TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Janar Takis 205954 IAIB
Sven Erko Jaroševitš 213097 IAIB
Karl Erik Seeder 213095 IAIB

VEEBIPÕHINE FINANTSANALÜÜSI PLATVORM

Bakalaureusetöö

Juhendaja: Siim Rebane
Külalisõppejõud

Kaasjuhendaja: Gert Kanter
Vanemlektor

Tallinn 2024

Autorideklaratsioon

Kinnitame, et oleme 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.

Autorid: Janar Takis, Sven Erko Jaroševitš ja Karl Erik Seeder

27.05.2024

Annotatsioon

Viimaste aastate kõrge inflatsioon on sundinud inimesi otsima traditsioonilistele hoiustamisviisidele alternatiive, muutes pikaajalise investeerimine kui ka lühiajalise kauplemise üha populaarsemaks.

Samas ei ole olemasolevad finantsplatvormid suutnud pakkuda kasutajatele piisavalt kasutajasõbralikku ja terviklikku lahendust, nõudes informeeritud investeerimisotsuste tegemiseks aega ja tihtipeale ka raha erinevate platvormide ja tööriistade uurimiseks.

Antud lõputöö raames sidusid töö autorid omavahel teadmised tarkvaraarendusest matemaatilise analüüsi ja finantsturgudega. Eesmärk oli luua nii algajatele kui edasijõudnutele kasutatav veebipõhine finantsanalüüsi platvorm, andes kasutajale tööriistad, et teha targemaid investeerimisotsuseid.

Lõputöö korraldati grupitööna koostöös kliendiga. Töö tulemusena valmis kliendi soovidega kooskõlas veebirakenduse prototüüp https://investingedge.pro, mis on kasutatav nii lauarvutis kui mobiiltelefonis. Veebirakendus võimaldab sisse logitud kasutajatel suhelda vestlusrobotiga, küsides sellelt investeerimisalaseid nõuandeid ja infot erinevate varade kohta. Samuti pakub rakendus võimalust jälgida varade ajaloolist hinnainfot, tutvuda aktsiate fundamentaalnäitajatega ning konfigureerida ja jagada ostu- või müügisignaale. Prototüüp täidab valdava enamuse esialgselt püstitatud funktsionaalsetest nõuetest ning kõik mittefunktsionaalseid nõuded.

Teoreetiline osa kirjeldab valminud prototüübi arendusprotsessi. Töö käigus analüüsiti olemasolevaid lahendusi, kaardistati kliendi soovid, määrati prototüübi ulatus ning uuriti tehnoloogiaid, mis vastaksid töö nõuetele. Valminud prototüübil analüüsiti nõuete täitmist ning hinnati selle sobivust probleemi lahendamiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 43 leheküljel, 7 peatükki ja 27 joonist.

Abstract

Web-based financial analysis platform

Recent years of high inflation have forced people to seek alternatives to traditional savings methods, making both long-term investing and short-term trading increasingly popular. However, existing financial platforms have not been able to provide users with a sufficiently user-friendly and comprehensive solution, requiring time and often money to explore different platforms and tools for making informed investment decisions.

In this thesis, the authors combined their knowledge of software development with mathematical analysis and financial markets. The aim was to create a web-based financial analysis platform that can be used by both beginners and advanced users, providing the user with tools to make smarter investment decisions.

The thesis was organized as a group work in cooperation with a client. As a result of the work, a web application prototype https://investingedge.pro was created in accordance with the client's wishes, which can be used on both a desktop and mobile devices.

The web application functionalities are grouped into three main categories: market analysis, portfolio management, and investment tools. The market analysis category includes showing and visualizing the current state of the markets, collecting news, visualizing fundamental indicators, and aggregating buy and sell indicators. The portfolio management category allows the user to view the historical performance of the portfolio. The investment tools category includes educational resources and an AI chatbot, allowing users to interact with the chatbot by asking it investment-related advice or information about different assets. Not all functional requirements were fully met, nevertheless, both the authors and the client consider the work as successful.

The theoretical part describes the development process of the completed prototype. Existing solutions were analyzed, the client's wishes were mapped, the scope of the prototype was determined, and technologies that would meet the requirements of the work were studied.

The thesis is written in Estonian and is 43 pages long, including 7 chapters and 27 figures.

Lühendite ja mõistete sõnastik

ACID	*Atomicity, Consistency, Isolation, Durability,* andmebaasi tehingute omadused, mis tagavad andmete terviklikkuse
Aegrida	Andmete kogum, mis on järjestatud ajaliselt ning tavaliselt seotud kindla ajaperioodiga. Aegridu kasutatakse sageli trendide analüüsimiseks.
Agiilne arendus	*Agile development* ,tarkvaraarenduse meetod, mis põhineb iteratiivsel ja järk-järgulisel lähenemisel
Algoritmiline kauplemine	*Algorithmic trading,* kauplemisviis, kus tehingud tehakse automaatselt vastavalt etteantud reeglitele
API	*Application Programming Interface,* rakendusliides serveriga suhtluseks
ChatGPT	OpenAI poolt välja töötatud keeletehnoloogia mudel
Clean code	Tarkvaraarenduse põhimõte, mis rõhutab koodi loetavust ja arusaadavust
CI/CD	*Continuous Integration/Continuous Deployment,* tarkvaraarenduse protsess, mis automatiseerib tarkvara testimist ja väljalaset
CSS	*Cascading Style Sheets,* kaskaadlaadistik ehk keel, mida kasutatakse veebilehtede kujundamiseks
DBMS	*Database Management System,* andmebaasi haldussüsteem
Digitaalvara	Antud töö kontekstis viitab digitaalvara plokiahelal või sarnasel tehnoloogial põhinevale varale, milleks võivad olla krüptovaluutad või krüptotõendid
DTO	*Data Transfer Object,* andmeedastusobjekt ehk objekt, mis kasutatakse andmete edastamiseks süsteemide vahel
Eesrakendus	*Frontend,* kasutajaliides
Endpoint	Lõpp-punkt ehk URL, mille kaudu saab teha päringuid veebirakendusele
Flask	Väga kerge kaaluga veebirakenduste raamistik (Pythoni jaoks)
GenAI	*Generative artificial intelligence,* tehisintellekti haru, mis kasutab masinõppe tehnikaid, et luua uusi andmeid või sisu, mida pole varem otseselt loodud.

