Peamine põhjus nõuete täitmata jätmiseks oli ajaline piirang.

## **5.6 Edasised arendusvõimalused**

Arendatud süsteemile on võimalik rakendada mitmesuguseid tehnoloogilisi täiustusi, mis varieeruvad nii mahu kui keerukuse poolest. Järgnevalt on loetletud mõned ideed kasutajakogemuse parandamiseks, lisaks täitmata jäänud nõuete realiseerimisele.

Hetkel võimaldab rakenduse registreerimisvoog autentimist vaid Google SSO kaudu, mistõttu üheks esimeseks planeeritud edasiarenduseks on lisada tugi GitHubi ning kasutajanime ja parooliga registreerimiseks ja sisselogimiseks.

Teise olulise täiendusena võiks Entigo Platvormile integreerida teavitussüsteemi, mis võimaldaks registreerimissüsteemi tagarakendusel edastada kasutajale informatsiooni maksemeetodi kehtivuse lõppemise või muude Stripe'i keskkonnas esinenud tõrgete kohta. See suurendaks läbipaistvust ja vähendaks makseprobleemidest tulenevat segadust.

Kolmanda edasiarendusena oleks otstarbekas rakendada mitmekeelne kasutajaliides, et toetada rahvusvahelist kasutajaskonda ning lihtsustada süsteemi kasutuselevõttu erinevates keelekeskkondades.



Lisavõimalusena võiks kasutajaliidesesse tuua teemade toe, võimaldades kasutajatel vahetada näiteks hele- ja tumerežiimi vahel, mis aitaks registreerimisrakenduse ja platvormi visuaalset ilmet paremini ühtlustada lõppkasutaja seadme üldise stiiliga.

Lõpuks oleks kasulik lisada rakendusse kasutajatoega otse suhtluse võimalus, mille kaudu saaksid kasutajad pöörduda ettevõtte esindaja poole juhul, kui mõni registreerimisvoo samm jääb arusaamatuks või tekib tehniline tõrge. See aitaks vältida potentsiaalsete klientide kaotamist süsteemivigade korral.

## 6 Kokkuvõte

Käesoleva lõputöö eesmärk oli välja töötada registreerimis- ja maksehalduse süsteem Entigo Platvormi jaoks. Töö eesmärgid ja nõuded sõnastati koostöös Entigo OÜ esindajaga, tuginedes tegelikele äri vajadustele. Arendusprotsessis lähtuti eeldusest, et alustaval ettevõttel ei ole otstarbekas arendada ise keerulisi alamsüsteeme kasutajate või maksete haldamiseks. Seetõttu keskendus töö olemasolevate usaldusväärsete teenuste integratsioonile vastavates valdkondades ning nende põhjal sobiva lahenduse väljatöötamisele.

Töö käigus defineeritud funktsionaalsed ja mittefunktsionaalsed nõuded täideti suures ulatuses. Rakendus on üles ehitatud arvestades turvalisuse, skaleeritavuse ja modulaarse arhitektuuri põhimõtteid ning juurutatud Amazon Web Services pilveteenuses Kubernetes klasteri kaudu.

Olulise parendusvaldkonnana jäi lõputöö ajalise piirangu tõttu realiseerimata süsteemi tõrketaluvus ehk võimekus jätkata registreerimisprotsessiga ka juhul, kui mõni liidestatud teenus ajutiselt ebaõnnestub. Praegune lahendus eeldab, et kõik päringud õnnestuvad esimese katsega. Edaspidi tuleks arendada süsteemi selliselt, et see suudaks automaatselt ebaõnnestunud toiminguid korrata või pakkuda alternatiivseid lahendusi, et vältida ärilisi kaotusi ja parandada kasutajakogemust.

Valminud rakenduse täiustamisega kavatseb töö autor jätkata. Lähitulevikus on plaanis tegeleda tõrketaluvuse funktsionaalsusega, lisada registreerimise voogu rohkem autentimisviise ning arendada Entigo Platvormile teavituste funktsionaalsus.

