TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Maanus Indov 234160IADB

Sotsiaalmeedia platvorm PosTree infomüra vähendamiseks

Bakalaureusetöö

Juhendaja: Kristiina Hakk
PhD

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: Maanus Indov

06.01.2025

Annotatsioon

Käesoleva lõputöö eesmärk on luua hübriidmobiil- ja veebirakendus, mis vastab sotsiaalmeedia põhifunktsioonidele, kuid eristub sellega, et keskendub infomüra vähendamisele. Rakendus piirab kasutajate postituste hulka, suunates neid looma läbimõeldumat ja kvaliteetsemat sisu. Esialgne tooteversioon ja selle nõuded on kujundatud globaalse turu perspektiivist, et tagada kiire skaleerimisvõimalus vastavalt vajadusele.

Rakendus on loodud kasutades kaasaegseid ja populaarseid raamistikke, võttes aluseks sotsiaalmeediaplatvormide kasutusmustreid, et pakkuda sujuvat kasutajakogemust ja optimeeritud funktsionaalsust. Rakenduse esiliides on ehitatud Quasari raamistikul, mis tugineb Vue3-le, ning tagaserver on programmeeritud PHP keeles. Andmeid hallatakse MySQL andmebaasis, millele lisandub kiire andmevoo tagamiseks mälupõhine andmehoidja Redis.

Valminud rakendus pakub mitmekesiseid sotsiaalmeedia funktsioone, sealhulgas postituste loomist, piltide üleslaadimist, kasutajate jälgimist, postituste kommenteerimist ja lemmikute märkimist. Infovood ja kasutajasisu esitatakse erinevates postituste jadades, mis koosnevad peamiselt globaalsetest postitustest, kuid kasutajatel on ka võimalus luua isiklikud sisuvood.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 42 leheküljel, 6 peatükki, 22 joonist, 9 tabelit.

Abstract

Social Media Platform PosTree to Reduce Information Overload

The objective of this thesis is to develop a hybrid mobile and web application that fulfills the core functions of social media applications while addressing the issue of information overload. The application limits the number of posts users can make, encouraging more thoughtful and higher-quality content creation. The initial product version and its requirements are designed with a global market perspective, ensuring rapid scalability when needed.

The application has been developed using modern and popular frameworks, based on the usage patterns of social media platforms to provide a seamless user experience and optimized functionality. The front-end of the application is built using the Quasar framework, which relies on Vue3, while the backend is programmed in PHP. Data is managed in a MySQL database, complemented by an in-memory data store, Redis, to ensure fast data flow.

The completed application offers a variety of social media features, including post creation, image uploading, user following, post commenting, and marking favorites. The information feeds and user content are presented in different post streams, primarily consisting of global posts, while users also have the option to create their own personalized content streams.

The thesis is in written in Estonian language and contains 42 pages of text, 6 chapters, 22 figures, 9 tables.

Lühendite ja mõistete sõnastik

ACID	*Atomicity, Consistency, Isolation, Durability* on andmebaasi omaduste kogum mis tagab andmete töökindluse
Android	Avatud lähtekoodiga Linuxil põhinev operatsioonisüsteem mobiilsetele seadmetele
Beta	Tarkvaraarenduse etapp kus toode on testimisel
API	*Application Programming Interface* ehk rakenduse liides mis võimaldab erinvatel rakendustel infot vahetada
CI/CD	*Continuous Integration and Continuous Delivery/Deployment* ehk tarkvaraarenduse protsess mis automatiseerib koodi integreerimise, testimise ja väljalaskeprotsessi
CDN	*Content Delivery Network* ehk serverite võrgustik mis toob sisu inimestele geograafiliselt lähemale tagades kiirema sisu laadimise
Deep link	Ehk süvalink on URL, mis kontrollib rakenduse olemasolu ja avab selle veebilehitseja asemel, kui rakendus on installitud
DNS	*Domain Name System* on süsteemi mis tõlgib domeeni nimed IP aadressideks
Docker	Konteineriseerimisplatvorm mis võimaldab rakendusi ja nende sõltuvusi pakendada konteinerisse tagades järjepideva töö nii arendus- kui ka tootmiskeskkonnas
ER	*Entity Relationship* on andmebaaside projekteerimismeetod mis kirjeldab andmestruktuuri üksuste atribuutide ja seoste kaudu
EXIF	*Exchangeable Image File Format* on metaandmete standard mis salvestab digifotode ja -videode kohta teavet
Git	Versioonihaldussüteem mis jälgib koodimuutusi
HTML	*Hypertext Markup Language* on veebilehtede struktuuri ja sisu loomise märgistuskeel
HTTPS	*Hypertext Transfer Protocol Secure* on turvaline andmevahetuse protokoll mis tagab serveri ja veebilehitseja vahel oleva privaatsuse
iOS	*iPhone Operating System* on Apple poolt loodud operatsioonisüsteem mobiilsetele seadmetele
JavaScript	Programmeerimiskeel mida kasutatakse veebitarkvara loomiseks

JSON	*JavaScript Object Notation* ehk lihtne andmevorming mida kasutatakse andmete edastamiseks, on inim- kui ka masinloetav
NoSQL	Mitte relatsiooniline andmebaas
MIT-litsents	Vabavara tarkvaralitsents, mis lubab tarkvara vabalt kasutada, kopeerida, muuta ja levitada, nõudes ainult algse autori õiguste säilitamist
MVP	*Minimum Viable Product* on minimaalne kasutuskõlbulik toode mis on loodudesmase tagasiside saamiseks
NPM	*Node Package Manager* on Node.JS protsessi ja teekide halduse tööriist
PHP	Serveripoolne skriptikeel mida kasutatakse dünaamiliste veebilehtede ja/või tagarakenduste arendamiseks
PSR	*PHP Standards Recommendation* on standard mis määrab reeglid PHP klasside automaatseks laadimiseks
PUG	PUG (varasemalt Jade) on HTML-i mallikeel, mis võimaldab luua lihtsustatud ja loetavama süntaksiga struktuuri
Tõuketeavitus	Tõuketeavitused (push notifications) on reaalaajas saadetavad teated mis edastatakse kasutaja seadmesse ka siis kui seadet ei kasutata
RDB	*Relational Database* on andmebaas mille tabelite vahel on seosed
Redis	Kiire mälupõhine andmebaas
REST	*Representational State Transfer* on tarkvaraarhitektuuri stiil mis seab veebirakenduse loomisele kindlad piirid
SSH	*Secure Shell* on turvaline protokoll ja tööriist kaugserverite haldamiseks käsurea kaudu kasutades krüpteeritud sidekanalit
SSL	*Secure Sockets Layer* on turvaprotokoll mis tagab andmete krüpteeritud ja turvalise edastamise võrgu kaudu
TypeScript	JavaScript keele laiendus mis lisab keelele võimaluse luua staatilisi tüüpe
URL	*Uniform Resource Locator* on ühene aadress mida kasutatakse infoallikate leidmiseks ja kasutamiseks internetis
UX	*User Experience* on kasutajakogemuse disain, mis keskendub toote või teenuse lihtsusele, mugavusele ja rahuldavusele kasutaja jaoks

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Sotsiaalmeedia rakendused kuuluvad maailma enimkasutatavate igapäevaste rakenduste hulka. Neid rakendusi kasutab igapäevaselt üle kolme miljardi inimese [1]. Selline suur kasutajate hulk on toonud kaasa uue väljakutse: infomüra suurenemine, mis põhjustab inimeste tähelepanu hajumist ja vähendab sisulise kommunikatsiooni efektiivsust.

Käesoleva lõputöö eesmärk on luua hübriid-mobiilirakendus, mille keskseks ideeks on sisuloome piiramine. Piirangu eesmärk on suunata kasutajaid looma sisukamat ja tähendusrikkamat sisu, vähendades seeläbi infomüra, pakkudes alternatiivi tänastele platvormidele, kus kvantiteet tihti kvaliteedi üle domineerib.

Lõputöö autor on arvestanud, et sisuloome piirang võib olla vastuolus kasutajate ootustega, ning see võib mõjutada platvormi vastuvõttu. Siiski viitavad andmed sellele, et keskmine kasutaja ei loo igapäevaselt suurt hulka postitusi, kui siis maksimaalselt ühe päevas [2]. Sellest lähtudes võib piirang osutuda vähem piiravaks, kui esmapilgul tundub.

Käesoleva lõputöö raames ei viida läbi uuringut selle kohta, kuidas loodud rakendus inimesi mõjutab. Arenduse eesmärk on luua toote prototüüp hüpoteesi testimiseks.

Lõputöö teises peatükis vaadeldakse probleemi olemust lähemalt ning kirjeldatakse eesmärke ja metoodikat rakenduse loomiseks. Kolmandas peatükis analüüsitakse maailma populaarsemaid sotsiaalmeedia rakendusi ja nende funktsionaalsust. Selle analüüsi põhjal püstitatakse nõuded, mis viiakse ellu neljandas peatükis, kus kirjeldatakse rakenduse loomise protsessi. Viiendas peatükis antakse ülevaade valminud lahendusest ning arutletakse edasiarendusvõimaluste üle.

2 Ülesande püstitus

Kaasaegne sotsiaalmeedia on loonud keskkonna, kus iga päev tarbitakse tohutul hulgal informatsiooni. Suur osa tarbitavast sisust ei vasta inimeste tegelikele huvidele ega vajadustele. Selle tulemusena on kujunenud välja käitumismustrid, mis viivad sisulise tähelepanu hajumiseni ja olulise teabe varju jäämiseni.

Käesolevas peatükis käsitletakse lõputöös kajastuvat probleemi põhjalikumalt, määratletakse selle lahendamiseks seotud eesmärk ning kirjeldatakse metoodikat, mille alusel probleemile lähenetakse.

2.1 Probleem

Uuringud on näidanud, et liigne infovoog piirab kasutajate võimet olulist sisu tõhusalt omandada ja levitada [3]. Liiga erineva, mitte huvidele vastava sisu kombinatsioon põhjustab olukorra, kus paljudel inimestel on raskusi olulise teabe omandamisega, sest aju harjub end välja lülitama. Sotsiaalmeedia platvormide massiline kasutamine on soodustanud kiiret, läbimõtlemata sisutarbimist, mis viib kasutajate tähelepanu hajumise ning informatsiooni töötlemise raskusteni [4].

2.2 Eesmärk

Käesoleva lõputöö eesmärk on luua sotsiaalmeediarakendus, mis on suunatud esialgu Euroopa turule ja on kättesaadav nii veebis kui ka Android ja iOS (*iPhone Operating System*) rakenduste poodides. Rakendus peab sisaldama põhilisi sotsiaalmeedia funktsionaalsusi, tagades kasutajatele kiire ja sujuva registreerimise ning platvormi kasutamise.

Loodav rakendus peab olema tehniliselt valmis skaleeruma globaalseks platvormiks, mille tagamiseks tuleb pöörata tähelepanu tehnoloogilistele lahendustele juba arenduse varases etapis.

Lõpptoote üheks osaks saab olema dokumenteeritud API (*Application Programming Interface)*, mis võimaldab arendajatel luua rakendusega ühilduvaid laiendusi ja skripte. See suurendab rakenduse kasutusvõimalusi ning muudab selle atraktiivsemaks nii kasutajatele kui ka arendajate kogukonnale.

2.3 Metoodika

Lõputöö raames kasutatakse kombineeritud arenduslähenemist, mis ühendab agiilsete meetodite, testimise ja analüüsiprotseduuride elemente, et luua kasutajasõbralik ja efektiivne platvorm.

Enne rakenduse arendamist koostatakse analüüs populaarsete sotsiaalmeediarakenduste funktsionaalsustest, mille põhjal kujundatakse rakenduse kasutajaliides, et pakkuda kasutajatele tuttavat ja intuitiivset kasutajakogemust. Selline lähenemine vähendab platvormi kasutamise õppekõverat ning suurendab tõenäosust, et kasutajad naasevad platvormile ka edaspidi.

Testimisfaas viiakse läbi kahes etapis. Esmalt testitakse rakendust lokaalsel tasandil, et avastada ja parandada tehnilisi puudusi. Seejärel kaasatakse beetatestimise faasi valitud sihtgrupp, kellelt kogutakse tagasisidet. Kogutud andmete põhjal optimeeritakse rakendus enne avalikku väljalaset, tagades parema kasutajakogemuse ja funktsionaalsuse.

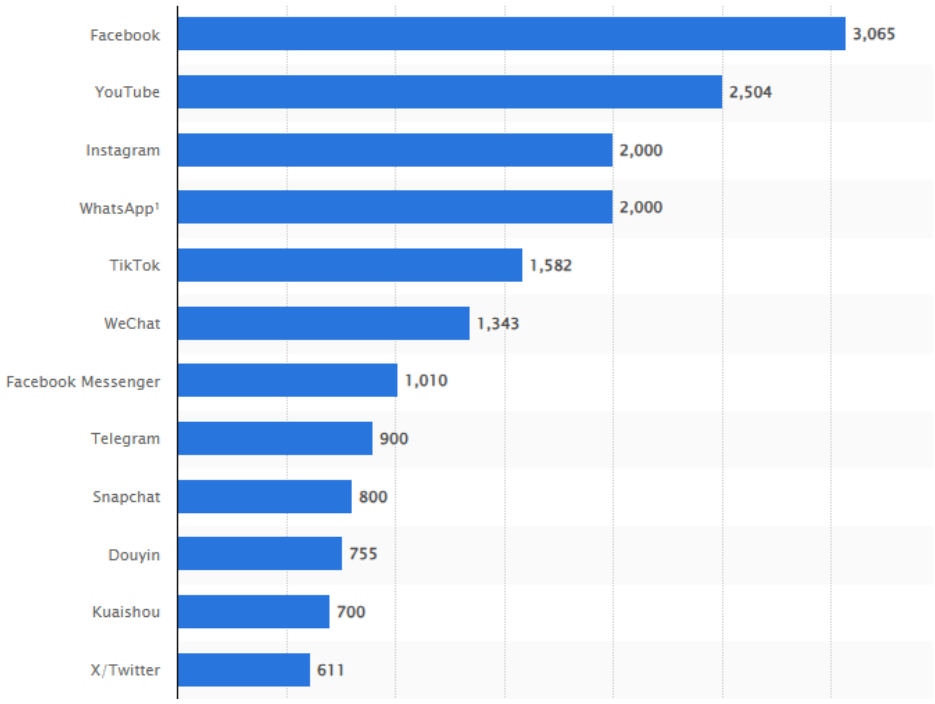
3 Analüüs

Antud lõputöö raames analüüsitakse suurimaid sotsiaalmeediarakendusi ja nende peamisi funktsioone. Keskendutakse kasutajate kaasamise ja platvormi püsimise edendamisele suunatud strateegiatele ning funktsionaalsustele, mis toetavat kasutajakogemuse tõhustamist.

Analüüs keskendub sotsiaalmeedia rakendustele, mis on mõeldud piltide, postituste ja muu sisu jagamiseks, pakkudes kasutajatele vooge, kommenteerimisvõimalusi ja sotsiaalseid interaktsioone.

3.1 Turul olevad rakendused

Rakenduse loomisel on oluline analüüsida maailma populaarsemaid sotsiaalmeediaplatvorme, et mõista, millised nõuded ja funktsionaalsused on kasutajate ootuste täitmiseks vajalikud. Selles töös jäetakse kõrvale platvormid, mille keskseks funktsionaalsuseks on video: YouTube ja TikTok, sest need ei vasta loodava rakenduse fookusele. Samuti ei käsitleta vestluspõhiseid platvorme, nagu WhatsApp, WeChat, Facebook Messenger, Telegram ja Snapchat, sest nende funktsioonid keskenduvad peamiselt kiirsuhtlusele, mitte sotsiaalmeedia sisuloomele. Joonisel 1 on esitatud erinevate rakenduste kasutajate hulk miljonites.