Fundamentaalne analüüs (FA)	*Fundamental analysis,* meetod, mida kasutatakse finantsturgude hindade ennustamiseks ettevõtte majandusnäitajate põhja
HTTP	*Hypertext Transfer Protocol,* hüperteksti edastusprotokoll
HTTPS	*Hypertext Transfer Protocol Secure*, turvaline hüperteksti edastusprotokoll
Issue	Probleem või ülesanne projektihalduskeskkonnas
JDBC	*Java Database Connectivity*, Java andmebaasiühendus mis võimaldab Java rakendustel andmebaasiga suhelda
JPA	*Java Persistence API,* Java püsivusliides ehk Java andmebaasiobjektide salvestamise ja lugemise liides
JSON	*JavaScript Object Notation*, üldlevinud andmevorming, mida kasutatakse andmete vahetamiseks veebis
JWT	*JSON Web Token,* turvaline viis kasutaja autentimisek
Kasutajalugu	*user story,* lühike kirjeldus sellest, mida kasutaja soovib teha
Milestone	Verstapost või tähtpunkt, oluline sündmus või eesmärk projekti käigus
OHLC	*Open, High, Low, Close*, mingi intervalli hinnainfo avamishind, kõrgeim hind, madalaim hind ja sulgemishind
ORM	*Object-Relational Mapping,* objektirelatsiooniline kaardistamine ehk tehnika, mis võimaldab objektorienteeritud programmeerimiskeelte ja relatsioonandmebaaside vahel andmeid konverteerida
Prop	*Property,* kasutatakse Reactis komponentidele andmete edastamisek
Proxy	Klientide ja serveri päringute vahendaja
Python	Interpreteeritav programmeerimiskeel
REST	*Representational State Transfer,* arhitektuuristiil, mida kasutatakse veebiteenuste loomise
RESTful	Omadussõna, tähendab API-t, mis lähtub REST printsiipidest
Sprint	Arendustsükkel, mille jooksul arendatakse välja kindel hulk funktsionaalsusi
SQL	*Structured Query Language,* struktureeritud päringukeel, mida kasutatakse andmebaaside haldamiseks
Tehniline analüüs (TA)	*Technical analysism*, meetod, mida kasutatakse finantsturgude hindade ennustamiseks minevikuhindade ja turu käitumise põhjal

Tagarakendus	*Backend,* serveripoolne loogika mis vastutab andmete töötlemise ning kasutajaliidesele andmete edastamise ees
UI	*User interface,* kasutajaliides
URI	*Uniform Resource Identifier,* tähistab süsteemi- või võrgueressursi ainulaadset identifikaatorit, mis võimaldab seda leida.
URL	*Uniform Resource Locator,* URI alamtüüp, mis täpsustab ressursi asukohta võrgus
UX	*User experience,* kasutajakogemus veebilehe sirvimisel

Sisukord

Jooniste loetelu

1. Sissejuhatus

Viimaste aastate kõrge inflatsioon ja madalad intressimäärad on sundinud inimesi otsima alternatiive traditsioonilistele raha hoiustamise viisidele. Nii on investeerimine ja lühiajaline kauplemine muutunud viimastel aastatel üha populaarsemaks. Finantsturgude kiire areng ja digitaliseerumine on muutnud investeerimise kättesaadavamaks ning sellega koos on arenenud investeerimisplatvormid ja -vahendid [1].

Samas võib investeerimisega tegelemine olla keeruline ja riskantne, eriti kui puuduvad vajalikud teadmised ja kogemused. Investeerimisotsuste tegemine nõuab põhjalikku analüüsi ja teadmisi finantsturgudest. Ühtlasi võib kauplemine nõuda suurt ajakulu ja pidevat tähelepanu turu muutustele reageerimiseks.

Käesoleva töö eesmärk oli luua veebirakendus, mis aitaks kasutajatel teha targemaid investeerimisotsuseid ning vähendaks ajakulu. Rakenduse funktsionaalsuste kavandamisel palusid töö autorid abi kliendilt Darbinvest OÜ, kes seisis silmitsi sarnaste investeerimisteemaliste muredega.

Valminud rakendus pakub kasutajatele turuanalüüsi, investeerimisportfelli haldamise ja erinevate investeerimistööriistade kasutamise võimalusi, olles mugav nii algajale kui ka kogenud kauplejale või investorile.

Käesolev töö on jaotatud mitmeks peatükiks. Teises peatükis antakse ülevaade probleemist ning analüüsitakse olemasolevaid lahendusi. Kolmandas peatükis antakse ülevaade kliendi soovidest, neljandas analüüsitakse tehnoloogiaid ja nende valikut lõputöö raames. Viiendas peatükis tutvustatakse rakenduse arhitektuurilisi põhimõtteid ja omadusi. Kuuendas peatükis analüüsitakse prototüübi tulemusi ning vastavust nõuetele.

2. Probleem

Lõputöö inspiratsioon tuli lüngast olemasolevates lahendustes. Nimelt keskenduvad investeerimisega seotud veebilehed tavaliselt ühele spetsiifilisele valdkonnale - olgu selleks uudised, hinnainfo, portfelli jälgimine, fundamentaalanalüüs või siis vastupidi väga laialivalguvale, üldisele vaatele, kus on võimatu andmete külluses orienteeruda. Olemasolevad lahendused on üldjuhul väga kulukad ja ei kata kõiki funktsionaalsusi. Olemasolevaid platvormide analüüsitakse detailsemalt peatükis 2.2.

2.1 Aktuaalsus

Töö autorid on seisukohal, et investeerimine on aktuaalne ja ajakohane teema. Tihtipeale takistab inimesi investeerimisega alustamast teadmiste ja/või aja puudumine. Investeerimine on aga üks parimaid viise oma raha kasvatamiseks ja tuleviku kindlustamiseks [2]. Seetõttu on oluline, et inimesed saaksid vajalikud teadmised ja oskused, et teha targemaid investeerimisotsuseid.

Antud teema on aktuaalne, kuivõrd investeerimisotsuseid tehakse mitmetes valdkondades ja muutub ajas üha olulisemaks.

2.2 Olemasolevad lahendused

Ka praegu leidub mitmeid rakendusi turgude jälgimiseks ning investeeringute haldamiseks. Järgnevalt on võrreldud populaarsemaid investeerimisplatvorme ja nende funktsionaalsusi. Olemasolevate lahenduste analüüsil kasutati järgmiseid kriteeriumeid: varade hinnad; diagrammid ja graafikud; turu-uudised; kasutajasõbralik kasutajaliides; jälgimisnimekirjad; harivad materjalid; turgude simuleerimine; varade analüüs; AI funktsioonid; ettevõtete profiilid; finantsaruanded; optsioonide analüüs; integratsioon tehnilise analüüsi platvormidega; ostu- ja müügisignaalide jagamine; integratsioon kauplemisplatvormidega; algoritmiline kauplemine; riskijuhtimise tööriistad; portfellihaldus; kogukonna funktsionaalsused; klienditugi; hind. Antud kriteeriumid leiti jooksvalt platvormide analüüsimise käigus.

Lisaks allpool välja toodud platvormidele katsetasid autorid veel teisi olemasolevaid investeerimisvõimaluste funktsionaalsust pakkuvaid rakendusi, kuid kõik lahendused olid sarnased allpool kirjeldatud probleemide ja puuduste poolest, mistõttu neid eraldi välja ei

tooda. Kokkuvõtlik tabel analüüsitud lahendustest on leitav peatükist Lisa 3. Järgnevalt on välja on toodud kõige populaarsemad rakendused.

2.2.1 Yahoo Finance

Yahoo Finance on üks populaarsemaid finantsuudiste ja turuinfo platvorme, mis pakub kasutajatele laia valikut finantsandmeid, uudiseid ja analüüsi. Yahoo Finance pakub kasutajatele võimalust jälgida aktsiahindu, lugeda uudiseid, luua portfelle ning jälgida nende tootlust. Samuti pakub platvorm kasutajatele võimalust saada teavitusi hinnamuutuste kohta [3]

Kuigi Yahoo Finance on populaarne ja laialt kasutatav platvorm, on sellel mõned puuduseid. Üks suurimaid puudusi on platvormi kasutajaliides, mis on aegunud ja mida on ebamugav kasutada. Samuti ei pakuta tehisintellektiga seotud võimalusi. Lisaks võib algajatel raske platvormil orienteeruda andmekülluse tõttu.