## Kasutatud kirjandus

- [1] Eesti Riigi Info- ja Kommunikatsioonitehnoloogia Keskus. *Tarkvara kui teenus*. [Kasutatud: 03.02.2025]. URL: <https://riigipilv.ee/teenused/tarkvara-kui-teenus>.
- [2] Arsenii Panov. *Subscription and payment systems for SaaS applications*. [Kasutatud: 03.02.2025]. URL: <https://www.theseus.fi/handle/10024/751313>.
- [3] Entigo. *Entigo Platform*. [Kasutatud: 21.04.2025]. URL: <https://www.entigo.com/platform>.
- [4] Joe Stech. *Why Outsource Your Auth System And How To Sell The Decision To Your Company*. [Kasutatud: 24.04.2025]. URL: <https://fusionauth.io/articles/identity-basics/outsource-auth-system-blueprint>.
- [5] Marty Cagan. *Inspired: How to Create Tech Products Customers Love*. John Wiley ja Sons Inc, 2017. ISBN: 978-1119387503.
- [6] Eric Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional, 2004. ISBN: 978-0-321-12521-7.
- [7] Github. *Features*. [Kasutatud: 21.04.2025]. URL: <https://github.com/features>.
- [8] GitHub. *Understanding GitHub Actions*. [Kasutatud: 21.04.2025]. URL: <https://docs.github.com/en/actions/about-github-actions/understanding-github-actions>.
- [9] Atlassian. *Project management features for all teams*. [Kasutatud: 22.04.2025]. URL: <https://www.atlassian.com/software/jira/features>.
- [10] Milit Panchasara. *TypeScript vs JavaScript: Know The Difference*. [Kasutatud: 21.04.2025]. URL: <https://radixweb.com/blog/typescript-vs-javascript>.
- [11] JetBrains. *JavaScript*. [Kasutatud: 21.04.2025]. URL: <https://www.jetbrains.com/lp/devecosystem-2023/javascript/>.
- [12] Meta. *React*. [Kasutatud: 21.04.2025]. URL: <https://react.dev/>.
- [13] Ant Design. *Design Values*. [Kasutatud: 21.04.2025]. URL: <https://ant.design/docs/spec/values>.
- [14] SmartBear Software. *OpenAPI Specification*. [Kasutatud: 21.04.2025]. URL: <https://swagger.io/specification>.
- [15] TanStack. *TanStack Query: React Overview*. [Kasutatud: 21.04.2025]. URL: <https://tanstack.com/query/latest/docs/framework/react/overview>.
- [16] Express.js. *Fast, Unopinionated, Minimalist Web Framework for Node.js*. [Kasutatud: 22.04.2025]. URL: <https://expressjs.com/>.

- [17] OpenJS Foundation. *About Node.js*. [Kasutatud: 22.04.2025]. URL: <https://nodejs.org/en/about>.
- [18] Axios. *Getting Started*. [Kasutatud: 22.04.2025]. URL: <https://axios-http.com/docs/intro>.
- [19] Mo'men Sherif. *React OAuth2 | Google*. [Kasutatud: 22.04.2025]. URL: <https://www.npmjs.com/package/@react-oauth/google>.
- [20] ZITADEL. *Use Cases*. [Kasutatud: 23.04.2025]. URL: <https://zitadel.com/usecases>.
- [21] PostHog. *Why PostHog?* [Kasutatud: 22.04.2025]. URL: <https://posthog.com/why>.
- [22] Stripe. *One platform for all your payments needs*. [Kasutatud: 22.04.2025]. URL: <https://stripe.com/en-se/payments/features>.
- [23] Revolut. *Set up a subscription management process*. [Kasutatud: 02.06.2025]. URL: <https://developer.revolut.com/docs/guides/accept-payments/tutorials/save-and-charge-payment-methods/subscription-management>.
- [24] Chargebee. *Stripe*. [Kasutatud: 02.06.2025]. URL: <https://www.chargebee.com/docs/payments/2.0/payment-gateways-and-configuration/stripe>.
- [25] Salesforce. *Get the nr 1 CRM with AI, data, and trust built in*. [Kasutatud: 22.04.2025]. URL: <https://www.salesforce.com/crm/worlds-number-one-crm/>.
- [26] Entigo. *Create an Account*. [Kasutatud: 24.04.2025]. URL: <https://app.entigo.com/sign-up>.