Joonis 1. Maailma kõike populaarsemad sotsiaalmeedia rakendused, aprill 2024. [1]

Lähtudes kasutajate arvust, uurib lõputöö autor millised on kõige suuremate kasutajatega platvormide põhifunktsionaalsused. Tabel 1 võtab kokku sotsiaalmeedia rakenduste põhifunktsionaalsused, tabel on loodud autori kogemuse põhjal rakendusi katsetades.

Tabel 1. Turul olevate sotsiaalmeedia rakenduste põhifunktsionaalsused.

Funktsionaalsus	**Facebook**	**Instagram**	**Twitter**
Postituste voog	+	+	+
Pildid	+	+	+
Videod	+	+	+
Otseülekanded	+	+	+
Kommenteerimised	+	+	+
Jälgimissüsteem	+	+	+
Privaatsõnumid	+	+	+
Tõuketeavitused	+	+	+
Lühivideod	+	+	+

Tabelist võib järeldada, et kõik analüüsitud platvormid jagavad mitmeid olulisi põhifunktsioone, mis moodustavad sotsiaalmeedia platvormide tuumfunktsionaalsuse ja vastavad kasutajate ootustele. Selline analüüs annab tugeva aluse nõuete püstitamiseks,

tagades, et loodav rakendus vastab turul levinud standarditele. Need funktsioonid on muutunud *de facto* standardiks, mida kasutajad sotsiaalmeediaplatvormidelt ootavad.

3.2 Nõuded

Eduka sotsiaalmeediarakenduse loomiseks on oluline tagada funktsionaalsuste komplekt, mis vastab kasutajatele ootustele ja turustandarditele.

MVP (*Minimum Viable Product)* versiooni loomisel on välja jäetud kõik videoga seotud funktsionaalsused. Videote töötlemine nõuab märkimisväärseid ressursse nii arenduse kui ka infrastruktuuri poolelt [5], mistõttu on see otsus põhjendatud, et keskenduda platvormi esmasele ülesehitusele ja kvaliteedi tagamisele.

3.2.1 Funktsionaalsed nõuded

Funktsionaalsed nõuded määratlevad rakenduse võimekuse ja need tuleb realiseerida, et platvorm vastaks oma eesmärgile. Kui kõik funktsionaalsed nõuded on täidetud, on rakendus valmis testimiseks. [6]

Loodava rakenduse funktsionaalsed nõuded on järgmised:

- Rakendusse peab olema võimalikult lihtne siseneda.
- Rakenduses peab olema otsing.
- Rakenduses peab olema globaalne postituste voog.
- Rakenduses peab olema kasutaja kontrollitud voog.
- Rakenduses peab olema algoritmil baseeruv voog.
- Rakenduses peab olema halbade sõnade filter.
- Postitusele peab olema võimalik lisada meediafaile.
- Postitusele peab olema võimalik lisada silte (*tag*).
- Postitust peab olema võimalik muuta ja kustutada.
- Kasutaja peab saama korra päevas postitada.
- Kasutaja peab saama märkida postitusi meeldivaks.
- Kasutaja peab saama kommenteerida teisi postitusi.
- Kasutaja peab saama oma kommentaare muuta ja kustutada.
- Kasutaja peab saama jälgida teisi kasutajaid.
- Kasutaja peab saama blokeerida teisi kasutajaid.

- Kasutaja peab saama üles laadida pilte ja heli.
- Kasutaja peab saama erinevaid tõuketeavitusi.
- Kasutaja peab saama tõuketeavitusi välja lülitada.
- Kasutaja peab saama postitusi jagada.
- Kasutaja peab saama pilte suurendada.
- Kasutaja peab saama oma profiili muuta.

3.2.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded kirjeldavad süsteemi omadusi, mis tagavad rakenduse kvaliteedi ja toimivuse. Need nõuded määratlevad, kuidas süsteem peab töötama ja millistele standarditele vastama. [6]

Loodava rakenduse mittefunktsionaalsed nõuded on:

- Rakendusel peab olema avalik API liides.
- Rakendus peab olema kättesaadav veebilehitsejast.
- Rakendus peab olema kättesaadav mobiiliseadmest.
- Rakendusel peab olema allalaetav mobiilirakendus.
- Rakendus peab hoidma klientide üleslaetud meediafaile eraldiseisvalt.
- Süsteem peab olema töökindel ja suure kättesaadavusega.
- Süsteemil peab olema erinev konfiguratsioon vastavalt keskkonnale.
- Süsteemi peab olema kergesti teisaldatav teise serverisse.

3.3 Tehnoloogiate ülevaade

Rakenduse loomiseks on saadaval lai valik programmeerimiskeeli ja andmebaase. Lisaks tehnoloogia valikule on oluline määrata, kus majutatakse tagarakendus ja veebirakendus, et tagada katkematu kättesaadavus. Käesolevas peatükis analüüsitakse erinevaid tehnoloogilisi lahendusi, arvestades rakendusele esitatud nõudeid, olemasolevat infrastruktuuri ja autori kogemust.

3.3.1 Server

Rakenduse serveri ja majutusteenuse valik on kriitilise tähtsusega, sest see mõjutab otseselt süsteemi töökindlust, skaleeritavust ja jõudlust. Käesolevas alapeatükis

analüüsitakse erinevaid võimalusi tagarakenduse ja veebirakenduse majutuseks, võttes arvesse nii kulusid, halduskeerukust kui ka sobivust loodava rakenduse tehniliste nõuetega. Tabelis 2 on välja toodud erinevate teenusepakkujate võrdlus.

Tabel 2. Erinevate serveriteenuste võrdlus.

Teenus	Lihtsus	Kulu	Skaleeritavus	Kättesaadavus
Zone.ee [7]	+++	+	+	+++
AWS [8]	+	+++	+++	+++
Azure [9]	+	+++	+++	+++
DigitalOcean [10]	++	++	++	++

Tabelist võib järeldada, et erinevatel serveriteenustel on omad tugevused ja nõrkused, mis sõltuvad nii nende keerukusest, kuludest kui ka skaleeritavusest.

Zone.ee (veebimajutus teenus) paistab silma oma lihtsuse ja kättesaadavuse poolest, olles eriti sobiv väiksemate ja keskmise suurusega projektide jaoks. Teenus on kiiresti kasutusele võetav ning pakub eelseadistatud tuge selliste tehnoloogiate jaoks nagu PHP, MySQL ja Redis, mis vastavad loodava rakenduse tehnilistele nõuetele. Samas on skaleeritavus piiratud, kuna Zone.ee serverid asuvad ainult Eestis [11], mis seab piiranguid rakenduse laiendamisel Euroopa turust väljapoole.

AWS ja Azure on äärmiselt võimsad ja paindlikud teenused, mis tagavad kõrge skaleeritavuse ja kättesaadavuse, kuid nende kasutamine toob kaasa suuremad kulud ning keerukama halduse. Need platvormid sobivad hästi suuremahuliste ja globaalselt skaleeritavate rakenduste jaoks, kus on vajalik detailne infrastruktuuri juhtimine [12] [13].

DigitalOcean asetseb omadustelt AWS-i ja Zone.ee vahel. Teenus pakub head tasakaalu kulude, skaleeritavuse ja halduskeerukuse vahel. Siiski ei paku see sama kõrget töökindlust ja globaalset infrastruktuuri nagu AWS või Azure. [14]

Arvestades loodava rakenduse eesmärki ja praegust mahtu, on Zone.ee valik kõige otstarbekam. Selle lihtsus ja eelseadistatud lahendused võimaldavad keskenduda arendustööle, vältides keerukat infrastruktuuri haldust ja tarbetuid kulusid. Tulevikus, kui

rakenduse infrastruktuuri nõuded kasvavad, saab vajadusel üle minna skaleeritavamatele teenustele, nagu AWS või Azure.

3.3.2 Andmebaasid

Andmebaasid on andmete ja/või dokumentide kogumikud, mis kasutavad erinevaid meetodeid andmete lisamiseks, muutmiseks või kustutamiseks. Andmebaasid jagunevad üldiselt kahte kategooriasse: relatsioonilised (RDB) ja mitte-relatsioonilised (NoSQL) andmebaasid. Tabelis 3 on välja toodud erinvate populaarsete [15] andmebaaside võrdlus.

Tabel 3. Andmebaaside võrdlus [16].

Faktor	**MySQL**	**PostgreSQL**	**SQLite**	**MongoDB**
Tüüp	RDB	RDB	RDB	NoSQL
Õppekõver	+++	++	+++	++
Jõudlus	+++	+++	+++	+++
Skaleeritavus	++	+++	+	++
Populaarsus	+++	+++	+++	+++
Kogemus	+++	+++	+++	+

Relatsioonilised andmebaasid, nagu MySQL ja PostgreSQL, paistavad silma oma töökindlusele poolest mida tagab ACID (*Atomicity, Consistency, Isolation, Durability*) mudel [17]. See mudel on kriitilise tähtsusega, et iga kasutaja tehtud muudatus oleks korrektselt salvestatud ja andmete terviklikkus säiliks isegi süsteemi rikete korral. Lisaks on autori kogemus relatsiooniliste andmebaaside haldamisel suurem, mis lihtsustab arendusprotsessi ja võimaldab vältida tarbetuid vigu.

Lisaks MySQL ja PostgreSQL andmebaasidele võrreldi ka SQLite ja MongoDB kasutusvõimalusi. SQLite, kuigi kerge ja lihtsasti seadistatav, ei ole sobiv valik suuremahuliste rakenduste jaoks, kuna sellel puudub sisseehitatud tugi mitme kasutaja samaaegsetele päringutele ja see ei ole loodud keerukamate süsteemide skaleerimiseks [18]. Seetõttu ei vasta see loodava rakenduse nõuetele.

MongoDB, kui NoSQL andmebaas, pakub paindlikkust ja jõudlust skeemivabade andmete töötlemisel, kuid loodava rakenduse jaoks on vajalik andmete selge struktuur ja

relatsioonilised seosed, mida MongoDB ei toeta piisavalt hästi [19]. Lisaks ei ole autori kogemus MongoDB puhul piisav, et arendusprotsessi tõhusalt juhtida.

Arvestades, et MySQL on Zone.ee serveris eelseadistatud, muudab see arenduse ja juurutamise oluliselt lihtsamaks ning vähendab konfiguratsiooniga seotud ajakulu. Samuti vastab MySQL kõigile loodava rakenduse nõuetele, pakkudes piisavat jõudlust, turvalisust ja stabiilsust. Seetõttu otsustab autor kasutada MySQL-i andmete salvestamiseks.

3.3.3 Tagarakendus

Tagarakenduse arendamiseks on saadaval mitmeid populaarseid programmeerimiskeeli, mille valik sõltub süsteemi nõuetest ja jõudlusest. Selles alapeatükis analüüsitakse sobivaid keeli ja tehakse otsus, millist keelt rakenduse arendamiseks kasutada. Tabelis 4 on kokku pandud erinvate populaarsete keelte [15] võrdlus, mida saab kasutada tagarakenduste ehitamisel.

Tabel 4. Erinevate tagarakenduse programmeerimiskeelte võrdlus [20].

Faktor	**PHP**	**C#**	**Java**	**Node.js**	**Rust**	**Go**
Õppekõver	+++	++	++	+++	+	++
Jõudlus	++	+++	+++	+++	++++	++++
Skaleeritavus	++	++	++	++	++++	++++
Populaarsus	+++	+++	++	+++	++	+++
Kogemus	+++	++	+	+++	++	++

Tabelist selgub, et erinevatel programmeerimiskeeltel on oma tugevused ja nõrkused, mis sõltuvad rakenduse eesmärkidest ja nõuetest. Loodava rakenduse puhul on olulised piisav jõudlus, töökindlus ning arendusprotsessi lihtsus ja efektiivsus MVP etapis.

Kuigi keeled nagu Rust ja Go paistavad silma oma suure jõudluse ja parema skaleeritavusega keerukamates süsteemides [21], ei ole need MVP arendusetapis praktilised. Infrastruktuuri kohandamine ja arenduskeskkonna seadistamine nende keelte jaoks nõuaks autorilt oluliselt rohkem ressursse kui eelseadistatud keskkonna kasutus. Samuti ei ole rakenduse praegused nõuded nii spetsiifilised, et vajaksid sellist kõrgtasemel optimeerimist.

C# ja Java nõuavad infrastruktuuri, mis võimaldaks suuremat arendaja kontrolli, kuid see tähendaks samuti suuremat esialgset ressursikulu. Näiteks eeldab C# sageli Windowsi-põhiseid servereid (mis on kallimad) ja .NET Core seadistamist [22], samas kui Java nõuab keerukamat arendus- ja juurutuskeskkonda. Mõlemad variandid võivad MVP arenduse kontekstis olla liiga ressursimahukad.

Autori valikuks osutub PHP, peamiselt tänu oma sobivusele olemasoleva infrastruktuuriga. Zone.ee serverikeskkond pakub eelseadistatud PHP tuge, sealhulgas *Zone FastCGI Process Manager*, mis parandab oluliselt jõudlust, võimaldades suuremat koormust ja kiiremat vastamisaega [23]. Optimeeritud seadistused Zone.ee keskkonnas ja PHP versioonihaldus tagavad, et PHP põhine rakendus töötab tõhusalt nii algfaasis kui ka kasvades.

3.3.4 Esirakendus

Esirakenduse arendamiseks on saadaval mitmeid raamistikke, mis võimaldavad luua hübriidrakendusi, töötades nii veebis kui ka mobiilseadmetes. Selles alapeatükis analüüsitakse erinevaid raamistikke, pöörates tähelepanu nende jõudlusele, arenduslihtsusele, komponentide valikule ja mobiilseadmete toetusele. Valiku tegemisel on oluline arvestada rakenduse nõudeid, arendusprotsessi kiirust ning paindlikkust. Tabelis 5 on välja toodud enim populaarsust [15] kogunud hübriidrakenduse raamistikud.

Tabel 5. Hübriidraamistike võrdlus. [24], [25], [26].

Faktor	**Quasar**	**React Native**	**Flutter**	**Ionic**
Õppekõver	+++	++	++	++
Komponendid	+++	+++	+++	+++
Jõudlus	++	+++	+++	++
Populaarsus	++	+++	+++	++
Kogemus	+	++	+	++

React Native on laialdaselt kasutatav ja populaarne raamistik, mis võimaldab arendada platvormiüleseid rakendusi JavaScripti abil. Kuigi selle ökosüsteem on rikas ja toetab paljusid komponente, eeldab see tihti arendajalt suuremat süvenemist platvormipõhiste kohanduste tegemiseks [27]. Samuti on selle komponentide ökosüsteem tugevalt JavaScript keskne, mis ei sobi täielikult autori eelistustele.

Flutter on tehniliselt võimas ja pakub kõrget jõudlust, kuid see nõuab rohkem ressursse arenduse esimeses etapis, kui puudub kogemus raamistikuga. Dart programmeerimiskeele ja Flutteri spetsiifilise ökosüsteemi õppimine ning juurutamine muudab selle keerukamaks lahenduseks, mis ei pruugi MVP arendamisel olla ajaliselt ja ressursiliselt efektiivne.