2.2.2 MarketWatch

Veel üks populaarne finants-teemaline platvorm on MarketWatch [4]. MarketWatch pakub kasutajatele võimalust analüüsida erinevate aktsiate hinnainfo graafikuid ning ka erinevaid signaale, mis graafikutega seostuvad. Samuti pakub ka MarketWatch ligipääsu aktsiate fundamentaalinfole ning ka piiritletud koguses analüütikute ennustustele.

Platvormil aga ei ole võimalust hallata portfelle ning veebileht on üsnagi aegunud kasutajaliidesega, muutes navigeerimise keeruliseks. Üldiselt on veebileht aga võimekas. Suur rõhk tundub platvormil olevat uudistel, ka esilehel võtavad uudised valdava enamuse lehest. Lehel pakutakse mitmeid tasulisi investeerimisteemalisi tööriistu, mis võivad aidata investoritel oma investeerimisotsuseid paremini informeerida ja ajakohastada. Nende hulka kuuluvad:

- TOP-reitinguga aktsiate nimekirjad: ülevaated kõrgeima hinnatud aktsiate kohta, mis võivad aidata investoritel leida atraktiivseid investeerimisvõimalusi.
- Tegevuslik turuanalüüs: praktilised turuanalüüsi raportid ja soovitused, mis aitavad investoritel mõista turuolukorda ja teha teadlikumaid investeerimisotsuseid.
- Aktsiahinnangud: hinnangud aktsiatele, mis on esitatud lihtsas ja arusaadavas vormingus, aidates investoritel hõlpsalt mõista aktsiate käekäiku.
- Kauplemissoovitused: soovitused kauplemisvõimaluste kohta, mis sisaldavad ostu- ja müügihindu.

- Teavitused: automaatsed teavitused oluliste turusündmuste või muutuste kohta, mis võivad mõjutada kasutaja investeerimisportfelli.
- Lihtsalt kasutatavad aktsiahinnangud: hinnangud aktsiatele, mis on esitatud lihtsas ja arusaadavas vormingus, mis aitab investoritel hõlpsalt mõista aktsiate hinnangut.
- Kauplemiskavad ostu/müügi hindadega: detailne juhendamine kauplemiskavade kohta, mis sisaldavad soovitusi ostu- ja müügihindade kohta.
- Lühiajalised kõikumiskauplemise (*swing trading*) ideed: soovitused lühiajaliste kauplemisvõimaluste kohta, mis võivad aidata investoritel kiiresti kasumit teenida.
- Kohesed hoiatused: automaatsed teavitused oluliste turusündmuste või muutuste kohta, mis võivad mõjutada investeerimisportfelli.
- Automaatne mustrituvastus: vahendid, mis tuvastavad automaatselt graafikutel esinevaid mustreid ja suundumusi, mis võivad olla olulised investeerimisotsuste tegemisel.
- Aktsiate sõelumise tööriistad: tööriistad, mis võimaldavad investoritel filtreerida ja leida aktsiaid vastavalt kindlatele kriteeriumidele ja eelistustele.

2.2.3 TradingView

TradingView on veebipõhine finantsturgude platvorm, mis pakub võimalust analüüsida erinevate varaklasside hinnainfo graafikuid. TradingView pakub laia valikut varaklassidest, eesotsas aktsiate, valuutade, digitaalvarade ja indeksfondidega, võimaldades jagada oma kauplemisideid ka teistega [5]. Platvormile on loodud skriptimiskeel Pine Script, millega saab luua tehnilise analüüsi indikaatoreid, mida graafikutel kuvada ja testida [6]. Samuti on kasutajatel võimalus siduda oma kauplemisplatvorm (Interactive Brokers, TradeStation, Binance jm) TradingView platvormiga ning selle kaudu saata otse ostu- või müügikäske.

Platvorm on populaarne ning pakub üht parimat võimalust tehnilise analüüsi teostamiseks [7]. Kahjuks on platvorm suunatud pigem kogenud kauplejatele või investoritele. Algajatel võib olla raske platvormiga toime tulla, eriti olukorras, kus ei ole kindlat arusaama, kuhu investeerima peaks. Sellest hoolimata mängis TradingView lõputöö raames olulist rolli, olles väline platvorm, millelt sai kokku koguda tehnilise analüüsi indikaatoreid.

Puuduste poole pealt ei ole platvormil võimalik hallata investeeringuid ega portfelle. Varasemates lõputöödes on märgatud puudusi ka väiksemate börside korral: näiteks ei pruugi olla Tallinna börsi aktsiate kõik olulised finantsnäitajad saadaval [8].

2.2.4 MQL5

TradingView'le sarnane tehnilise analüüsi platvorm on MQL5, mis on suunatud ennekõike algoritmilisele kauplemisele. MQL5 kasutajaid saavad jagada oma kauplemisstrateegiaid ja indikaatoreid teistega. Kasutajad saavad jagatud strateegiaid näha ning kopeerida [9].

Ehkki platvormil on suhteliselt suur kogukond, esineb sellel nõrkuseid. Platvormi välimus on aegunud ning navigeerimine raske. Ühtlasi on kauplemisstrateegiad kasutatavad ainult MetaTrader programmi kaudu, mille peab eraldi installeerima. Lisaks on kõige tulemuslikumad strateegiad tihtipeale tasulised ja nende hinnad võivad küündida kuni 1000 USA dollarini kuus [10]. Fundamentaalanalüüsi, portfellihalduse või tehisintellekti võimekused puuduvad.

2.2.5 Tabelarvutustarkvara

Paljud investeerijad eelistavad oma investeeringuid hallata tabelarvutustarkvara abil, nagu Microsoft Excel või Google Sheets. Tabelarvutustarkvara võimaldab kasutajatel luua ja hallata investeerimisportfelle, jälgida aktsiahindu ja luua erinevaid finantsmudeleid. Samuti võimaldab see kasutajatel luua graafikuid ja aruandeid, mis aitavad neil teha paremaid investeerimisotsuseid.

Ehkki tabelarvutustarkvara on kasulik vahend investeeringute haldamiseks, on sellel mitmeid puudusi. Üks suurimaid puudusi on andmete käsitsi sisestamine ja uuendamine, mis võib olla aeganõudev ja tüütu. Lisaks võib käsitsi andmeid sisestades tekkida vigu. Probleemi lahendaks andmete küsimine väliselt rakenduselt, kuid päringute koostamine võib olla keeruline ja nõuda tehnilisi oskusi. Hallatavate või analüüsitavate varade arvu suurenedes võib tabelarvutustarkvara hakata andmepäringute hulka piirama või aeglustub [11] Samuti on piiratud tehnilise analüüsi võimalused ja teavituste saatmine

3. Eesmärk

Olemasolevaid lahendusi analüüsides tuli välja, et ühtset ja kasutajasõbralikku veebirakendust, mis ühendaks kõik lõputöö autorite soovidega arvestatavad funktsionaalsused, ei ole olemas.

Mitmed finantsanalüüsi platvormid nagu Yahoo Finance ja MarketWatch pakuvad pealiskaudset – ja mõnikord ka ajalise viitega – infot finantsturgudega seotu kohta, ja seda kõike üpriski vananenud kasutajaliidese näol. Ligipääs detailsele analüüsile või funktsionaalsustele on hinnabarjääri taga.

Tehnilise analüüsi platvorm TradingView võib olla algajale raske kasutada, olles samas hea tööriist vilunud kauplejale ja investorile. Samas ei paku platvorm tehniliste indikaatorite edasitöötlust, näiteks võimalust kombineerida omavahel mitut signaali, nõudes tihti juurdepääsu ka indikaatori lähtekoodile. Eelnevalt nimetatud rakendused vajavad pidevat uuendamist, et ajaga kaasas käia, kuid uuendustega kaasaskäimine on kulukas.