## **Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks<sup>1</sup>**

Mina, Rasmus Vananurm

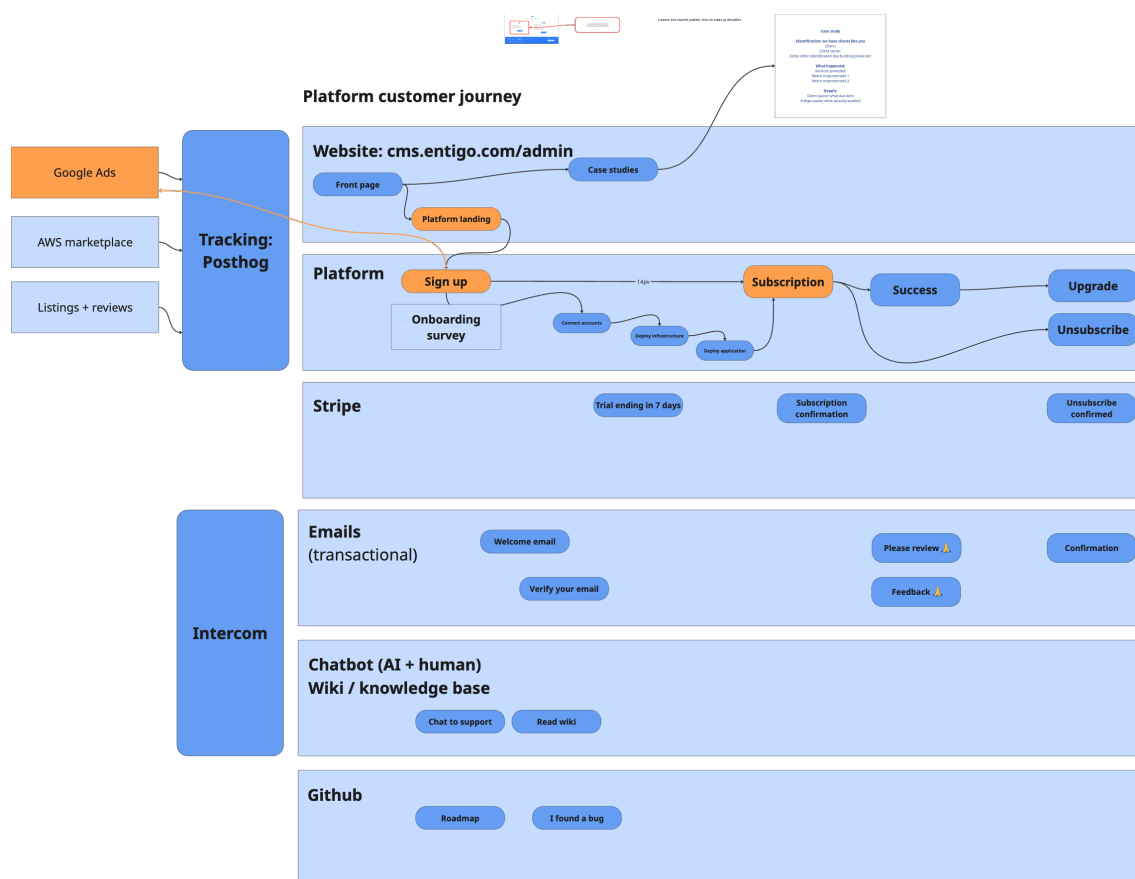
1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose “Ise-teeninduslik registreerimisrakendus Entigo Platvormi jaoks”, mille juhendajad on Siim Rebane ja Rein Remmel
  - 1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
  - 1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskonna 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.