Ionic ja Quasar on mõlemad ehitatud Capacitori peale, pakkudes suurepärast tuge hübriidrakenduste loomiseks. Nende eeliseks on ühtne koodibaas, mida saab kasutada nii veebis kui ka mobiilseadmetes. Mõlemad raamistikud toetavad Vue Frameworki, mis on loodava rakenduse tehnilise struktuuri jaoks sobivaim valik. [28] [29]

Quasar Framework paistab silma oma komponentide kogumi ja moodulite eelseadistusega, mis kiirendavad arendust ja lihtsustavad juurutamist [30]. See pakub valmis tööriistu, mis vähendavad arendusaega ja võimaldavad keskenduda rakenduse põhifunktsionaalsusele. Lisaks on Quasar vabavara (*open-source)* raamistik, mille MIT-litsents [31] võimaldab selle vabalt kasutamist nii isiklikel kui ka kommertseesmärkidel, pakkudes samas läbipaistvust ja laialdast tuge arendajate kogukonnas.

Autori kogemus Vue Frameworkiga muudab Quasar Frameworki tugevaks kandidaadiks, ning see valitakse esirakenduse arendamiseks, tagades kiirema töövoo ja vähendades tehniliste takistuste tekkimise riski arenduse käigus..

3.4 Kaasatud lisavalikud

Loodava rakenduse arendamisel on lisaks põhitehnoloogiatele valitud mitmeid täiendavaid tööriistu ja raamistikke, mis aitavad parandada süsteemi jõudlust, kasutajakogemust ja funktsionaalsust. Need valikud toetavad rakenduse tehnoloogilist arhitektuuri ja tagavad selle vastavuse kaasaegsetele standarditele.

3.4.1 Mälupõhised andmehoidjad

Rakenduse nõuetes on oluline tagada kiire andmete kättesaadavus, eriti suurema kasutajakoormuse korral. Selleks saab kasutada mälupõhist andmehoidjat, mis salvestab sageli kasutatavaid andmeid ajutiselt mällu, vähendades seeläbi relatsioonilise andmebaasi koormust. Mälupõhise (*In-Memory*) andmehoidja kasuteguri selgitamiseks viis autor läbi teste, et määrata hetk, mil lisaprotsesside kasutamine end ära tasub. Testid

viidi läbi eraldatud keskkonnas, kus mõõtmistest arvestati maha ühenduste loomisele kuluv aeg. Tabelis 6 on kujutatud testi tulemused mälupõhise lugemise ja kirjutamise ajalisest mõjust, kus protsesside järjekord on fikseeritud.

Tabel 6. Mälupõhisest andmehoidjast lugemise ja kirjutamise ajaline mõju.

Iteratsioon	**IM (lugemine)**	**DB (lugemine)**	**IM (kirjutamine)**	**Kokku**
1	0.07ms	0.31ms	0.07ms	0.45ms
2	0.09ms	0.41ms	0.10ms	0.60ms
3	0.09ms	0.34ms	0.07ms	0.50ms
4	0.08ms	0.33ms	0.07ms	0.48ms
5	0.07ms	0.32ms	0.08ms	0.47ms

Tulemused näitavad, et mälupõhise andmebaasi lahenduse kasutamine lisab iga andmebaasi päringu ette (info olemasolu kontrollimiseks) ja järgi (info salvestamiseks) täiendava ajakulu. Kuigi individuaalne lisanduv aeg on minimaalne, on oluline hinnata selle mõju suurema koormuse või korduvate päringute korral. Tabelis 7 on kujutatud viis päringut samale ressursile.

Tabel 7. Mälupõhise andmehoidja kasutegur sama ressursi pärimisel.

Protsess	**1**	**2**	**3**	**4**	**5**
IM (lugemine)	0.08ms	0.08ms	0.08ms	0.08ms	0.08ms
DB (lugemine)	0.36ms	0ms	0ms	0ms	0ms
IM (kirjutamine)	0.08m	0ms	0ms	0ms	0ms
Kokku	0.52ms	0.08ms	0.08ms	0.08ms	0.08ms

Tabelis esitatud andmed näitavad mälupõhise andmehoidja lahenduse efektiivsust korduvate päringute korral. Kuigi esimene päring on aeglasem täiendavate protsesside tõttu, muutuvad järgnevad päringud märgatavalt kiiremaks, kuna mälupõhisest andmehoidjast lugemine on oluliselt kiirem kui otse relatsioonilisest andmebaasist..

See näitab, et korduvate päringute korral vähendab mälupõhise andmehoidja lahendus andmebaasi koormust ja tagab kiirema vastamisaja. Seetõttu on vahemälulahenduse kasutamine õigustatud, eriti olukordades, kus samu andmeid tarbitakse sageli ja kõrge jõudlus on kriitilise tähtsusega.

Mälupõhised andmehoidjad, nagu Redis, KeyDB ja Memcached, pakuvad sarnaseid funktsioone, võimaldades kiiret andmete lugemist ja kirjutamist [32]. Zone.ee serveris on saadaval eelseadistatud Redis, mis vastab loodava rakenduse nõuetele ja tagab vajaliku jõudluse. Arvestades selle kättesaadavust ja sobivust rakenduse infrastruktuuriga, on Redis parim valik mälupõhiseks andmehoidjaks.

3.4.2 TypeScript

TypeScript on staatilise tüübiga programmeerimiskeel, mis laiendab JavaScripti võimalusi, lisades tüübitoe ja parandades tööriistade tuge. Enne rakenduse juurutamist kompileeritakse TypeScript JavaScriptiks, säilitades täieliku ühilduvuse JavaScripti keskkondade ja brauseritega. Selle kasutamine aitab ennetada tüüpilisi JavaScripti vigu, muutes koodi hooldatavamaks ja usaldusväärsemaks. TypeScript on täielikult toetatud Vue3 raamistikus [33], võimaldades kasutada tüübituge koos Vue komponentide ja mallidega. Joonisel 2 on kujutatud kohene tagasiside tehtud veast, mis suure JavaScript rakenduse koodibaasi puhul võib jääda kergelt märkamata.

Joonis 2. Vasakul JavaScript paremal TypeScript.

3.4.3 PUG

PUG on lihtne ja efektiivne mallikeel, mis muudab HTML-i (*Hypertext Markup Language)* kirjutamise kiiremaks ja loetavamaks, eemaldades liigse süntaksi ning pakkudes selgemat ja loogilisemat struktuuri. PUG-i võlu peitub selle minimalistlikus lähenemises, kus HTML sildid ja hierarhia saab väljendada puhtamal ja loetavamal kujul. Selline lähenemine aitab vähendada arendajate vigu ning kiirendab keerukamate struktuuride loomist. [34]

PUG integreerub sujuvalt Vue3 raamistikuga, kus *vue-language-server* teek pakub tuge PUG mailide süntaksi valideerimiseks ja automaatseks täiendamiseks. See lihtsustab arendusprotsessi tagades mallide ja komponentide koodi vahelise sünkroonsuse. Joonisel 3 on kujutatud PUG lihtsuse võlu.

Joonis 3. Vasakul HTML paremal PUG.

3.4.4 Capacitor

Capacitor on kaasaegne platvorm, mis võimaldab luua hübriidrakendusi. See pakub juurdepääsu paljudele seadmespetsiifilistele funktsioonidele, nagu tõuketeavitused, süvalingid (*deep-links*) ja lokaalne andmesalvestus [35]. Quasar Framework on loodud toimima koos Capacitoriga, võimaldades kasutada ühtset koodibaasi nii mobiili- kui ka veebirakendustes [36]. See arhitektuuriline lahendus vähendab arenduskulusid, kiirendab töövoogu ning lihtsustab seadmespetsiifiliste funktsioonide lisamist ja haldamist.

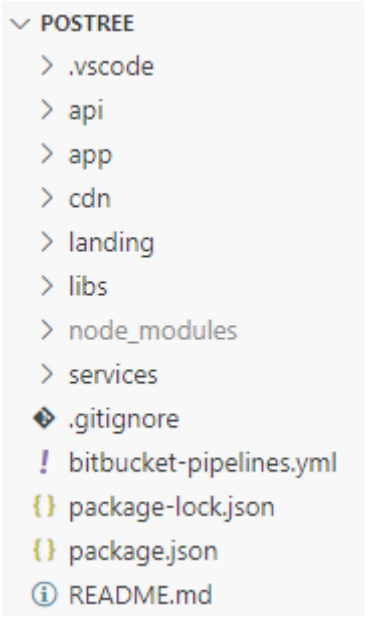
Lisaks toetab Capacitor mitmesuguseid pistikprogramme ja API-sid, mis hõlbustavad keerukamate funktsioonide, näiteks kaamerakasutuse, failitöötluse või võrguühenduse haldamist [37]. See teeb Capacitorist hädavajaliku vahendi kaasaegse hübriidrakenduse arendamiseks, tagades laialdase seadmetoe, stabiilsuse ja funktsionaalsuse. Integreerimisvõimalused seadmespetsiifiliste funktsioonidega muudavad selle eriti väärtuslikuks loodava rakenduse kontekstis.

4 Rakenduse loomine

Zone.ee teenus pakub mitmeid olulisi vahendeid, mis toetavad rakenduse arendust ja haldust. Antud lõputöös kasutatud tehnoloogiad, nagu MySQL, Redis ja PHP on Zone.ee keskkonnas eelseadistatud, mis lihtsustab arendusprotsessi märkimisväärselt [7]. See võimaldab autoril keskenduda arendustööle, vältides keerulisi infrastruktuuriga seotud probleeme. Zone.ee platvormi pakutavad ressursid ja tööriistad loovad tugeva ja paindliku aluse rakenduse stabiilseks käitamiseks ja tõhusaks haldamiseks.

4.1 Arenduskeskkond

Rakenduse repositoorium seatakse üles *monorepo* stiilis, kus ühe süsteemi all on koondatud kõikide rakenduse osade konfiguratsioonid ja koodifailid. Selline struktuur võimaldab *full-stack* arendajal hõlpsasti muuta ja kasutada ühiseid komponente nii tagaserveris kui ka esiliideses. Lisaks lihtsustab see andmestruktuuride ühtlustamist rakenduse erinevate osade vahel. Arenduskeskkonna seadistamisel tehakse otsuseid, mis valmistavad rakenduse ette tulevaseks skaleerimiseks. Näiteks on erinevad teenused ja komponendid eraldatud Docker konteineritesse, mis muudab nende haldamise ja laiendamise oluliselt lihtsamaks. Selline paindlik ja kaasaegne lähenemine arenduskeskkonnale tagab, et rakendus suudab tulevikus vastata nii tehnilistele kui ka kasvava kasutajaskonna vajadustele. Joonisel 4 on kujutatud repositooriumi failistruktuur.



Joonis 4. Monorepo struktuur.

Allolev nimekiri kirjeldab repositooriumi struktuuri põhikomponente, kus iga kaust või fail täidab konkreetset rolli rakenduse arenduse ja halduse protsessis. See struktuur mitte ainult ei lihtsusta arendustööd, vaid võimaldab ka Docker konteinerite tõhusamat konfigureerimis.

Monorepo struktuuriüksused:

- api: tagarakendus;
- app: esirakendus;
- cdn: CDN (*Content Delivery Network*) konteineri staatiline sisu;
- landing: koduleht;
- libs: kohandatud teegid ja jagatud koodilõigud;
- services: arenduskeskkonna konfiguratsioon, Docker konfiguratsioon;
- bitbucket-pipelines.yml: Bitbucket pipeline konfiguratsioon;
- package.json: NPM (*Node Package Manager*) konfiguratsioonifail;
- README.md: info ja juhised repositooriumi kasutamiseks.

Bitbucket

Rakenduse lähtekood talletatakse Bitbucket versioonihaldus süsteemis (*repository)*, mis pakub tugevat ja paindlikku lahendust versioonihalduseks. Bitbucket integreerub sujuvalt Atlassian tööriistade ökosüsteemiga, võimaldades kasutada efektiivseid projektihaldustööriistu, nagu Jira või Confluence, mis võivad tulevikus arendusprotsessi veelgi optimeerida [38]. Lisaks pakub Bitbucket sisseehitatud CI/CD (*Continuous Integration and Continuous Delivery/Deployment*) tööriistu, mis automatiseerivad koodi testimise ja juurutamise protsessi, muutes arendustsükli kiiremaks ja tõhusamaks [39].

Docker

Docker võimaldab luua isoleeritud keskkondi, kus iga teenus töötab eraldi konteineris, parandades süsteemi skaleeritavust ja hallatavust. Konteinerite vahelise liikluse haldamiseks kasutatakse Traefiku pöördproksi (*reverse proxy*) tööriista [40], mis suunab päringud vastavatele teenustele.

Täiendavalt on lokaalses keskkonnas konfigureeritud *hosts* faili muudatused, mis võimaldavad ligipääsu rakendustele läbi kohandatud domeeni või DNS-i (*Domain Name System)*, näiteks app.domeen.dev. See on eriti kasulik näiteks Google ja Microsoft sisselogimisvõimaluste juurutamisel, sest sisselogimise *callback* URL-id (*Uniform Resource Locator)* nõuavad kehtivat domeeni. Selline seadistus võimaldab arendajal simuleerida päris keskkonnas rakenduse käitumist, kuid seda kõike juba lokaalses keskkonnas, vähendades hilisemaid probleeme rakenduse juurutamisel ja testimisel. Joonisel 5 on kujutatud ühe konteineri konfiguratsioon (API).

```
api:
  image: api:lcl
  container_name: api
  build:
    context: ./../api
  volumes:
    - "${APP}/api:/var/www/html"
    - "${DEV}/cdn/uploads:/var/www/uploads"
  deploy:
    restart_policy:
      condition: on-failure
  depends_on:
    - traefik
    - mysql
    - redis
  labels:
    - "traefik.backend=api"
    - "traefik.enable=true"
    - "traefik.frontend.rule=Host:api.postr.dev"
    - "traefik.docker.network=traefik_proxy"
```

Joonis 5. API konteineri konfiguratsioon.

Lokaalne arenduskeskkond on seadistatud kasutama turvalist HTTPS (*Hypertext Transfer Protocol Secure)* ühendust. See võimaldab testida selliseid funktsioone nagu turvalised API päringud ja kolmandate osapoolte teenused, mis nõuavad kehtivat HTTPS protokolli (näiteks Google ja Microsoft sisselogimisprotsessid). Selline seadistus viib arenduskeskkonna lähemale reaalsete kasutustingimuste simuleerimisele , lihtsustades hilisemaid juurutus- ja testimisprotsesse.

Selle saavutamiseks genereeritakse ise allkirjastatud SSL (*Secure Sockets Layer)* sertifikaat, kasutades *mkcert* teeki [41]. Sertifikaadid lisatakse nii *traefik* konfiguratsioonifaili kui ka arendaja operatsioonisüsteemi sertifikaatide hulka. See tagab, et arenduskeskkond jäljendab võimalikult täpselt tootmiskeskkonna seadistusi. Joonisel 6 on kujutatud *traefik* konfiguratsioon, mis suunab kõik liiklused HTTPS ühendusele.

```
[entryPoints]
  [entryPoints.http]
  address = ":80"
    [entryPoints.http.redirect]
    entryPoint = "https"
  [entryPoints.https]
  address = ":443"
    [entryPoints.https.tls]
      [[entryPoints.https.tls.certificates]]
      certFile = "/etc/certs/postr.dev+1.pem"
      keyFile = "/etc/certs/postr.dev+1-key.pem"
```

Joonis 6. Traefiku SSL konfiguratsioon.