Tehisintellekti-funktsionaalsusi sisaldavaid platvorme esines veel vähem ning sealgi kasutati AI-d ainult spetsiifiliste ülesannete täitmisel. Need asjaolud tõid esile probleemi, et vajaminevad funktsionaalsused on laiali erinevatel platvormidel, nõudes kasutajapoolset lisatööd veebilehtede külastamisel ja kasutama õppimisel. Lisaks on eelpool nimetatud veebilehed tavaliselt tasulised, mis tähendab, et kasutajad peavad haldama erinevate teenuse eest maksmist, muutes investeerimise ebamugavaks ja kulukas.

Kokkuvõttes kulub uue aktsia või digitaalvara analüüsiks palju aega ja erinevatel veebilehtedel navigeerimist. Pärastpoole tuleb hoida hinnagraafikud lahti ja oodata hetke, mil kasutaja investeerimisstrateegia alusel oleks õige aeg välja valitud varasid osta või müüa. Kui tegu on algajaga, on veel raskem uudsete mõistete ja üleüldise investeerimiskeskkonnaga toime tulla.

Käesoleva lõputöö eesmärgiks oli luua veebirakendus, mis ühendaks endas ülalpool nimetatud funktsionaalsused ühtsesse kasutajasõbraliku liidesega platvormi, sisaldades endas ka unikaalseid funktsionaalsuseid nii tehnilise analüüsi kui tehisintellekti valdkonnast. Töö autorid mõistsid töö mahtu ning keerukust, kuid uskusid, et selline veebirakendus saaks olema kasulik nii algajale kui ka kogenud kauplejale või investorile.

3.1 Prototüübi ulatus

Kõikehõlmava finantsplatvormi loomine on aeganõudev ja kulukas protsess, mis ei mahtunud antud bakalaureusetöö raamidesse. Seetõttu otsustasid töö autorid luua veebirakenduse prototüübi, mis keskendus kolmele peakategooriale, mis valiti koostöös kliendiga. Need kategooriad on:

- **turuanalüüs**, mis hõlmab endas fundamentaal- ja tehnilist analüüsi;
- **investeerimisportfelli haldamine**, mis hõlmab tehingute sisestamist, ajaloolise tootluse arvutamist ning automaatseid soovitusi;
- **investeerimistööriistad**, mille alla kuulub ennekõike tehisintellekti põhine vestlusrobot, mis oskab meie platvormi andmebaasist küsida kasutaja soovitud infot, luua uudiste kokkuvõtted ning anda turgude kiirülevaated.

Selline lahendus annaks kätte üldpildi turgudest, kasutaja portfelli ajaloost ja hetkeseisust ning sellega seonduvatest uudisest või hinnainfost. Edasijõudnutel oleks võimalik seadistada tehnilise analüüsi indikaatoreid, integreerides neid välistest platvormidest nagu TradingView ning sirvida detailseid finantsnäitajaid.

3.2 Klient

Prototüübi ulatuse paika panemiseks pöördusid töö autorid kliendi Darbinvest OÜ poole, kes soovis eelkõige algajatele sobivat kasutaja oskustega kohanduvat veebirakendust. Firma tegeleb investeeringute haldamisega aktsiate, digitaalvarade ning kinnisvara valdkonnas.

Klient soovis saada investeerimisega seotud uudiseid ja teavet mugavalt ühest kohast. Tema visioon hõlmas investeerimisvõimaluste sirvimist ja hindamist, tuginedes finantsnäitajatele, hinnainfole ning uudistele.

Investeerimisvõimaluste all peetakse silmas aktsiaid, digitaalvarasid ning valuutasid – teised varaklassid nagu kinnisvara või võlakirjad jäid antud töö raamidest välja. Pärast varade väljavalimist soovis klient, et talle saadetakse ostu- või müügisignaale ja muid teavitusi, mis aitaksid tal informeeritult investeerimisotsuseid langetada. Ühtlasi soovis klient kasutada tehisintellekti võimekusi, et platvormil navigeerimise asemel oleks võimalik kiirelt küsida infot hoopiski vestlusrobotilt.

Seega kujutas klient endast pikaajalist investorit, kel ei ole iga päev aega turge ega finantsuudiseid jälgida, soovides seevastu kiirkokkuvõtteid ning soovitusi oma portfelli strateegia

parendamiseks.

3.3 Nõuded

Nõuete koostamisel lähtuti nii kliendi soovidest, mis olid seotud pikaajalise investeerimisega kui ka töö autorite ideedest, mis olid seotud lühiajalise kauplemisega.

Koostöös kliendiga sõnastati järgmised platvormi funktsionaalsed nõuded:

1. Turuanalüüs
    - aktsiate fundamentaalinfo näitamine;
    - aktsiate hindamine, et leida potentsiaalseid investeerimisvõimalusi;
    - varade ja ka turgudega üldisemalt seotud uudiste näitamine;
    - varade hinnagraafikute näitamine;
    - graafikutele tehnilise analüüsi indikaatorite lisamine;
    - tehniliste analüüsi indikaatorite loomine ja kasutamine platvormisiseselt;
    - ostu- ja müügisignaalide seadistamine ning nende jagamine teistega.
2. Portfellihaldus
    - kasutajal on võimalik luua portfelle;
    - kasutajal on võimalik portfelli sisestada käsitsi tehinguid;
    - kasutajal on võimalik vaadata portfelli ülevaadet, tootlust ning jaotust;
    - kasutajal on võimalik näha portfelli hinda soovitud kurssiga;
    - platvorm oskab hinnata kasutaja portfelli riskitolerantsi ning selle abil anda soovitusi portfelli strateegia muutmiseks.
3. Investeerimistööriistad
    - platvormil on saadaval tehisintellekti vestlusrobot;
    - kasutajal on võimalik küsida vestlusrobotilt investeerimisega soetud küsimusi;
    - kasutajal on võimalik saada vestlusrobotilt uudiste kokkuvõtteid;
    - kasutajal on võimalik saada vestlusrobotilt turgude kiirülevaateid;
    - platvormil on saadaval harivad materjalid, et oma investeerimisteadmisi arendada.

Mittefunktsionaalsete nõuete alla kuulusid:

- platvorm peab olema kasutajasõbralik ja lihtsasti navigeeritav;
- platvorm peab olema kättesaadav nii arvutis kui ka mobiilis;
- platvorm peab olema turvaline;
- platvorm peab töötama kõikides laialt kasutatavates brauserites, ennekõike Chrome ja Firefox.

4. Tehnoloogiate valik

Nõuete alusel valisid töö autorid tehnoloogiad, mis võimaldaksid luua vastava veebirakenduse. Tagarakenduse loomiseks valiti Java programmeerimiskeel Spring raamistikuga, eesrakenduse loomiseks kasutati JavaScripti koos React raamistikuga ning andmebaasisüsteemi PostgreSQL koos TimescaleDB laiendusega. Projektihalduseks kasutati GitLabi keskkonda. Järgnevad peatükid räägivad valikutest detailsemalt.