04.06.2025

---

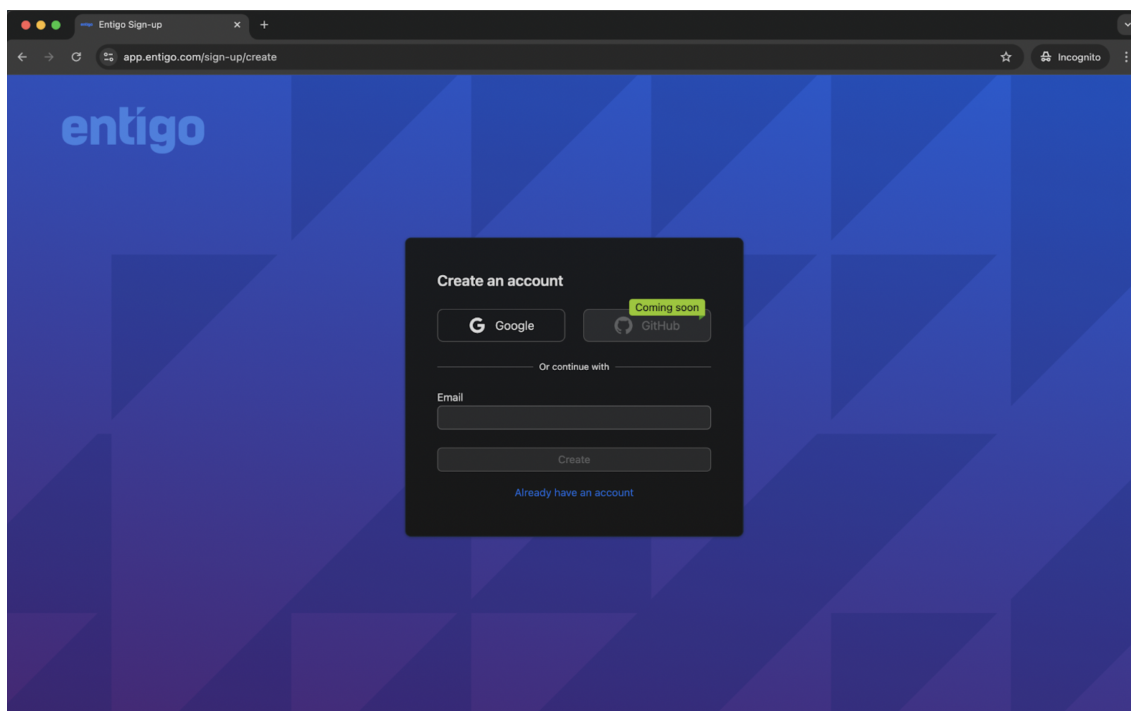
<sup>1</sup>Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud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

## Lisa 2 – Entigo Platvormi kasutajateekond



Joonis 6. Entigo Platvormi kasutajateekond.