4.2 Andmebaas

Redis

Redis integreeritakse juba arenduse varajases faasis, sest vahemälulahenduse lisamine hilisemas etapis on keerukam ja kallim. Selline lahendus valmistab süsteemi ette suure koormuse käsitlemiseks, näiteks kümnete tuhandete päringute sekundis töötlemiseks. Joonisel 7 on kujutatud lihtsustatud näide Redise kasutamisest.

```
public function getUserById(int $user_id): ?User
{
    $redis_key = Redis::id(User::class, $user_id);

    $cached = Redis::get($redis_key);
    if (!empty($cached)) {
        $cached = unserialize($cached);
        return $cached;
    }

    $query = "SELECT ... WHERE `user_id` = ?";
    $params = [$user_id];
    $entity = DB::fetch($query, $params);
    Redis::set($redis_key, serialize($entity));
    return $entity;
}
```

Joonis 7. Lihtsustatud näide Redise kasutamisest.

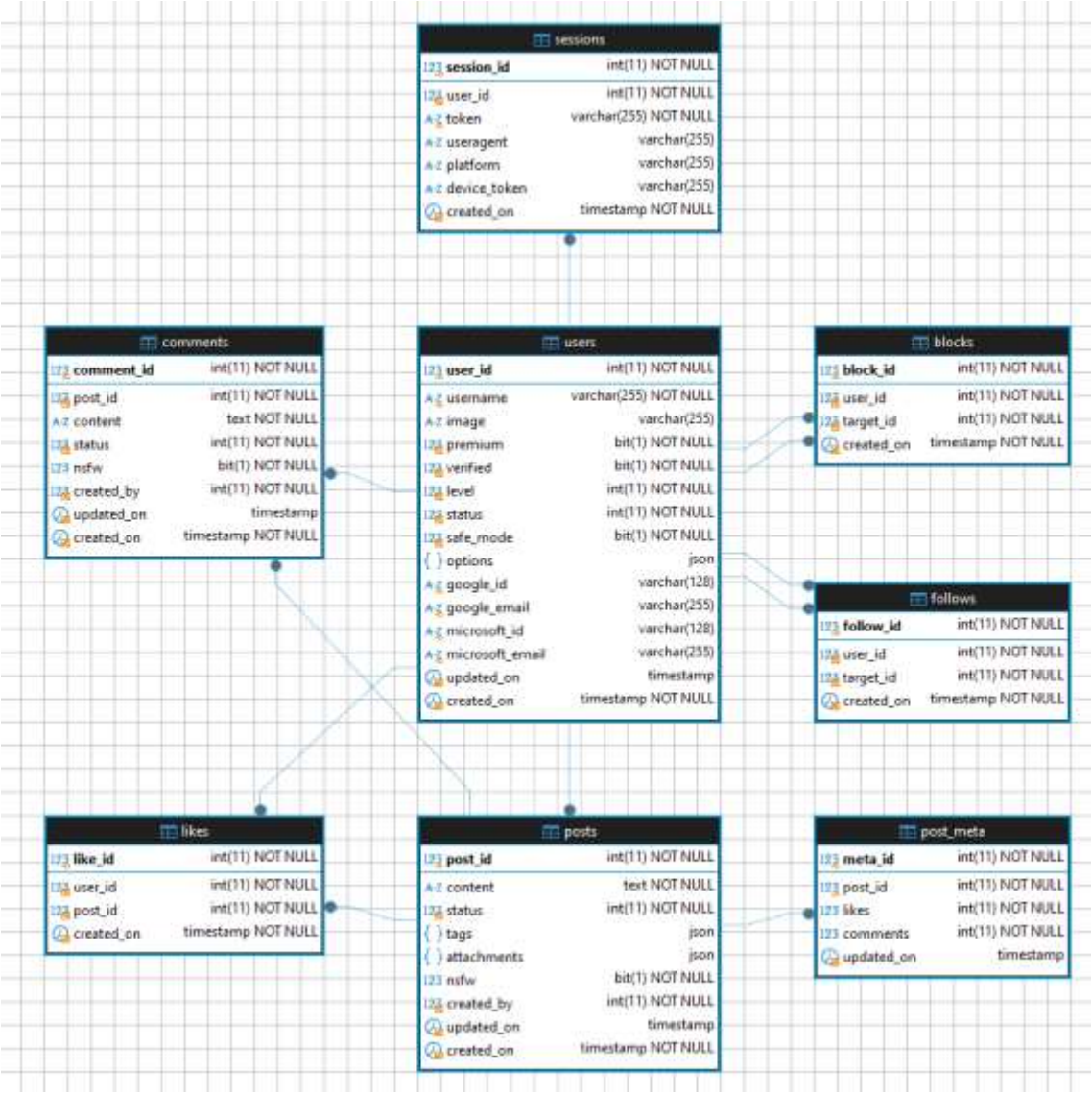
MySQL

Rakenduse andmeid hallatakse MySQL andmebaasis, mis on loodud süsteemi andmete talletamise ja haldamise aluseks. Andmebaasi struktuur koosneb kaheksast tabelist, mis katavad rakenduse põhifunktsionaalsuste vajadused.

Optimeerimise eesmärgil on andmebaasis rakendatud 18 indeksit, sealhulgas 10 unikaalset indeksit, et tagada kiire ja tõhus andmete otsing ning päringute töötlemine. Indeksite kasutamine vähendab päringute töötlemisaega ning parandab andmebaasi üldist jõudlust, eriti suurte andmehulkade korral.

Kõik tabelid on omavahel seotud välisvõtmete (*Foreign Key)* abil, mis tagavad andmete terviklikkuse ning välistavad duplikaatkirjete tekke. See võimaldab säilitada andmebaasi loogilist struktuuri ning toetab keerukate seoste ja päringute efektiivset töötlemist.

Joonisel 8 on esitatud andmebaasi ER (*Entity-Relationship*) mudel, mis näitab tabelite omavahelisi seoseid koos oluliste andmeväljadega.



Joonis 8. Rakenduse andmebaasi ER diagramm.

4.3 Tagarakendus

Tagarakendus on loodud, et pakkuda rakendusele tugevat ja paindlikku serveripoolset infrastruktuuri. Rakenduse loogika ja andmete töötlemine on üles ehitatud PHP-s, järgides REST (*Representational State Transfer)* API põhimõtteid, mis võimaldavad sujuvat ja modulaarset suhtlust esirakenduse ja tagarakenduse vahel [42].

PHP

Tagarakenduse loomisel ei kasutata täiendavaid raamistikke, mis võimaldab autoril säilitada täielikku kontrolli koodi struktuuri ja rakenduse arhitektuuri üle. See lähenemine

pakub paindlikkust arenduse igas etapis, võimaldades lahendusi täpselt kohandada loodava rakenduse spetsiifilistele nõuetele.

Kuigi täiendavaid raamistikke pole kasutusel, toetatakse arendust Composer [43] abil, mis lihtsustab teekide haldamist ja sõltuvuste lisamist. Lisaks rakendatakse PSR-4 (*PHP Standards Recommendation*) autolaadimise standardit, mis võimaldab klasside ja failide automaatset laadimist vastavalt nende asukohale. See parandab koodi organiseeritust ja muudab projekti lihtsamini laiendatavaks ning hooldatavaks.

PHP paindlikkus ja laialdased funktsionaalsused võimaldavad ehitada jõudlusele optimeeritud lahendusi, mis on samaaegselt lihtsad ja tõhusad. Rakenduse arendamisel on kasutatud kaasaegseid PHP parimaid praktikaid, sealhulgas sisendite valideerimist, et ennetada potentsiaalseid turvariske. Samuti on rakendatud kaitsemehhanismid, nagu SQL süstimisrünnakute [44] ennetamine, mis tagavad andmete turvalisuse ja süsteemi vastupidavuse võimalike rünnakute vastu. PHP-st tulenev lai ökosüsteem ja dokumentatsioon pakuvad tuge ka keerukamate funktsioonide juurutamisel.

REST

Tagarakendus on üles ehitatud REST API arhitektuurilise stiili alusel, mis võimaldab standardiseeritud ja tõhusat suhtlust kliendi ja serveri vahel. Kõik serveri funktsionaalsused on jaotatud erinevateks REST marsruutideks, mis hõlmavad kasutajate autentimist, postituste loomist ja haldamist, kommentaaride lisamist ning jälgimiste haldamist. Andmevahetus toimub JSON (*JavaScript Object Notation)* formaadis, mis on kergekaaluline ja sobib hästi kasutamiseks kaasaegsete esirakendustega. REST API järgib olekuta (*stateless*) arhitektuuri, kus kõik päringud on iseseisvad ega sõltu serveri eelnevast olekust. See muudab süsteemi skaleeritavamaks ja lihtsustab hooldust. Tabelis 8 on toodud näide ühe postitusega seotud otspunktidest.

Tabel 8. Postitusega seotud API otspunktid.

Päringu tüüp	**Otspunkt**	**Kirjeldus**
GET	/post/:post_id	Tagastab ühe postituse
PATCH	/post/:post_id	Ühe postituse muutmine
DELETE	/post/:post_id	Ühe postituse kustutamine

Apache

Tagarakendus töötab Apache veebiserveri abil, mis on tuntud oma stabiilsuse ja paindlikkuse poolest. Apache on seadistatud nii, et see toetaks REST otspunktide loomist, kasutades *mod_rewrite* moodulit. See võimaldab puhast ja kasutajasõbralikku URL struktuuri, kus andmetega seotud päringud suunatakse õigetele otspunktidele ilma ebavajalike parameetriteta. Joonisel 9 on kujutatud päringute ümbersuunamise konfiguratsioon, mis suunab kõik URL-id rakenduse peamisse sisendpunkti, milleks on index.php.

```
DirectoryIndex html/index.php

RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ html/index.php [L,QSA]
```

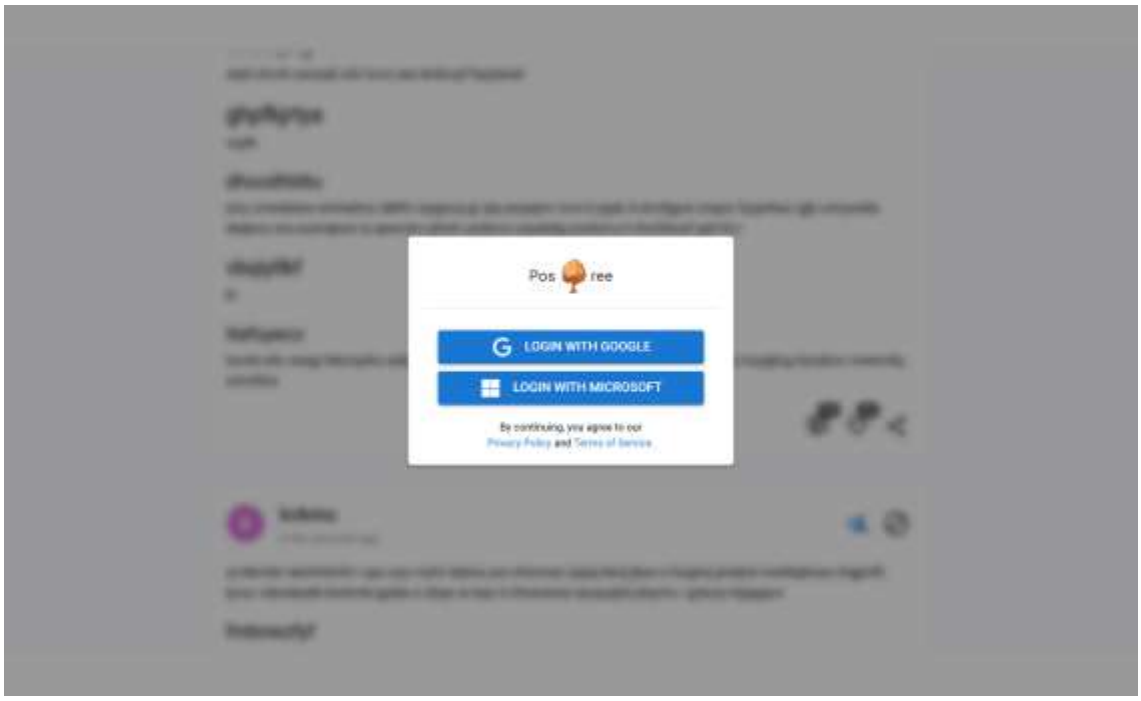
Joonis 9. Päringute ümbersuunamise konfiguratsioon.

4.4 Esirakendus

Esirakendus on loodud eesmärgiga pakkuda kasutajasõbralikku ja visuaalselt atraktiivset liidest, mis töötab sujuvalt erinevates keskkondades, sealhulgas veebis ja mobiilirakendustes. Valitud tehnoloogiad võimaldavad kiiret arendust ja ühtset koodibaasi, mis lihtsustab hooldust ning skaleeritavust. Lisaks pakuvad need tööriistad tuge seadmespetsiifilistele funktsioonidele, nagu tõuketeavitused ja lokaalne andmesalvestus, mis on oluline kaasaegsete hübriidrakenduste loomisel.

4.4.1 Sisselogimine

Sisselogimisvaate loomisel keskenduti lihtsusele ja sujuvale autentimisprotsessile. Kasutajatel on võimalus siseneda rakendusse Google või Microsoft kontoga, kasutades visuaalselt selget ja intuitiivset liidest. Minimalistlik kujundus ja selged juhised aitavad tagada, et sisselogimisprotsess oleks kiire ja arusaadav. Joonisel 10 on kujutatud rakenduse sisselogimise vaade.



Joonis 10. Rakenduse sisselogimise vaade.

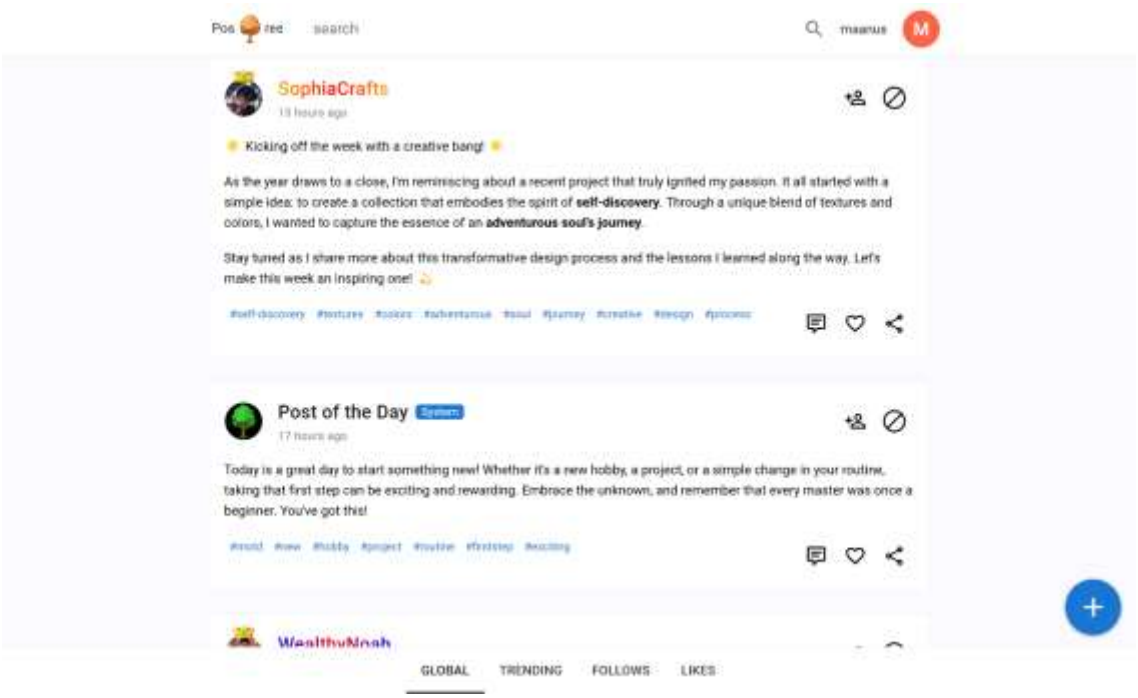
4.4.2 Põhivoog

Peamine sisuvoog on rakenduse keskne funktsioon, mis võimaldab kasutajatel avastada ja hallata sisu vastavalt nende eelistustele. Globaalne voog kuvab kõikide kasutajate postitusi, pakkudes võimalust avastada uusi ja huvipakkuvaid teemasid. Kasutajatel on võimalus blokeerida või jälgida teisi kasutajaid, muutes sisu nende jaoks paremini kontrollitavaks ja personaliseerituks.