4.1 Projektihaldustarkvara

GitLab valiti projektihaldustarkvaraks, kuna see pakub kõiki vajalikke tööriistu, et hoida projekti arendusprotsessi korras. GitLab võimaldab luua projekte, hallata koodibaase, luua *issue*'sid ning jälgida nende täitmist, hallata kasutajaid ja õigusi ning teha koostööd teiste arendajatega. GitLab pakub ka integreeritud CI/CD tööriistu, mis võimaldavad automatiseerida testimist ja rakenduse väljalaset [12].

Platvormi arenduseks loodi kaks eraldiseisvat koodibaasi, üks eesrakenduse ning teine tagarakenduse tarvis. Ees- ja tagarakenduse eraldamine võimaldas autoritel arendada ja eraldiseisvalt hallata rakenduse osi, parandades koodi hooldatavust.

4.2 Andmebaasisüsteemi tehnoloogiad

Üks kõige tähtsamatest tehnoloogilistest valikutest oli andmebaasisüsteem. See pidi olema kiire, turvaline ja võimaldama keerukaid päringuid. Andmebaaside võrdlusel uuriti eelnevaid lõputöid, mis olid seotud käesoleva lõputöö teemaga [13, 14]. Eelnevate lõputööde arhitektuurilised otsused ja tulemused aitasid ka käesoleva töö autoritel oma valikuid paremini teha. Autorid otsustasid, et antud veebirakenduse vajadustega sobib kõige paremini SQL-põhine andmebaasisüsteem.

Valik osutus PostgreSQLi kasuks, võttes arvesse selle võimsust, kasutusvõimalusi ja dokumentatsioonirohkust. Samuti sobib PostgreSQL kasutamiseks eri suuruse ja keerukusega andmebaaside realiseerimiseks [14].

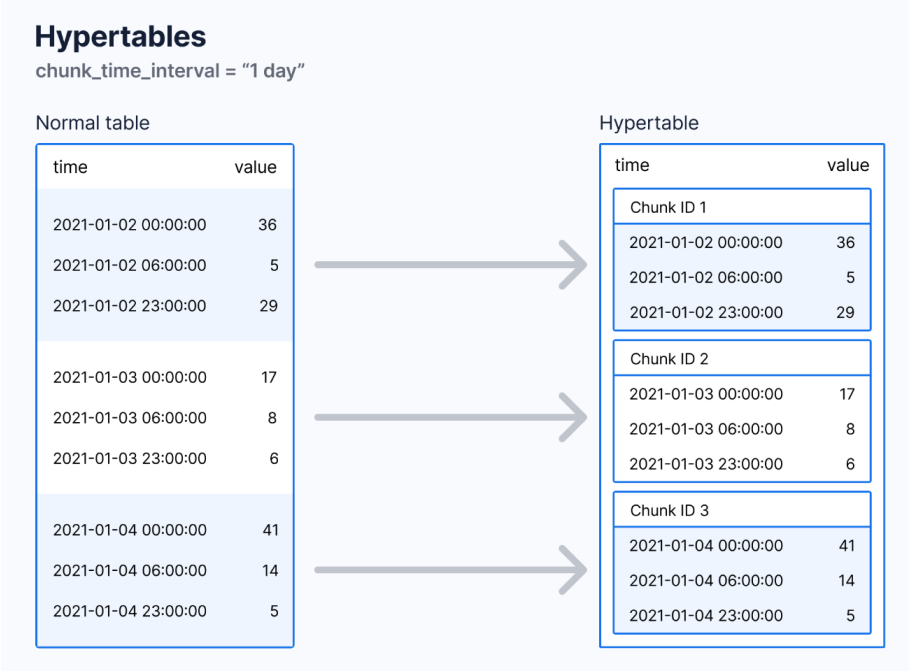
PostgreSQL on avatud lähtekoodiga objekt-relatsiooniline andmebaasisüsteem. PostgreSQL toetab indekseerimist ja tehingute haldamist, järgib ACID põhimõtteid, on laiendatav ja kohandatav ning toetab erinevaid andmetüüpe. See on laialdaselt kasutatav

andmebaasisüsteem, mis on tuntud oma stabiilsuse ja jõudluse poolest. PostgreSQL on skaleeritav, omab aktiivset kogukonda ja tuge, ühendades relatsiooni- ja objektorienteeritud andmebaasihaldussüsteemide parimad omadused [13].

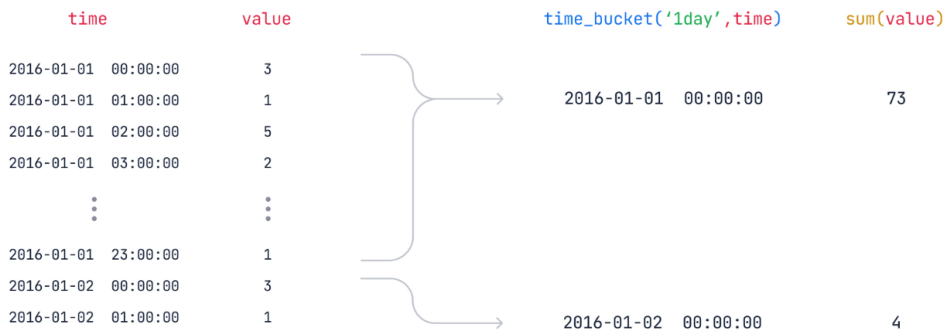
Töö autorid eeldasid, et kõige suurem koormus andmebaasile tekib ajaloolise hinnainfo haldamisel. Seega pidi andmebaas olema võimeline efektiivselt sisestama, hoiustama ning pärima andmeid ka olukorras, kus tabelites on miljoneid ridu.

Seetõttu valiti PostgreSQL DBMS peale laienduseks TimescaleDB, mis muutis aegrealiste andmete haldamise efektiivsemaks, säilitades samal ajal jõudluse [15]. TimescaleDB on loodud PostgreSQLi laiendusena, mis tähendas seda, et olemasolevaid andmebaasistruktuure ei olnud tarvis muuta. TimescaleDB oli mugavalt integreeritav olemasolevate PostgreSQL tabelitega ning toetas SQL päringukeelt.

TimescaleDB on vabavaraline ja spetsiaalselt mõeldud aegrealise info haldamiseks. Seda on kasutatud edukalt mitmete finants- ja IoT-projektide puhul [16]. TimescaleDB kasutab erilisi "hüpertabeleid" (vt Joonis 1), mis taustal jagab tabeli ajatempli alusel väiksemateks tükkideks, võimaldades andmete jätkuvalt kiiret pärimist ka suurte andmemahtude korral [17] (vt Joonis 1). Peale selle on TimescaleDB juurde ehitatud mitmeid lisafunktsioone, mis teevad ajaloolise info haldamise lihtsamaks ja efektiivsemaks. Näiteks on võimalik grupeerida infot ajalise perioodi alusel, tagades niimoodi kokkuvõtliku info mingi kindla ajaperioodi kohta [18] (vt Joonis 2).