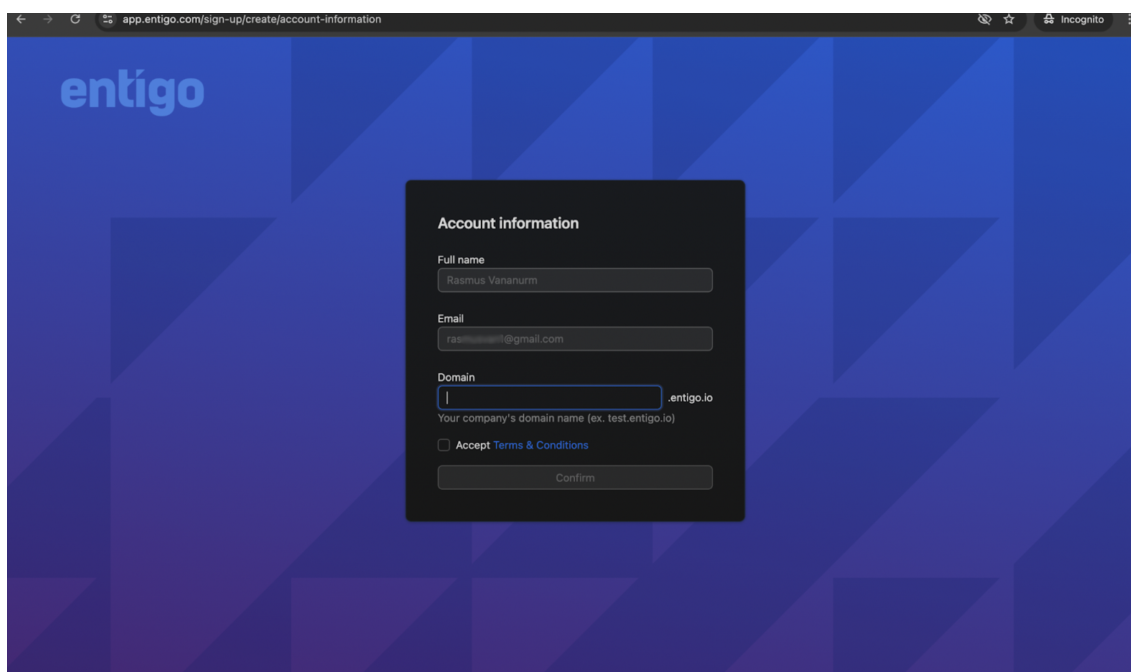
## Lisa 3 – Registreerimisvoo vaated



The screenshot shows a web browser window with the URL `app.entigo.com/sign-up/create`. The page features a dark blue background with a geometric pattern. In the top left corner, the 'entigo' logo is displayed. A central dark modal box titled 'Create an account' contains the following elements:

- Buttons for 'Google' and 'GitHub'. The 'GitHub' button has a yellow 'Coming soon' label above it.
- A link 'Or continue with'.
- An 'Email' input field.
- A 'Create' button.
- A link 'Already have an account' at the bottom.

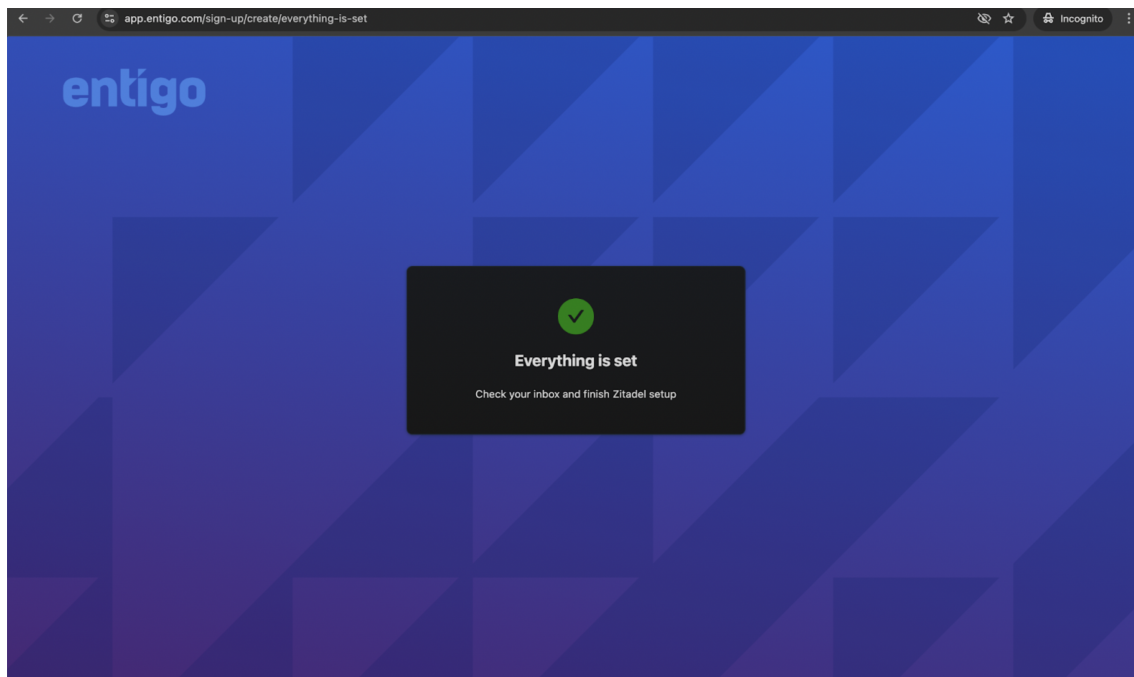
Joonis 7. Registreerimise voo avaleht



The screenshot shows a web browser window with the URL `app.entigo.com/sign-up/create/account-information`. The page features the same dark blue background and 'entigo' logo as the previous screen. A central dark modal box titled 'Account information' contains the following elements:

- A 'Full name' input field with the value 'Rasmus Vananurm'.
- An 'Email' input field with the value 'rasmusvan@ gmail.com'.
- A 'Domain' input field with the value 'entigo.io' and a dropdown menu showing '.entigo.io'.
- A text label 'Your company's domain name (ex. test.entigo.io)'.
- A checkbox labeled 'Accept Terms & Conditions'.
- A 'Confirm' button.