Lisaks globaalsele voole on rakenduses saadaval mitmed teised vood:

- Populaarsed (*Trending*) postitused: algoritmil põhinev voog, mis kuvab kõige populaarsemad postitused vastavalt nende aktiivsusele ja kaasatusele.
- Jälgitavad (*Follows*): näitab ainult nende kasutajate postitusi, keda on jälgitud, pakkudes personaalsemat kogemust.
- Meeldivaks märgitud (*Liked*): kogumik postitustest, mis on meeldivaks märgitud, võimaldades kasutajatel hõlpsasti tagasi tulla neile olulise sisu juurde.

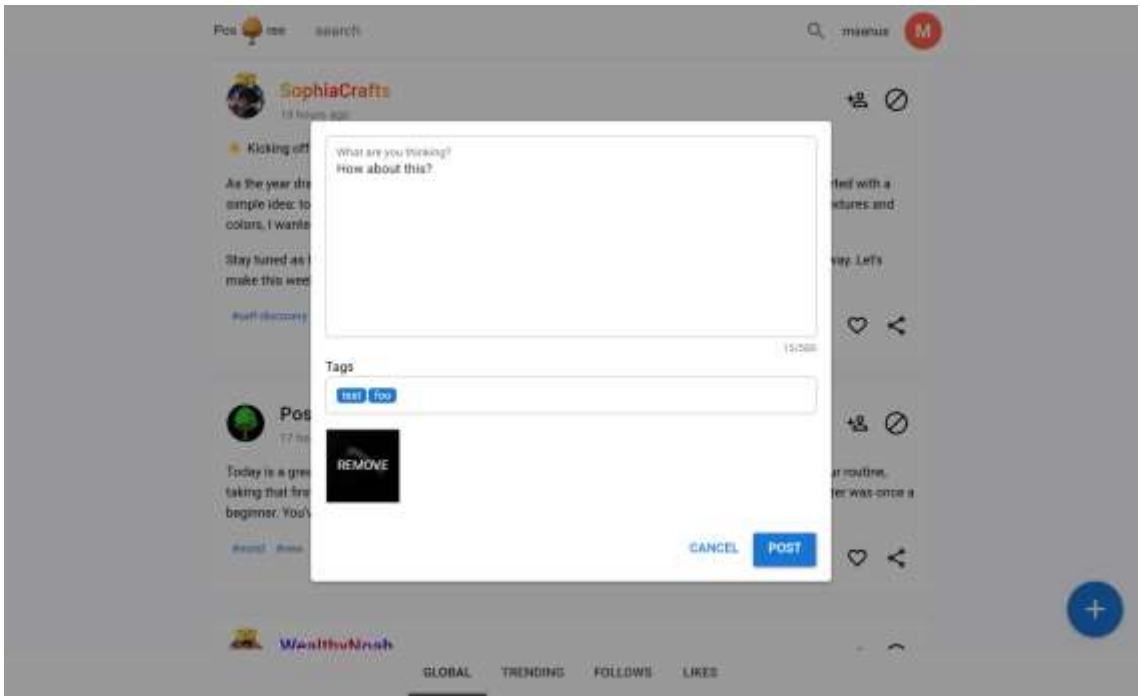
Jooniselt 11 on kujutatud rakenduse globaalne postituste voog.



Joonis 11. Rakenduse globaalne postituste voog.

4.4.3 Postituse loomine

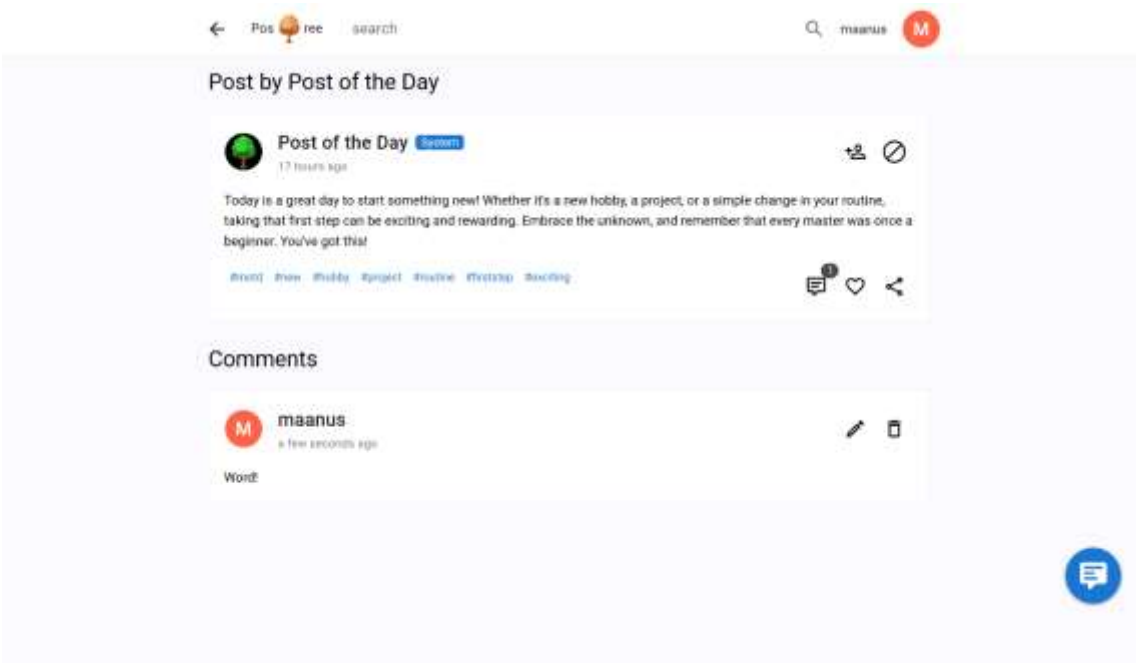
Postituse loomise vaade on loodud pakkuma lihtsat ja intuitiivset võimalust sisu jagamiseks. Kasutajad saavad lisada tekstipõhiseid postitusi ning täiendada seda pildi- või helifailiga. Postitusele saab lisada silte, mis aitavad sisu kategoriseerida ja laiema publikuni jõuda. Joonisel 12 on kuvatud uue postituse loomise vaade.



Joonis 12. Postituse loomise vaade.

4.4.4 Postituse vaade

Ühe postituse vaade pakub detailset ülevaadet konkreetsest postitusest ja sellega seotud arutelust. Vaates kuvatakse postituse sisu, sealhulgas tekst, pildid, helifailid ja seotud sildid, samuti postituse autor ning avaldamise aeg. Postituse all olev kommentaarium võimaldab kasutajatel arutada ja jagada mõtteid postituse teemal. Joonisel 13 on kujutatud ühe postituse ja kommentaariumi vaade.



Joonis 13. Ühe postituse ehk kommentaariumi vaade

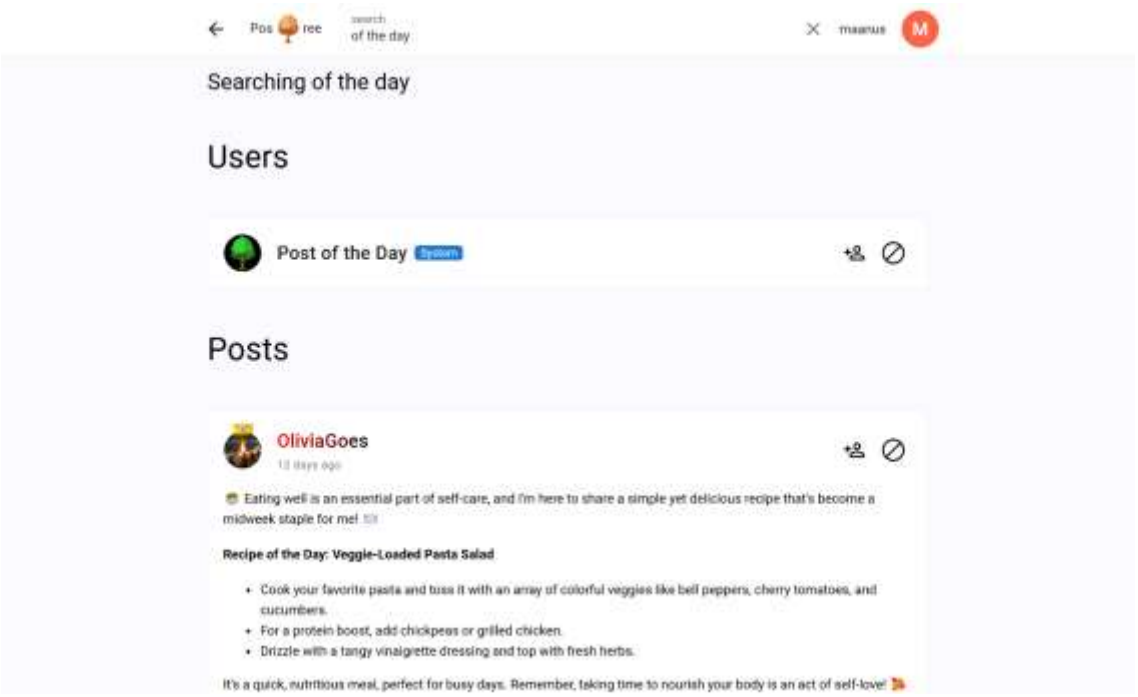
4.4.5 Otsingu tulemused

Otsingu tulemuste vaade võimaldab kasutajatel kiiresti leida sobivat sisu, kasutajaid ja silte, mis vastavad otsingupäringule.

Vaade jaguneb kaheks osaks:

- Kasutajad: kuvab kasutajakontod, mis vastavad otsingule. Kasutajad saavad otsingu kaudu jälgida, blokeerida või näha selle kasutaja loodud postitusi.
- Postitused: kuvab kõik postitused, mis vastavad otsingus sisestatud märksõnadele. Kasutajad saavad postitusi kohe vaadata, kommenteerida või meeldivaks märkida.

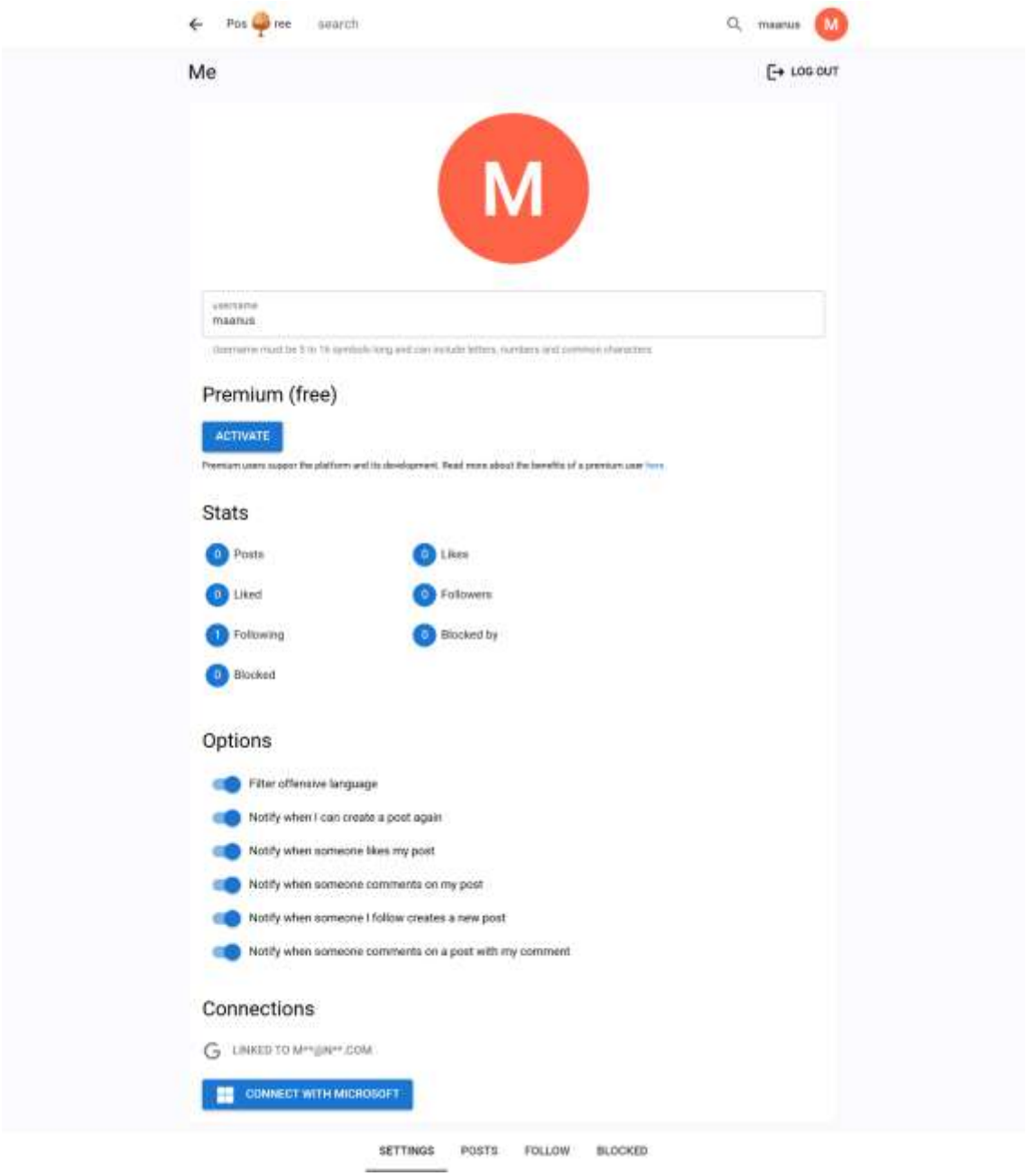
Joonisel 14 on kujutatud vaade rakenduse otsingu tulemustest.



Joonis 14. Rakenduse vaade otsingu tulemustest.

4.4.6 Profiili vaade

Profiilivaade võimaldab kasutajatel hallata oma isiklikku teavet, üleslaetud sisu ja jälgimissuhted. Kasutajatel on võimalus muuta profiilipilti, hallata teavitusseadeid ja vaadata oma tegevuslogi. Lisaks on profiili lehel kuvatud kontoga seotud statistika. Joonisel 15 on kuvatud rakenduse profiilivaade.