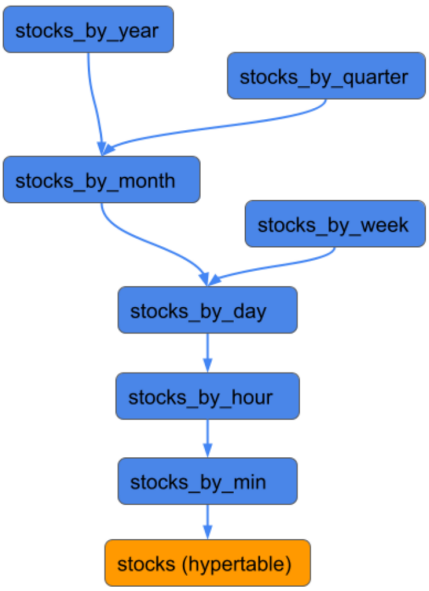


Joonis 1. *Näide hüpertabelist [17]*



Joonis 2. *Näide TimescaleDB time_bucket() funktsioonist [18]*

Ühtlasi on võimalik luua automaatselt uuenevaid SQL-vaateid (*view*) kasutades continous aggregate funktsiooni (vt Joonis 3). See oli eriti kasulik ajaloolise andmestiku kokkuvõtete tegemisel. Näiteks kui oli vaja leida mingi aktsia hinnainfo OHLC (*open, high, low, close*) väärtuseid iga tunni kohta, siis uute andmete sisestamisel uuendas continous aggregate automaatselt vaadet, võimaldades arendajal keskenduda rohkem andmete analüüsimisele, mitte pidavale haldamisele. Mitut sellist vaadet, igaüks pikema intervalliga kui eelmine, oli võimalik üksteise peale ehitada. See tõhustas andmete pärimist veelgi, kuna siis ei pidanud enam otse algse andmestiku poole pöörduma, vaid sai juba eelnevalt materialiseeritud andmeid kasutada [19].



Joonis 3. *Näide mitmest üksteise peale ehitatud continous aggregate vaatest[19]*

4.3 Tagarakenduse tehnoloogiad

Tagarakendus on veebirakenduse osa, mis vastutab andmete töötlemise ja edastamise eest kasutajaliidesele. Tagarakenduse loomiseks valiti Java programmeerimiskeel koos Spring raamistikuga. Spring on üks populaarsemaid Java raamistikke, mis pakub laia valikut tööriistu ja võimalusi veebirakenduste loomiseks [20].

Spring raamistik on modulaarne ja laiendatav, võimaldades arendajatel luua keerukaid ja skaleeritavaid veebirakendusi. See pakub laiapõhist toetust RESTful veebiteenuste loomiseks, andmebaasiühenduse loomiseks, turvalisuse tagamiseks ja ka programmi testimiseks [21].

Tagarakendusega on tihedalt seotud ka andmebaas, mille skeemi muudatuste haldamiseks kasutati *Liquibase'i*, mis tagas andmebaasi skeemi ühtsuse ja kiire muudatuste teostamise [22].

Liquibase võimaldas töö autoritel luua ja hallata andmebaasi skeemi muudatusi läbi XML või SQL failide, tagades selle läbi andmebaasi ühtsuse erinevates masinates ja arenduskeskkondades. Liquibase võimaldas arendajatel näha ka skeemi muudatuse ajalugu, saades parema arusaama andmebaasi arengust.

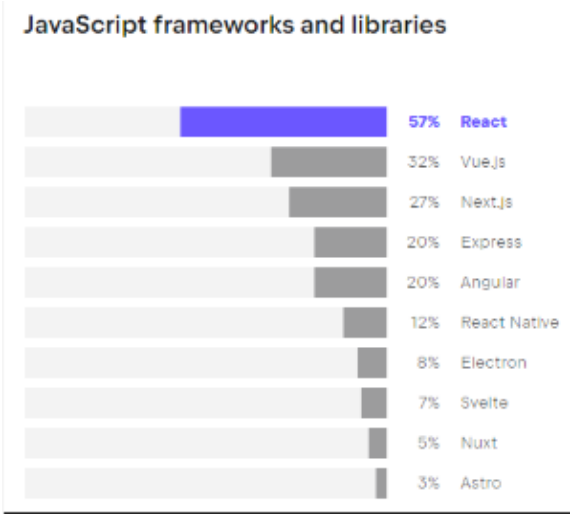
Töö käigus arendatav platvorm pidi kusagilt finantsinfot pärima. Selliseid andmeid pakuvad mitmed firmad, kes on loonud enda API-d. Töö autorid lõid tabelarvutustarkvaras kokkuvõtliku tabeli hindamaks erinevate API-de võimalusi ning - tulevikule vaadates – ka kommertslitsentsi legaalseid õiguseid ja kohustusi (vt ptk 5.6.1).

Finantsinfot otsustati pärida Twelve Data API-st. Antud API pakub laia valikut andmeid, sealhulgas aktsiate fundamentaalnäitajaid ning ajaloolist hinnainfot nii aktsiatele, valuutadele kui ka digitaalvaradele. Peale laiale infovalikule osutus Twelve Data kasuks mõistlik ja arusaadav hind, detailne dokumentatsioon ning õigus infot oma andmebaasi salvestada, kasutajatele kuvada ning ka rakenduses sisemiselt kasutada [23]. Teised API-d olid kas liiga kallid, ei lubanud andmeid sisemiselt kasutada või puudus litsentsis vastav selgitav klausel.

4.4 Eesrakenduse tehnoloogiad

Eesrakendus on veebirakenduse osa, mis vastutab kasutajaliidese eest. Eesrakenduse loomiseks valiti React raamistik. React on jätkuvalt üks populaarsemaid eesrakenduste

raamistikke, pakkudes võimalust luua dünaamilisi ja interaktiivseid kasutajaliideseid [24]. React raamistik on komponentpõhine, võimaldades arendajatel luua korduvkasutatavaid komponente, mis muudavad koodi loetavamaks ja hooldatavamaks [25].



Joonis 4. *React raamistiku populaarsus võrreldes teiste valikutega[24]*

Lehekülgede stiili haldamiseks kasutati CSS märgistuskeelt koos SCSS laiendusega, mis võimaldas kasutada muutujaid, funktsioone ja muid programmeerimiselemente stiilide kirjutamiseks. See muutis koodi lihtsamaks ja arusaadavamaks [26].

Komponentide loomise lihtsustamiseks kasutasid töö autorid vabavaralist MUI (*Material UI*) komponendiraamistikku. MUI raamistik võimaldas kiiresti luua stiililt ühtseid ja kasutajasõbralikke kasutajaliidese elemente. MUI raamistik on modulaarne ja laiendatav, võimaldades arendajatel vastavalt vajadustele luua ka enda kohandatud komponente [27].

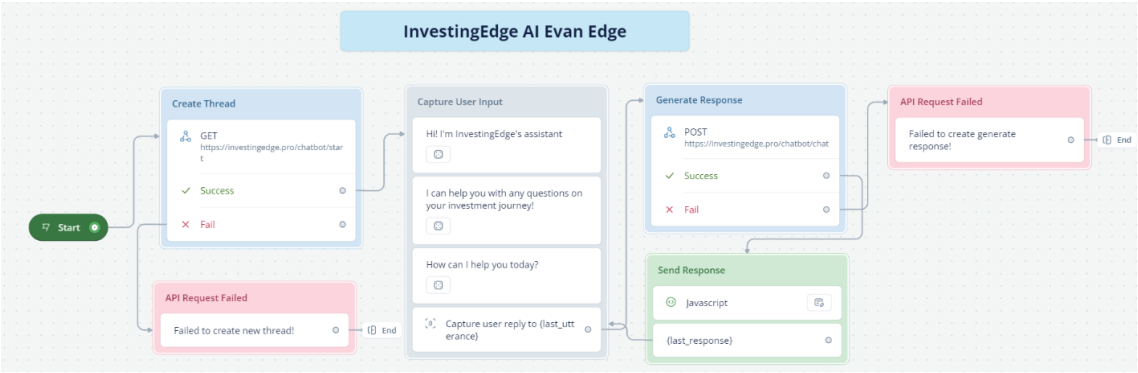
4.5 Vestlusroboti tehnoloogiad

Vestlusroboti loomiseks soovisid töö autorid algselt kasutada OpenAI ChatGPT robotit. Üpriski kiirelt sai aga selgeks, et ainuüksi ChatGPT-l baseeruv vestlusrobot ei suuda toime tulla antud töö nõuetega.