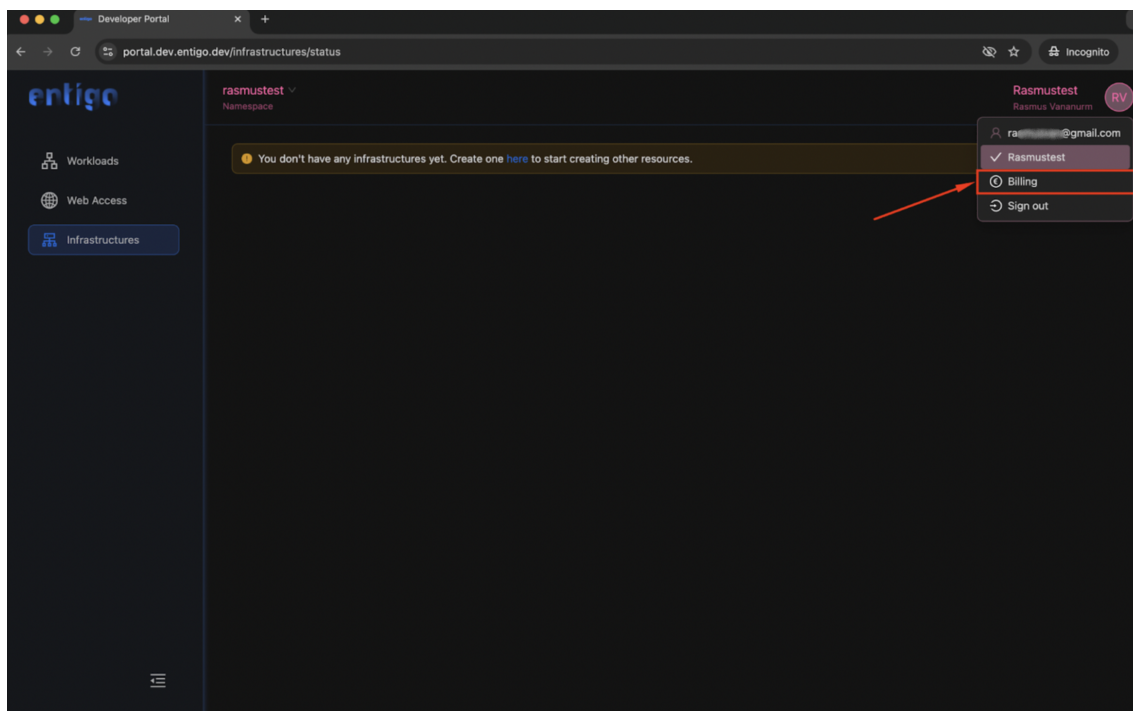
Joonis 8. Kasutajaandmete leht



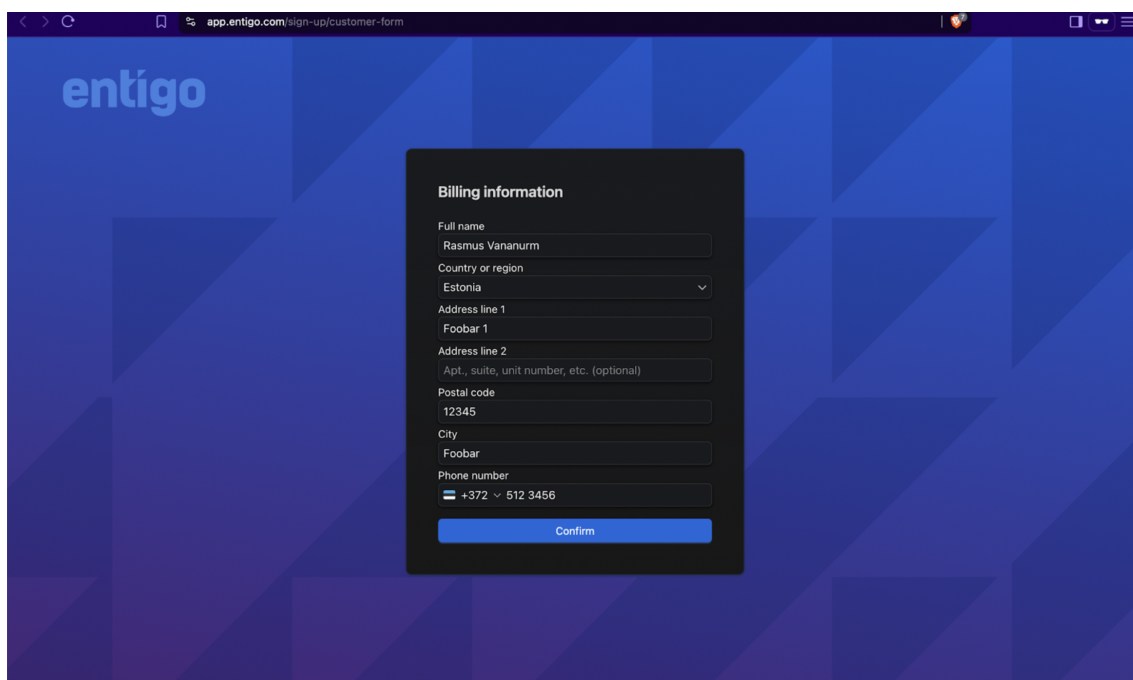
Joonis 9. Ressursid loodud leht