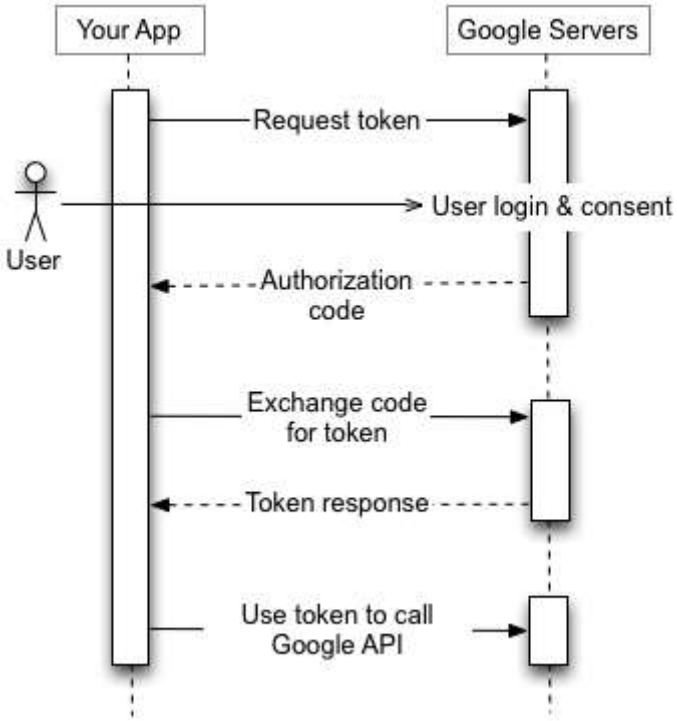
Joonis 15. Rakenduse profiili vaade

4.5 Kasutaja autentimine

Kasutajate autentimine rakenduses toimub OAuth 2.0 protokolli standardite järgi [45]. Seda standardit jälgivad Microsoft ja Google sisselogimisteenused. Selline lahendus vastab rakendusele seatud nõudele, kus sisselogimine peab olema võimalikult lihtne ja sujuv, pakkudes kasutajatele turvalist ja standardiseeritud viisi oma identiteedi kinnitamiseks.

Autentimisprotsess algab kasutaja sisselogimistaotlusega, mis suunatakse vastava teenuse (Google või Microsoft) autentimissüsteemi. Kasutaja sisestab seal oma kasutajanime ja parooli, mille järel teenus valideerib andmed ning suunab kasutaja tagasi loodud rakendusse, lisades päringu parameetrina autoriseerimise koodi (*authorization code*). See lühiajaline kood jõuab tagaserverisse, kus seda kasutatakse koos registreeritud salavõtmega (*secret key*), vahetades selle välja juurdepääsuvõtme (*access token*) vastu. See võti võimaldab tagarakendusel turvaliselt pärida sisselogimisteenustest kasutaja infot ning siduda selle rakenduses oleva kasutajaprofiiliga. Selline kaheetapiline vahetus tagab kõrgema turvalisuse, kuna tundlikku teavet töödeldakse ainult serveripoolselt ja see jääb kasutajaseadmest varjatuks.

Kuna autentimine toimub kolmandate osapoolte teenuste kaudu, ei pea rakendus ise salvestama kasutajate paroole ega haldama nende kontoinfot. Lisaks tagavad Google ja Microsoft kaheastmelise autentimise, mis suurendab turvalisust ilma rakenduse poolse lisaarenduseta. Joonisel 16 on kujutatud infovahetuse jada kasutades OAuth 2.0 standardit.

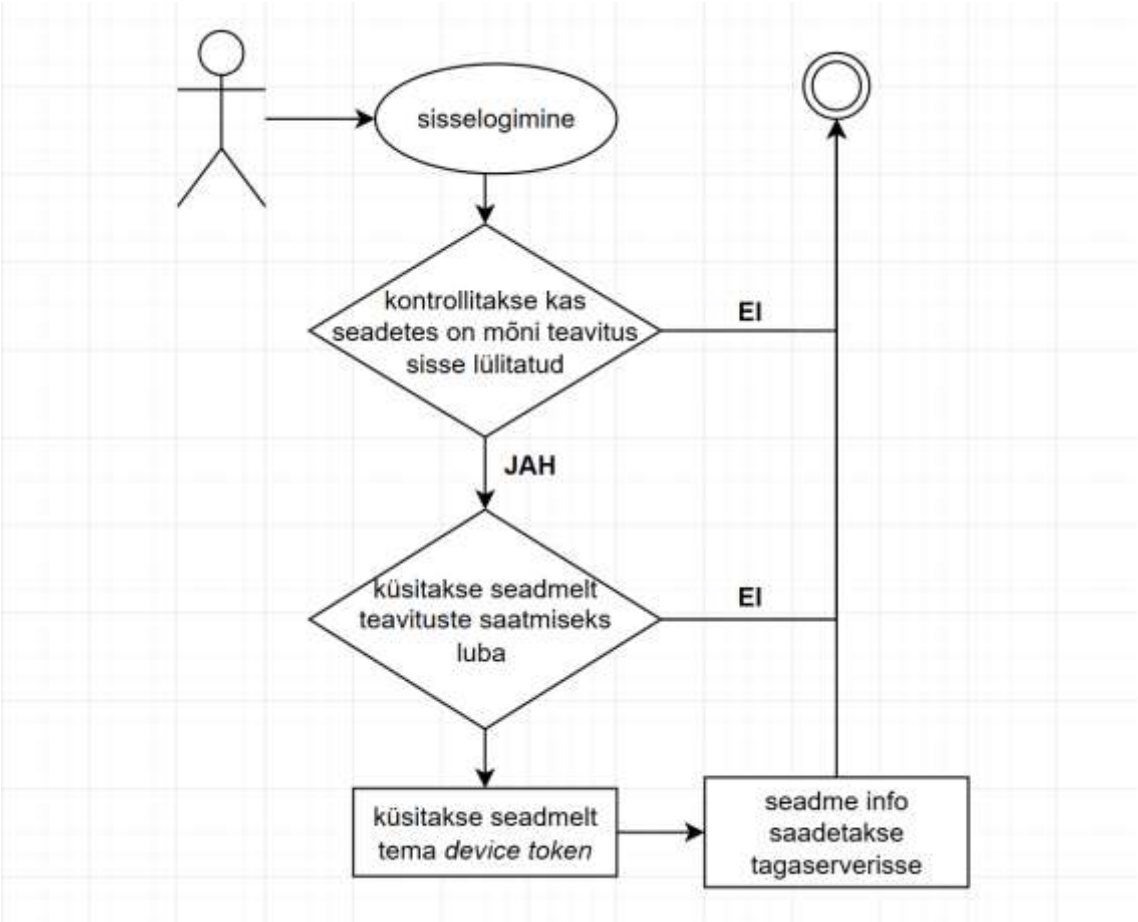


Joonis 16. OAuth2.0 skeem [46].

4.6 Tõuketeavitused

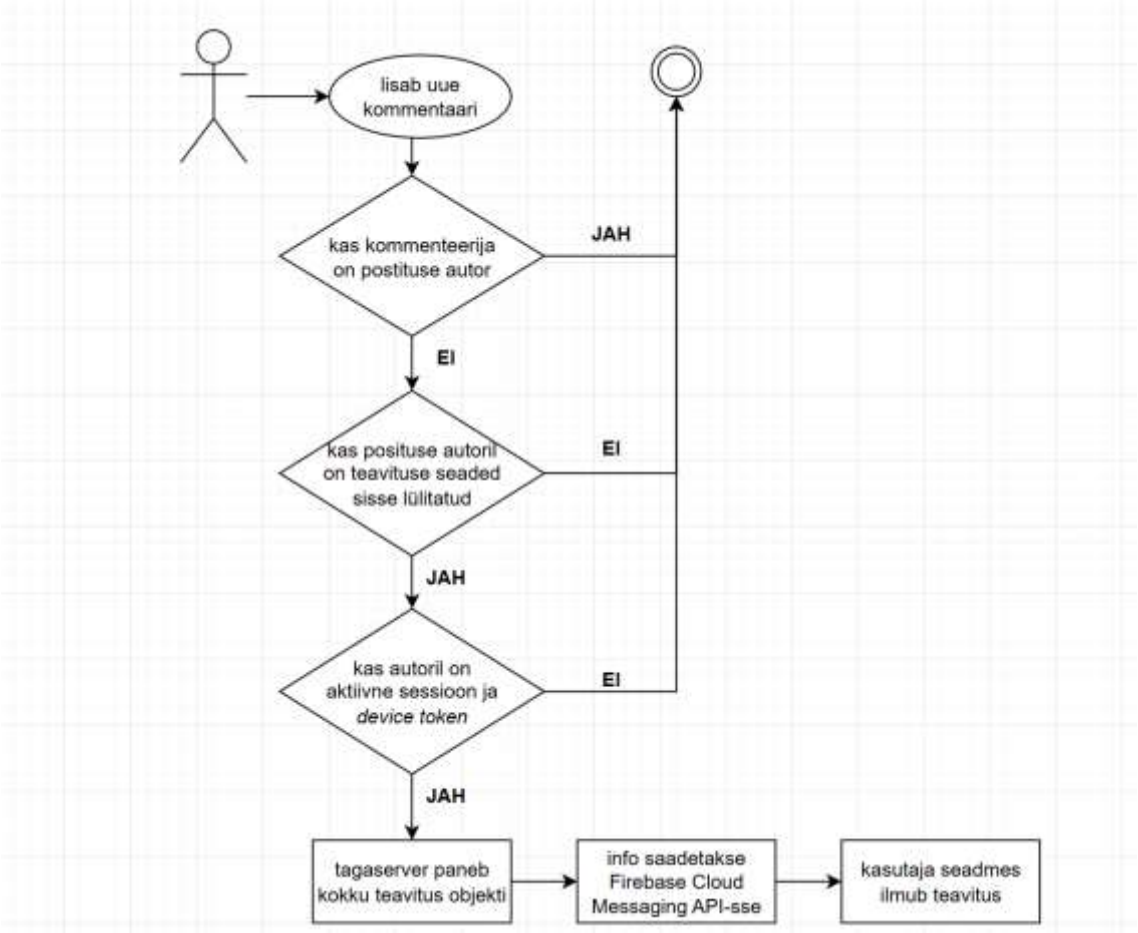
Rakenduse tõuketeavituste süsteem on üles ehitatud kasutades Firebase Cloud Messaging teenust, mis pakub tasuta ja lihtsasti integreeritavat lahendust. Firebase võimaldab rakendusel saata reaalajas teavitusi kasutajate seadmetele, aidates tõsta kasutajate kaasatust ja pakkuda kiiret teavitust oluliste sündmuste kohta. Firebase on Quasari raamistikuga täielikult integreeritud, mis võimaldab tõuketeavituste funktsionaalsust lisada minimaalse arenduskuluga. Lisaks on see optimeeritud töötama nii Android, iOS kui ka veebirakendustega, mis sobib ideaalselt hübriidse rakenduse konteksti.

Selleks et tõuketeavitusi saata on kõigepealt vaja kasutaja seadmest hankida seadme *token* mis saadetakse edasi tagaserverisse. Seal salvestatakse see andmebaasi ja seotakse kasutaja aktiivse sessiooniga, võimaldades personaalseid ja ajakohaseid teavitusi saata täpselt õigesse seadmesse. Joonisel 17 on kujutatud kasutaja seadme *token*-i küsimise jada.



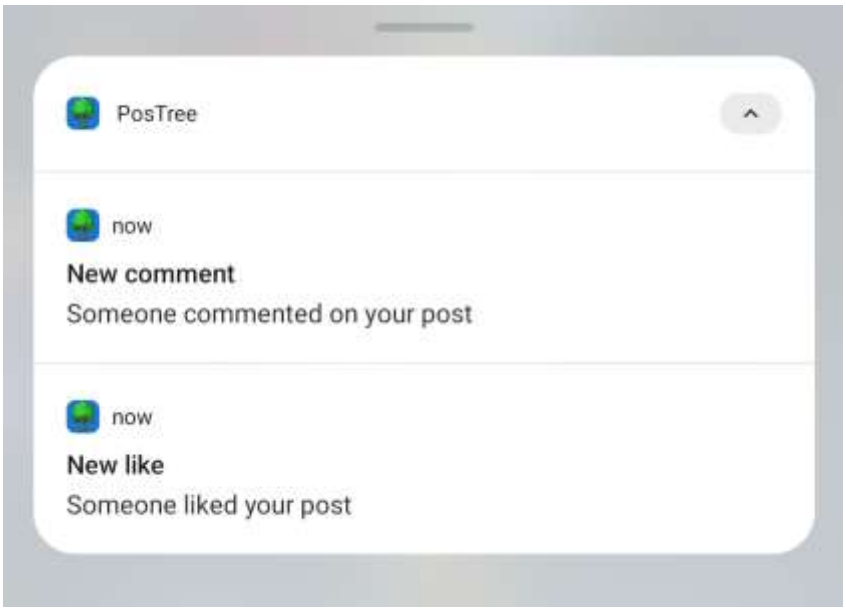
Joonis 17. Kasutaja seadme *token*-i pärimine.

Vastavalt päringutele ja käitatud koodile tagaserveris saadetakse kasutajatele tõuketeavitusi. Enne teavituse saatmist kontrollitakse erinevaid parameetreid, et selgitada välja millal ja kellele peaks teate saatma. Jooniselt 18 on kujutatud teekond teavituse saatmisest postituse autorile, kui postituse alla ilmub uus kommentaar.



Joonis 18. Uue kommentaari teavituse saatmise teekond.

Kõik saadetud teavitused töötavad justkui *out of the box* tänu Quasari ja Firebase integratsioonile, sealhulgas olles täielikult konfigureeritavad oma välimuse ja sisu poolest. Joonisel 19 on kujutad tõuketeavitus telefoni teavituste paneelis.



Joonis 19. Tõuketeavitused mobiilis.

4.7 Süvalink

Süvalink võimaldab rakendusel avada konkreetseid sisulehti või funktsioone otse URL-i kaudu, pakkudes kasutajatele sujuvat ja kiiret juurdepääsu soovitud sisule. Rakenduses on süvalinkimine realiseeritud Capacitor abil, mis toetab nii mobiilirakendusi kui ka veebiversioone. Praeguses arenduse etapis vaadeldakse ainult Android rakenduse seadistust, sest autoril puudub ligipääs Apple seadmele.

Süvalingi töö tagamiseks on veebiserverisse vaja luua *.well-known* kataloog, mis sisaldab seadistusefaili (nt Android puhul on selleks *assetlinks.json*). See fail kinnitab rakenduse ja domeeni seotust ning võimaldab süsteemil usaldusväärselt avada linke otse rakenduses. Lisaks Android rakenduse puhul on vaja seadistada seotud *intent-filter*. [35]

See funktsionaalsus lihtsustab kasutajate suunamist rakenduse kindlatesse osadesse näiteks teavituste, sotsiaalmeedia või e-kirjade kaudu, parandades kasutajakogemust ja suurendades rakenduse kasutusmugavust.