Selle lahendamiseks otsustati luua vestlusrobot, mis kasutab OpenAI ChatGPT-d ainult alumise kihina ning suurem osa vestlusroboti loogikast kirjutati Pythoni programmeerimisekeele abiga, kasutades ühtlasi Voiceflow platvormi võimalusi. Selline tehnoloogiate valik võimaldas autoritel modulaarset lähenemist ning äriloogika eraldatust. Vestlusroboti funktsioonid olid loodud vastavalt vajadustele, mis tulid esile arendusprotsessi käigus ning vestluses kliendiga.

Python on üks tehisintellekti valdkonnas üks populaarsemaid ja võimsamaid programmeerimiskeeli. See on lihtne ja arusaadav, võimaldades kiiret arendusprotsessi. Ühtlasi pakub Python arendamise lihtsustamiseks laia valikut tööriistu ja raamistikke [28].

Voiceflow on veebiplatvorm, mis võimaldab luua ja konfigureerida vestlusroboteid loogiliste omavahel ühendatud plokkide abil [29]. Selle platvormi kaudu saab konfigureerida mitmesuguseid vestlusroboteid, määratledes selgelt nende teatised ja suhtlusvoogude dünaamika (vt Joonis 5). Lisaks sellele võimaldab Voiceflow platvorm tõhusalt siduda erinevaid rakendusi ja programme, et laiendada vestlusrobotite funktsionaalsust ning tagada sujuv kasutajakogemus. Vestlusrobot paigutati eesrakenduse kasutajaliidesesse, võimaldades kasutajatel mugavalt vestlusrobotiga suhelda.



Joonis 5. *Voiceflow platvorm voogskeem tehisintellekti jaoks.*

Vestlusrobot jooksis eraldi programmina ning suhtlus kasutaja ning tagarakendusega käis HTTP päringute abil. Pythoni veebiserverina kasutati algselt kergekaalulist ja lihtsat Flask raamistikku, kuid hiljem liiguti üle Tornado raamistiku peale.

4.6 Veebirakenduse arhitektuur

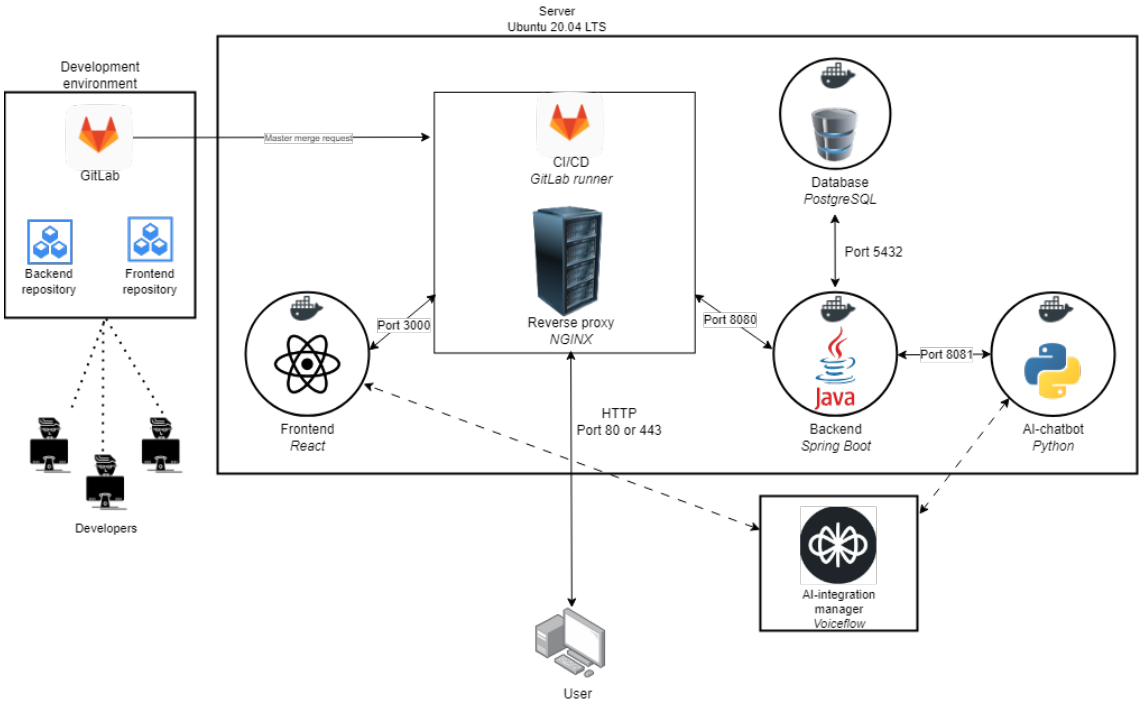
Prototüübi arendamisel kasutasid töö autorid veel mitut tehnoloogiat ja tööriista, mis aitasid luua, hallata ja käivitada antud veebirakendust.

Peale ülal mainitud tehnoloogiate kasutati ka Dockerit. See on tarkvaraplatvorm, mis võimaldab arendajatel luua, paigaldada ja käivitada rakendusi konteinerites. Dockeri konteinerid on kergesti hallatavad ja töötavad erinevates operatsioonisüsteemides, tagades rakenduse parema ühilduvuse. Dockeri konteinerid on kergekaalulised, võimaldades arendajatel mugavalt hallata veebirakenduse erinevaid osi [30]. Antud töö andmebaas, vestlusrobot, taga- ning eesrakendus jooksid kõik eraldi konteinerites, tagades niimoodi parema turvalisuse ja jõudluse.

Kiiruse ja stabiilsuse poolest tuntud NGINX serverit ei kasutatud mitte ainult eesrakenduse serveerimiseks lõppkasutajale, vaid ka erinevate konteinerite vahelise suhtluse koordineerimiseks. NGINX võimaldas hallata suhtlust erinevate konteinerite vahel ja klientide vahel tänu pöördproksile (*reverse proxy*).

NGINX on avatud lähtekoodiga veebiserver. See on tuntud oma kõrge jõudluse, stabiilsuse, rikkalike funktsioonide, lihtsa konfiguratsiooni ja madala ressursitarbimise poolest. NGINX oli loodud selleks, et lahendada C10K probleemi, kus veebiserver peab suutma korraga teenindada üle kümne tuhande ühenduse. NGINXi kasutatav tehnoloogia võimaldab sel saavutada madala mälukasutuse ja kõrge skaleeritavuse [31]. Lisaks hoolitses NGINX turvalise HTTPS ühenduse eest, mis tagas andmete krüpteerituse ja privaatsuse.

Joonis 6-l on kujutatud veebirakenduse arhitektuuriline diagramm, mis ühendab omavahel kõik komponendid.