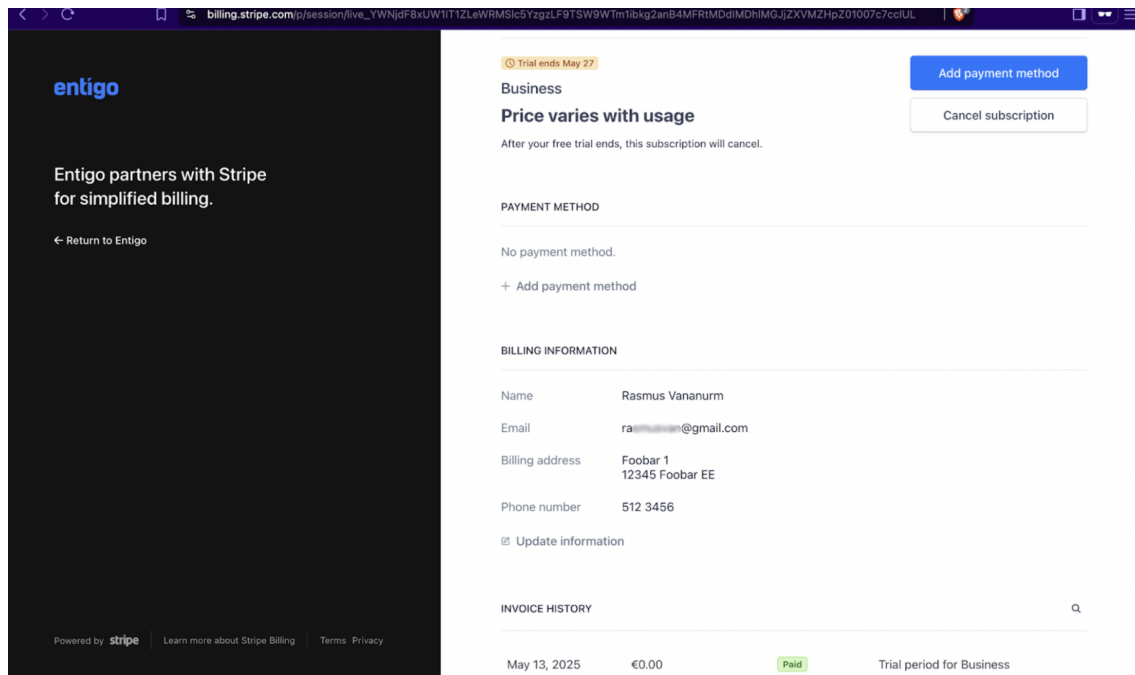
## Lisa 3 – Maksehaldusvoo vaated



Joonis 10. Platvormi “Billing” nupp



Joonis 11. Aadressivorm



Joonis 12. Stripe maksehaldusportaal