4.8 Juurutamine

Automaatse CI/CD lahenduse kasutamine vähendab käsitöö vajadust, tagades samal ajal, et rakendus püsib järjepidevalt funktsionaalne ja ajakohane. Juurutamisprotsessis kasutatakse Bitbucket Pipeline-i, mis pakub paindlikku ja hõlpsasti konfigureeritavat

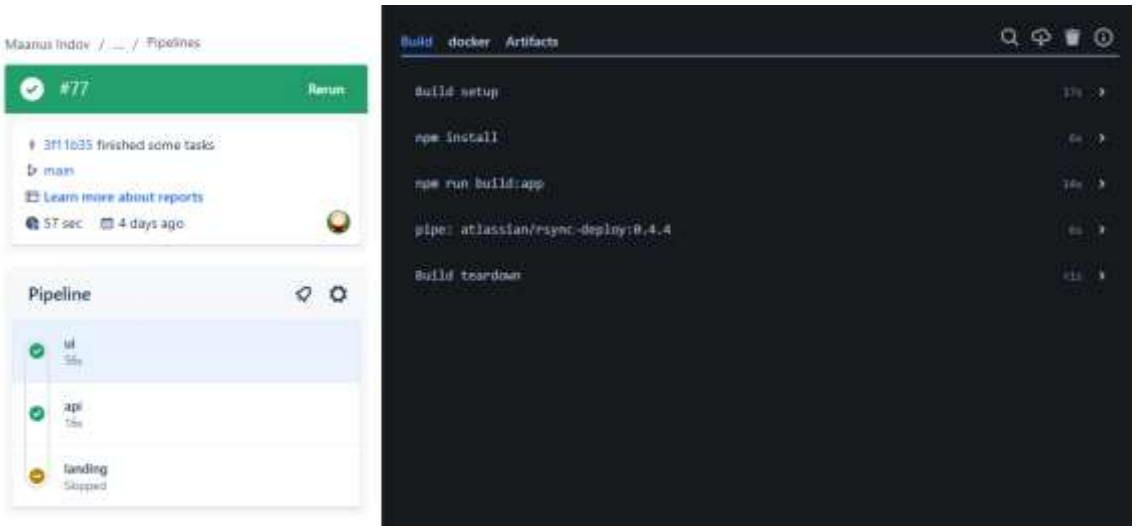
CI/CD lahendust. Bitbucket Pipeline võimaldab seadistada automatiseeritud töövooge, mis katavad nii esirakenduse kui ka tagarakenduse juurutamise. Suureks eeliseks on sisseehitatud tugi mitmetele tehnoloogiatele ja tööriistadele, nagu PHP, Python, Java, Ruby, NodeJS, SQLite, MySQL, Rsync, SSH jt [47]. Joonisel 20 on kujutatud lihtsustatud Pipeline konfiguratsioon mis jälgib versioonihalduses olevate failide muutust ja käitab vastavad käsud kui kindlad tingimused on täidetud.

```
image: composer:2.0

definitions:
  scripts:
    - step: &ui
        name: ui
        image:
          name: node:22.1.0
        caches:
          - node
        script:
          - npm install
          - npm run build:app
          - pipe: atlassian/rsync-deploy:0.4.4
            variables:
              USER: "$ssh_user"
              SERVER: "$host"
              REMOTE_PATH: "$root/www.mnz.ee/postreeapp"
              LOCAL_PATH: "./app/dist/spa/"
    - step: &api
        ...
pipelines:
  branches:
    main:
      - parallel:
          - step:
              <<: *ui
              condition:
                changesets:
                  includePaths:
                    - "app/src/**"
                    - "app/public/**"
          - step:
              <<: *api
                ...
```

Joonis 20. Pipeline konfiguratsioon lihtsustatud formaadis.

Sellises konfiguratsioonifailis määratakse kõik vajalikud sammud, sealhulgas koodi testimine, pakkimine ja lõppserverile juurutamine. Joonisel 21 on esitatud edukalt teostatud juurutusprotsess, mis hõlmab nii esirakendust kui ka tagarakendust.



Joonis 21. Õnnestunud juurutamine.

5 Tulemus

Tulemuste analüüsimiseks võrreldakse jaotises 3.2 välja toodud funktsionaalsusi arendatud rakenduse praeguse olekuga. Tabelis 9 on esitatud kõik seatud nõuded ning info nende täitmise kohta

Tabel 9. Nõuete täitmise võrdlus.

Nõue	**Täidetud**
Google kontoga sisselogimine	Jah
Microsoft kontoga sisselogimine	Jah
Platvormiülene otsing	Jah
Algoritmil põhinev voog	Jah
Kasutaja enda valikutest sõltuv voog	Jah
Meeldimised, jälgimised, blokeerimine	Jah
Tõuketeavitused	Jah
Profiilivaade, konto seaded	Jah
Postituste loomine, muutmine ja kustutamine	Jah
Piltide jagamine	Jah
Postituste kommenteerimine, nende muutmine ja kustutamine	Jah
Halva sõnastiku filter	Jah
Veebirakendus	Jah
Android rakendus	Jah
iOS rakendus	Ei
Avalik API dokumentatsioon	Jah
Üleslaetud meediafailide hoidmine CDN domeenil	Jah
Redis liidestus andmebaasi ees	Jah

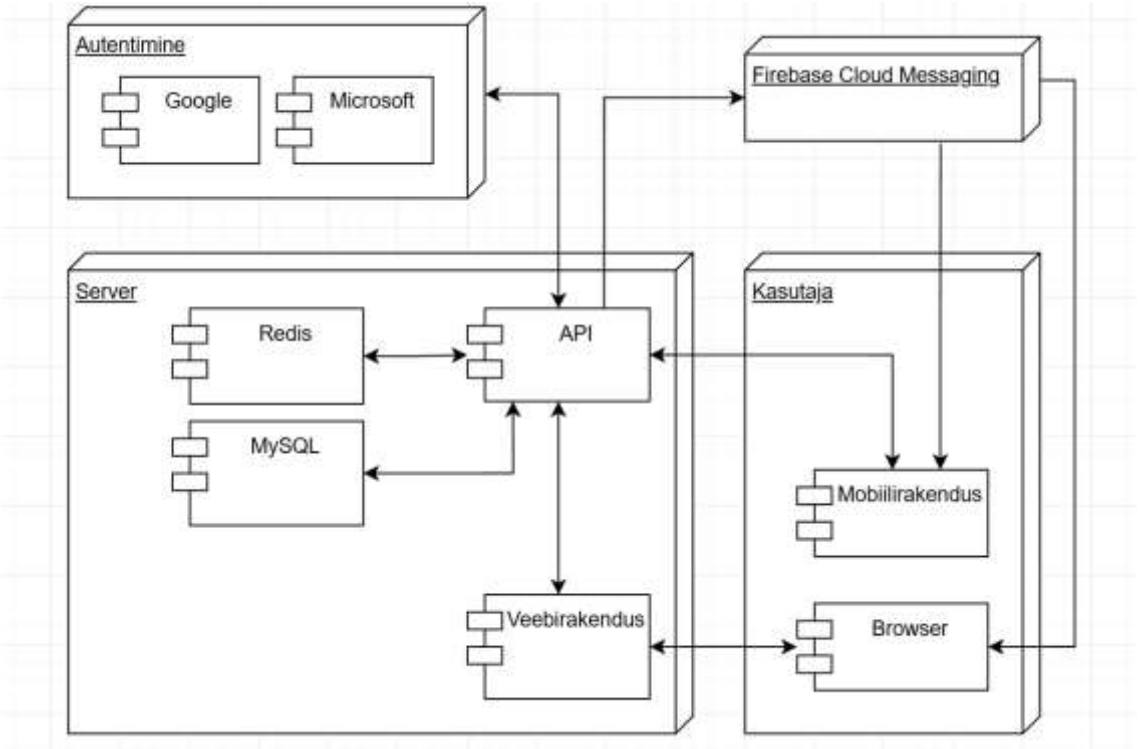
Hetkeseisuga jäi välja iOS versioon. Peamine põhjus on seotud Apple arendusnõuetega, kus iOS rakenduse loomine eeldab Apple seadmete (nt. iPhone) olemasolu, mida autoril polnud võimalik hankida. Lisaks on korduvad katsed saada juurdepääs Apple

arendajakontole ebaõnnestunud, mis takistab rakenduse testimist, sertifitseerimist ja levitamist App Store kaudu.

Kokkuvõttes on enamik nõudeid edukalt täidetud ning rakenduse põhifunktsionaalsus töötab vastavalt seatud eesmärkidele. iOS versiooni puudumine on siiski oluline piirang, mida saab lahendada tulevikus, kui vajalikud ressursid on tagatud.

5.1 Ülevaade rakendusest

Rakendus on loodud täitma kaasaegse sotsiaalmeediaplatvormi põhinõudeid, keskendudes lihtsusele ja sujuvale kasutajakogemusele. Rakendus on kättesaadav nii veebis kui ka Android-seadmetel, tagades mitmekülgse juurdepääsu. Joonisel 22 on kujutatud üldine rakenduse arhitektuur, mis selgitab, kuidas erinevad seadmed rakenduse osadega suhtlevad.



Joonis 22. Rakenduse üldine arhitektuur.

Rakenduse keskmeks on API, mis vahendab andmebaasis olevat infot veebirakenduse ja mobiilirakenduse vahel. Autentimine toimub turvaliselt väliste teenuste (Google, Microsoft) kaudu. Tõuketeadete edastamiseks kasutatakse FCM teenust, mis haldab teadete jõudmist kasutaja seadmesse.

5.1.1 Peamised omadused ja funktsionaalsus

Rakenduse esiliides on loodud olema intuitiivne ja hõlpsasti kasutatav, pakkudes sujuvat kogemust igapäevasteks sotsiaalmeediatoiminguteks. Kasutajad saavad platvormil sirvida sisu nii algoritmil põhinevas voos, mis prioriseerib asjakohast ja huvitavat sisu, kui ka nende enda valikutest lähtuvas voos, pakkudes personaalsemat kasutuskogemust.

Platvormiülene otsing võimaldab kiiresti ja tõhusalt leida konkreetset sisu, kasutajaid või teemasid, samal ajal kui sotsiaalsed interaktsioonid – nagu meeldimised, kommenteerimised, jagamised ning kasutajate jälgimine ja blokeerimine – tagavad täieliku funktsionaalsuse kaasaegse sotsiaalmeediaplatvormi ootustele vastamiseks.

Rakenduses on võetud kasutusele Firebase Cloud Messaging, mis annab kasutajatele tõuketeavituste abil kiire ülevaate olulistest sündmustest ja uuendustest. Sisselogimine Google ja Microsofti kontodega muudab autentimisprotsessi lihtsaks ja turvaliseks, välistades vajaduse salvestada kasutajate paroole ja võimaldades kolmanda osapoole poolt pakutavat kaheastmelist autentimist.

Rakendus toetab ka täiustatud funktsionaalsusi, nagu piltide automaatne orientatsiooni korrigeerimine EXIF (*Exchangeable Image File Format)* andmete alusel ja multi-touch žestidega piltide suurendamine ja liigutamine. Lisaks võimaldab rakenduse süvalingid kasutajatel avada kindlaid sektsioone otse väliste linkide kaudu, pakkudes sujuvat üleminekut erinevate keskkondade vahel ja ka sinna postituste jagamist.

5.1.2 Tehnoloogiline ülesehitus

Rakenduse tehnoloogiline infrastruktuur on ülesehitatud stabiilsusele ja tõhususele, kasutades kaasaegseid tööriistu ja raamistikke, mis võimaldavad hõlpsat arendust ja tulevikusüsteemi skaleerimist. Tagarakendus on loodud PHP-s, järgides REST API arhitektuuri, mis võimaldab sujuvat suhtlust esirakenduse ja serveri vahel. Andmete töötlemise kiiruse parandamiseks ja andmebaasi koormuse vähendamiseks on integreeritud Redis vahemälusüsteem, mis suudab sageli kasutatavaid andmeid korduvalt kiiresti serveerida ilma andmebaasiga ühendust võtmata.

Esirakenduse arenduses on kasutatud Quasar Framework-i, mis põhineb Vue3-l ja integreerib sujuvalt tööriistad nagu TypeScript ja PUG. TypeScript lisab arendusele

staatilise tüübituge, vähendades vigu ja parandades koodi usaldusväärsust, samas kui PUG muudab HTML mallide kirjutamise kiiremaks ja loetavamaks. Lisaks on raamistikus kasutusel Capacitorit, mis võimaldab ühe koodibaasi põhjal luua nii mobiili- kui ka veebirakenduse, lihtsustades hooldust ja vähendades arenduskulusid.

Autentimise ja turvalisuse aspektis toetab rakendus Google ja Microsofti autentimisteenuseid, pakkudes kasutajatele turvalist ja lihtsat sisselogimisprotsessi. Kolmanda osapoole standardite, nagu OAuth 2.0 ja kaheastmelise autentimise, kasutamine välistab vajaduse hoida rakenduses kasutajate tundlikku infot, vähendades sellega süsteemi turvariske.

Rakenduse turvalisust on veelgi suurendatud mitmete kaitsemehhanismidega, sealhulgas sisendite valideerimine ja SQL süstimisrünnakute ennetamine. Need meetmed tagavad süsteemi terviklikkuse ja kaitsevad kasutajate andmeid võimalike rünnakute eest.

Kogu infrastruktuur on loodud arvestusega, et see oleks hõlpsasti laiendatav ja tulevikus skaleeritav. Docker konteinerite kasutamine ja tarkvara jagamine sõltumatuteks komponentideks loovad tugeva aluse edasiseks arenduseks ja süsteemi kasvuks.

5.2 Kasutajatestid ja tagasiside

Esmane rakenduse versioon jagati sõpradele ja tuttavatele, et hinnata selle kasutatavust ja koguda olulist tagasisidet. Testijad katsetasid rakenduse põhifunktsionaalsusi nagu sisselogimine, postituste loomine ja sirvimine ja sisu interaktsioonid. Testimise käigus jälgis autor kasutajate tegevust ning märkis üles nende kogemused ja esitatud ettepanekud.

Kasutajatestid keskendusid peamiselt rakenduse esmakordsele kasutuskogemusele, sealhulgas sisselogimisprotsessile ja esialgsetele seadistustele. Google ja Microsofti sisselogimisvõimalused osutusid intuitiivseteks ja sujuvateks, vastates ootusele, et rakendusse sisenemine oleks lihtne ja probleemivaba. Kasutajad hindasid kõrgelt ka platvormi selget ülesehitust ja loogilist navigeerimist, mis võimaldas neil kiiresti rakenduse põhifunktsioonideni jõuda.

Testimise käigus ei esinenud olulisi takistusi, mis oleksid seganud rakenduse kasutamist. Siiski tõsteti esile mitmeid tähelepanekuid, mis olid seotud esiliidese disainivalikutega ja

ikoonide kasutamisega. Näiteks leidsid kasutajad, et mõne elemendi sümboolika või asukoht tekitab segadust, sest need ei andnud esmapilgul piisavalt selget konteksti. Lisaks jagasid testijad mitmeid ideid ja UX (*User Experience*) ettepanekuid, mis keskendusid olemasoleva kasutajakogemuse täiendamisele. Sageli esitati alternatiivseid lahendusi või küsimusi stiilis „kas siin võiks nii ka teha saada“, mis viitas kasutajate loomupärasele ootusele suurema paindlikkuse ja funktsionaalsuse järele.

Lisaks tõid testijad esile vajaduse tõuketeavituste järele, eriti soovina saada märguandeid, mis julgustaksid kasutajaid platvormile naasma. Selleks, et vältida liigset infomüra ja pakkuda kasutajatele suuremat kontrolli, lisati kasutajakonto seadetesse võimalus iga teavituse tüüpi eraldi välja lülitada.

5.3 Edasiarendus

Rakenduse edasine arendus keskendub eelkõige platvormi funktsionaalsuse laiendamisele, skaleerimisvõimele ning täiendavate ärimudelite loomisele, et tagada jätkusuutlikkus ja laiem kasutajaskond. Järgmised sammud on suunatud rakenduse kättesaadavuse parandamisele, uute funktsioonide lisamisele ning tehnilise infrastruktuuri tugevdamisele.

5.3.1 Rakenduste poed ja iOS tugi

Üks prioriteetidest on rakenduse kättesaadavaks tegemine Apple *App Store*-s, et täita iOS kasutajate ootused ja laiendada sihtrühma. Selleks on vajalik arendajakonto loomine, mis võimaldab rakenduse testimist ja avaldamist. Lisaks tuleb rakendus kohandada vastavalt App Store nõuetele, sealhulgas läbida sertifitseerimis- ja kvaliteedikontrolliprotsessid. iOS toe rakendamine nõuab ka testimist Apple seadmetel, et tagada rakenduse täielik ühilduvus ja stabiilsus. Androidi *Play Store* jaoks on rakendus valmis.

5.3.2 Premium mudel ja monetiseerimine

Platvormi jätkusuutlikkuse tagamiseks plaanitakse rakendusele lisada tasulise (premium*)* liikmesuse mudel, mis pakub kasutajatele eksklusiivseid funktsioone. Premium tellimuse raames saavad kasutajad kasutada mitmeid lisavõimalusi, nagu:

- Ühetähelised kasutajanimed: Premium kasutajad saavad valida lühemaid kasutajanimesid, mida mitte premium kasutajad ei saa kasutada (tavakasutajal minimaalne 3 tähte).
- Värviline kasutajanimi: Premium kasutajad saavad valida oma kasutajanimedele värve, samas kui mitte premium kasutajate nimed kuvatakse mustana.
- Avatari kaunistav kroon: Premium-kasutajate profiilipilte kaunistab eksklusiivne kroon, mis eristab neid teistest kasutajatest.
- Pikemad postitused: Premium kasutajatele on lubatud postitada kuni 1000 tähemärki, samas kui mitte premium kasutajate postitused on piiratud 500 tähemärgiga.

Need edasiarendused aitavad suurendada rakenduse atraktiivsust nii tasuta kui ka tasulistele kasutajatele, pakkudes samas paindlikku ja elujõulist ärimudelit.

5.3.3 Automaatne testimine

Rakenduse arenduse edenedes suureneb vajadus automaatsete testimislahenduste järele, et tagada uute funktsioonide kvaliteet ja olemasolevate süsteemide stabiilsus. Automaatne testimine aitab tuvastada võimalikke probleeme varases etapis, minimeerides vigade jõudmise tootmiskeskkonda ja kiirendades uute funktsioonide juurutamist. See lähenemine vähendab arendusriske ning võimaldab säästa aega ja ressursse.

Hetkeseisuga, kui rakenduse beeta versioon on veel arenduses, ei ole automaatseid teste loodud, kuna toode on pidevas muutumises. Üksikarendajana on autor keskendunud toote valmimisele kindlas ajaraamis, mistõttu on testimise rõhk olnud manuaalsetel meetoditel.

Tulevikus on plaanis lisada automaatne testimine, mis hõlmab endas kahte tüüpi teste milleks on: üksustestid ja integreerimistestid. Üksustestid keskenduvad väikeste koodilõikude, näiteks funktsioonide või klasside, eraldi testimisele, et tuvastada vigu juba arenduse varajases etapis. Integreerimistestid kontrollivad seevastu erinevate süsteemikomponentide koostööd ja terviklikku toimimist, et tagada, et need töötavad ootuspäraselt ka koos. Mõlema testimismeetodi rakendamine aitab tagada rakenduse kvaliteedi ja stabiilsuse kasvavas arendusprotsessis. [48]

6 Kokkuvõte

Lõputöö eesmärk oli luua hübriid sotsiaalmeediarakendus, mille eripäraks on postituste piiramise süsteem, soodustamaks kvaliteetsemat ja läbimõeldumat sisu. Rakendus vastab kaasaegsetele nõuetele ja on kättesaadav veebis ning Android rakenduste poes.

Rakendus on loodud kasutades kaasaegseid ja populaarseid raamistikke, võttes aluseks sotsiaalmeediaplatvormide kasutusmustreid, et pakkuda sujuvat kasutajakogemust ja optimeeritud funktsionaalsust. Rakenduse esiliides on ehitatud Quasari raamistikul, mis tugineb Vue3-le, ning tagaserver on programmeeritud PHP keeles. Andmeid hallatakse MySQL andmebaasis, millele lisandub kiire andmevoo tagamiseks mälupõhine andmehoidja Redis.

Valminud rakendus täidab enamikku seatud nõuetest, pakkudes sujuvat autentimist Google ja Microsofti kontodega, algoritmil põhinevat sisuvoogu, süvalingi tuge ja täiustatud kasutajakogemust. Kuigi iOS tugi jäi realiseerimata Apple arendustingimuste tõttu, on süsteem valmis edaspidiseks laienemiseks.

Kasutajatestid kinnitasid rakenduse tugevusi ja pakkusid väärtuslikku tagasisidet, mis võimaldas parandada kasutajakogemust. Tulevikus on plaanis rakendust laiendada, lisades iOS toe, premium teenused ja automaattestimine, et tagada kvaliteet ja jätkusuutlikkus platvormi arengul.

Rakendus on edukalt realiseeritud, luues tugeva tehnoloogilise aluse edasiseks arenduseks ja skaleerimiseks. Töö tõestab, et läbimõeldud tehnoloogilised valikud võimaldavad luua kvaliteetseid lahendusi ka piiratud ressurssidega.

Kasutatud kirjandus

[1] Statista, „Most popular social networks worldwide as of April 2024, by number of monthly active users,“ [Võrgumaterjal]. Available: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/. [Kasutatud 15 11 2024].

[2] K. Masterson, „How Many Instagram Posts Per Day for Success in 2024? An Expert Guide,“ 8 04 2024. [Võrgumaterjal]. Available: https://www.33rdsquare.com/how-many-instagram-posts-per-day-for-success-in-2023-an-expert-guide. [Kasutatud 01 11 2024].

[3] K. G. B. S. Manuel Gomez Rodriguez, „Quantifying Information Overload in Social Media and its Impact on Social Contagions,“ [Võrgumaterjal]. Available: https://arxiv.org/abs/1403.6838. [Kasutatud 19 11 2024].

[4] N. B. W. R. I. G. C. J. C. S. Chathika Gunaratne, „The Effects of Information Overload on Online Conversation Dynamics,“ [Võrgumaterjal]. Available: https://arxiv.org/abs/1910.09686. [Kasutatud 15 11 2024].

[5] M. G. Nasser Kehtarnavaz, Real-Time Image and Video Processing, Research to Reality, 2008.

[6] Enkonix, „Functional vs Non-Functional requirements,“ [Võrgumaterjal]. Available: https://enkonix.com/blog/functional-requirements-vs-non-functional/. [Kasutatud 02 12 2024].

[7] Zone.ee, „Virtuaalserverite detailne võrdlus,“ [Võrgumaterjal]. Available: https://www.zone.ee/et/virtuaalserver/vordlus/. [Kasutatud 02 11 2024].

[8] AWS, „AWS Pricing,“ [Võrgumaterjal]. Available: https://aws.amazon.com/pricing/?nc2=h_ql_pr_ln. [Kasutatud 06 12 2024].

[9] Microsoft, „Azure Pricing Overview,“ [Võrgumaterjal]. Available: https://azure.microsoft.com/en-us/pricing. [Kasutatud 06 12 2024].

[10] DigitalOcean, „Budget-Friendly Cloud Server Pricing,“ [Võrgumaterjal]. Available: https://www.digitalocean.com/pricing. [Kasutatud 06 12 2024].

[11] Zone.ee, „Privaatserveri KKK,“ 22 05 2024. [Võrgumaterjal]. Available: https://help.zone.eu/kb/privaatserveri-kkk/. [Kasutatud 05 01 2024].

[12] AWS, „Regions and Availability Zones,“ [Võrgumaterjal]. Available: https://aws.amazon.com/about-aws/global-infrastructure/regions_az/. [Kasutatud 03 01 2025].

[13] Microsoft, „Regions for virtual machines in Azure,“ 22 08 224. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/azure/virtual-machines/regions. [Kasutatud 03 01 2025].

[14] K. Sen, „DigitalOcean vs Microsoft Azure,“ 18 11 2024. [Võrgumaterjal]. Available: https://www.upguard.com/blog/azure-vs-digitalocean. [Kasutatud 01 04 2025].

[15] S. Overflow, „Technology | 2024 Stack Overflow Developer Survey,“ [Võrgumaterjal]. Available: https://survey.stackoverflow.co/2024/technolog.

[16] altexsoft, „Comparing Database Management Systems,“ [Võrgumaterjal]. Available: https://www.altexsoft.com/blog/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/. [Kasutatud 25 12 2024].

[17] GeeksforGeeks, „ACID Model vs BASE Model For Database,“ [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/acid-model-vs-base-model-for-database/. [Kasutatud 23 12 2024].

[18] SQLite, „Appropriate Uses For SQLite,“ [Võrgumaterjal]. Available: https://www.sqlite.org/whentouse.html. [Kasutatud 05 01 2025].

[19] M. R. T. Valeriu Crudu, „What are the differences between MongoDB and traditional relational databases,“ 15 08 2024. [Võrgumaterjal]. Available: https://moldstud.com/articles/p-what-are-the-differences-between-mongodb-and-traditional-relational-databases. [Kasutatud 01 04 2025].

[20] T. Facts, „2924s Fastest Web Servers for REST APIs,“ [Võrgumaterjal]. Available: https://medium.com/@hiadeveloper/2024s-fastest-web-servers-for-rest-apis-node-js-vs-go-vs-rust-vs-c-net-benchmark-665d8efd2f44. [Kasutatud 04 12 2024].

[21] E. Drosopoulou, „Rust vs. Go: Choosing the Right Language for High-Performance Systems,“ 27 12 2024. [Võrgumaterjal]. Available: https://www.javacodegeeks.com/2024/12/rust-vs-go-choosing-the-right-language-for-high-performance-systems. [Kasutatud 04 01 2025].

[22] Microsoft, „.NET Framework system requirements,“ 26 04 2024. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/framework/get-started/system-requirements. [Kasutatud 04 01 2025].

[23] Zone, „ZFPM (Zone FastCGIU Process Manager),“ [Võrgumaterjal]. Available: https://help.zone.eu/en/kb/zfpm-zone-fastcgi-process-manager-2/. [Kasutatud 03 12 2024].

[24] I. Luchaninov, „Cross-Platform Mobile App Development,“ [Võrgumaterjal]. Available: https://mobidev.biz/blog/cross-platform-mobile-development-frameworks-comparison. [Kasutatud 23 12 2024].

[25] J. Wallis, „Top 5 Underrated Native Mobile App Development Frameworks,“ [Võrgumaterjal]. Available: https://intuji.com/underrated-native-mobile-app-development-frameworks/. [Kasutatud 23 12 2024].

[26] JSCrambler, „Top 12 Frameworks for Hybrid Apps,“ [Võrgumaterjal]. Available: https://jscrambler.com/blog/12-frameworks-for-mobile-hybrid-apps. [Kasutatud 20 12 2024].

[27] M. Platforms, „Setup Up Your Environment - React Native,“ [Võrgumaterjal]. Available: https://reactnative.dev/docs/set-up-your-environment. [Kasutatud 22 12 2024].

[28] Ionic, „Ionic Vue Overview,“ [Võrgumaterjal]. Available: https://ionicframework.com/docs/vue/overview. [Kasutatud 04 01 2024].

[29] Quasar, „Why Quasar?,“ [Võrgumaterjal]. Available: https://quasar.dev/introduction-to-quasar. [Kasutatud 04 01 2025].

[30] Quasar, „Quasar Components,“ [Võrgumaterjal]. Available: https://quasar.dev/components/. [Kasutatud 18 11 2024].

[31] Quasar, „Quasar License,“ [Võrgumaterjal]. Available: https://github.com/quasarframework/quasar/blob/dev/LICENSE. [Kasutatud 2024 12 20].

[32] GeeksforGeeks, „Difference between Redis and Memcached,“ 18 06 2024. [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/difference-between-redis-and-memcached/. [Kasutatud 05 01 2025].

[33] Vue.js, „Using Vue with TypeScript,“ [Võrgumaterjal]. Available: https://vuejs.org/guide/typescript/overview. [Kasutatud 05 01 2025].

[34] Educative. [Võrgumaterjal]. Available: https://www.educative.io/answers/html-to-pug. [Kasutatud 04 01 2025].

[35] Capacitor, „Deep Links | Capacitor,“ [Võrgumaterjal]. Available: https://capacitorjs.com/docs/guides/deep-links. [Kasutatud 05 12 2024].

[36] Ionic, „Capacitor - Cross-platform Native Runtime for Web Apps,“ [Võrgumaterjal]. Available: https://capacitorjs.com/docs.

[37] Capacitor, „Official Plugins,“ [Võrgumaterjal]. Available: https://capacitorjs.com/docs/apis. [Kasutatud 05 01 2024].

[38] Atlassian, „ Enterprise Strategy and Planning Solutions | Atlassian,“ [Võrgumaterjal]. Available: https://www.atlassian.com/software/jira/align/solutions. [Kasutatud 20 11 2024].

[39] U. L. John Rofrano, „Benefits of Continuous Integration,“ [Võrgumaterjal]. Available: https://www.coursera.org/lecture/continuous-integration-and-continuous-delivery-ci-cd/benefits-of-continuous-integration-ci-zqLTF. [Kasutatud 24 12 2024].

[40] T. Labs, „Traefik Proxy Documentation,“ [Võrgumaterjal]. Available: https://doc.traefik.io/traefik/. [Kasutatud 20 11 2024].

[41] mkcert, „mkcert.org,“ [Võrgumaterjal]. Available: https://mkcert.org/. [Kasutatud 06 12 2024].

[42] L. J. Mitchell, PHP Web Services: APIs for the Modern Web, O'Reilly Media, 2013.

[43] J. B. Nils Aderman, „Composer,“ [Võrgumaterjal]. Available: https://getcomposer.org/. [Kasutatud 25 12 2024].

[44] Invicti, „SQL Injection Cheat Sheet,“ [Võrgumaterjal]. Available: https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/. [Kasutatud 08 11 2024].

[45] D. H. Ed, „The OAuth 2.0 Authorization Framework,“ 10 2012. [Võrgumaterjal]. Available: https://www.rfc-editor.org/rfc/pdfrfc/rfc6749.txt.pdf.

[46] Google, „Using Oauth 2.0,“ [Võrgumaterjal]. Available: https://developers.google.com/identity/protocols/oauth2. [Kasutatud 05 12 2024].

[47] Atlassian, „Use Pipelines in different software languages,“ [Võrgumaterjal]. Available: https://support.atlassian.com/bitbucket-cloud/docs/use-pipelines-in-different-software-languages/. [Kasutatud 07 11 2024].

[48] Codefresh, „Unit Testing vs. Integration Testing,“ [Võrgumaterjal]. Available: https://codefresh.io/learn/unit-testing/unit-testing-vs-integration-testing-5-key-differences-and-why-you-need-both. [Kasutatud 03 12 2024].

[49] Google, „Using OAuth 2.0 to Access Google APIs,“ [Võrgumaterjal]. Available: https://developers.google.com/identity/protocols/oauth2. [Kasutatud 10 11 2024].

[50] Microsoft, „OAuth 2.0 authorization with Microsoft Entra ID,“ [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/entra/architecture/auth-oauth2. [Kasutatud 10 11 2024].

[51] Quasar, „Quasar Framework,“ [Võrgumaterjal]. Available: https://quasar.dev/. [Kasutatud 23 12 2024].

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

Mina, Maanus Indov

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Sotsiaalmeedia platvorm PosTree infomüra vähendamiseks“, mille juhendaja on Kristiina Hakk
   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.

05.01.2025

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.