Joonis 6. *Veebirakenduse komponendidiagramm*

5. Prototüübi arendamine

Olles tehnoloogiad ning arhitektuurilised valikud kinnitanud, oli võimalik igal tiimiliikmel arendustööga aktiivselt alustada. Töö autorid jagasid ülesanded vastavalt rollidele: iga liige keskendus peamiselt ühele kategooriale (turuanalüüs, portfellihaldus või investeerimistööriistad), vastutades oma osa eest. Kõik tiimiliikmed olid aga kursis ka projekti üldise käekäiguga. Kirjutatud kood pidi olema loetav ja arusaadav teistele liikmetele, tagades niimoodi sujuva koostöö ja arendusprotsessi. Sellepärast proovisid autorid järgida *clean code* põhimõtteid, kasutades selgeid ja arusaadavaid muutujaid, funktsioone ja vajadusel kommentaare.

5.1 Arenduspõhimõtted

Prototüüpi arendades kasutati agiilset arendusmetoodikat *Scrum*. *Scrum* on paindlik ja iteratiivne arendusmetoodika, mis võimaldab arendajatel kiiresti reageerida muutustele ja kohandada projekti vastavalt vajadustele. *Scrum*i põhimõtete järgimine aitas tagada, et projekt arenes kiiresti ja tõhusalt [32].

Projekti arendamisel alustati esmalt eesmärkide seadmisest. Suuremate eesmärkide puhul jagati need projektihalduskeskkonnas epic *issue*'deks. Seejärel loodi igale *epic issue*'le kasutajalood ehk *user story*'d, et mõista, kuidas klient funktsionaalsust kasutada sooviks. Kasutajalugude loomise järel loodi väiksemad, üksteisest sõltumatud *task*'id ehk ülesanded, andes neile selged pealkirjad ja nõuded. Ülesannetele lisati ka ajalised hinnangud ning jälgiti nende täitmiseks kulunud aega. Ülesannete täitmiseks kuluv aeg võis varieeruda 30 minutist kuni 72 tunnini.

Iga ülesandega oli seotud eraldi haru ning enne kui see sai integreeritud arendusharusse *dev*, pidi mõni teine tiimiliige läbi viima koodi ülevaatuse ehk *code review*. Ülesandeid lahendati sprintide kaupa, kus iga sprint kestis 14 päeva. Iga sprindi lõppedes korraldati koosolek tiimiliikmete ja juhendaja vahel, kus arutati tekkinud probleemidest ja tehtud tööst. Koosolekul planeeriti ka järgmise sprindi tööd ning püüti kõik valminud harud *dev* harusse integreerida.

Uue veebirakenduse versiooni väljalaskel viidi läbi *dev* haru ühendamine *main* haruga. Selleks, et *main* harus olev programm jõuaks automaatselt serverisse, loodi GitLab CI/CD *pipeline*, mis jälgis muutusi *main* harus. Vastavalt juhistele ehitas see serveris uued Dockeri

pildid ja taaskäivitas rakenduse.

5.2 Andmebaasi arendamine

Andmebaasis hoiustati kõik platvormi tööks vajalikud andmed, mis olid seotud kasutajate, portfellide, tehingute, varade, uudiste, hinnainfo ning ostu- ja müügisignaalidega.

Platvormi tabelid, mis ei seostunud suuremahuliste aegrealiste andmetega ning seega ei nõudnud erilahendust, loodi tavaliste PostgreSQLi tabelitena. Tabelite omavaheliseks sidumiseks kasutati võõrvõtmeid ning andmete terviklikkuse tagamiseks kasutati unikaalsuse kitsendusi. Näiteks kasutajate tabelis olid sätestatud kasutajate andmed, nagu nimi, email, ja räsiga parool, ning võõrvõtmed viitasid portfellide, tehingute, teavituste ja ostu- või müügisignaalide tabelitele. Näited andmebaasi tabelite skeemidest on saadaval Lisa 7.

5.2.1 Hüpertabelid

Aegrealiste andmete haldamiseks kasutati TimescaleDB hüpertabeleid. Valik tavaliste tabelite ja hüpertabelite vahel sõltus andmete oodatavast mahust. Nii olid hüpertabelid kasutusel näiteks ajaloolise hinnainfo, fundamentaalnäitajate statistika ning ostu- ja müügiteavituste hoiustamiseks.

Seoses välisest API-st tuleva infoküllusega ning TimescaleDB sisemise arhitektuuriga tekkis vajadus hüpertabeleid kõigepealt natuke konfigureerida. Nimelt Twelve Data API kõige detailsem hinnainfo intervall oli 1-minutiline, mida oli võimalik küsida kuni 3 aastat tagasi. Pärast seda muutus intervall järk-järguliselt pikemaks: 5 minutit, 15 minutit jne kuni päeva- ja kuuintervallini välja. See tähendas aga seda, et eelneva paari aasta tagune info oli kordades mahukam kui viimase kümne aasta tagune info. Samas andmeid oli vaja kiirelt pärida olenemata hinnainfo mahust. Selleks loodi abistav SQL-keeles koostatud skript, mis programmi esmakäivitusel konfigureeris hüpertabeli selliselt, et eelneva 3 aasta info oli eraldatud 1-päevasteks tabelitükkideks (*chunk*), vanem info hoiustati aga 1-nädalaste tükkidena. Selline lahendus optimeeris mälukasutust: minutilise intervalli andmesiku pärimisel ei pidanud andmebaasisüsteem nii suurt tabelitükki ühekorraga mälus hoidma.

Salvestatava info mahu kasvades kasvas ka andmebaasi maht. Tahtes hoida andmebaasi suurust kontrolli all, kasutati *timescaledb.compress funktsiooni*. See funktsioon võimaldas andmeid kokku pakkida, vähendades mitmekordselt andmebaasi mahtu. Näiteks kui andmebaasis oli 50 erineva vara ajalooline hinnainfo, siis hüpertabeli kokkupakkimise järel vähenes maht pea neli korda: 5.6 GB pealt 1.5 GB peale.

5.2.2 Andmete sisestamine

Ühe välise API päringuga tagastati tagarakendusele keskmiselt 5000 rida, mis salvestati andmebaasi järk-järguliselt batch-päringutega. Selliste päringute haldamiseks kasutati JdbcTemplate klassi, mis võimaldas arendajatel teostada andmebaasi päringuid otse Java koodis.

Lisaks tuli muuta PostgreSQL konfiguratsiooni, mis oli sätestatud *postgres.conf* failis. Näiteks tuli suurendada *max_locks_per_transaction* väärtust. See on PostgreSQL konfiguratsiooniparameeter, mis määrab maksimaalse lukkude (*lock*) arvu, mida üks tehing saab korraga hoida. Kui tehing ületab selle piiri, tekib veateade ning muudatused lükatakse tagasi [33]. Arvestades andmebaasi sisestavate ridade mahtu, tuli probleemide vältimiseks seda parameetrit suurendada.

5.2.3 Koormustestid

Töö käigus teostati TimescaleDB andmebaasile koormustest, et hinnata andmebaasi jõudlust ja vastupidavust. Koormustestide eesmärk oli leida parimad parameetrid hinnainfo salvestamiseks ning seotud vaadete uuendamiseks.

Koormustestid loodi halvimat stsenaariumi silmas pidades: sisestades iga vara kohta 800000 rida 1-minutilise intervalliga andmeid ja seejärel uuendades 1-minutilist *continuous aggregate* vaadet. Testimise käigus mõõdeti protseduurideks kulunud aega.

Kolm peamist järeldust olid:

- Uute andmete sisestamisel oli mõttekas kasutada ajutist tabelit. Sinna salvestati uute varade hinnainfo enne andmete lõplikku ümberliigutamist hüpertabelisse. See protseduur aitas tagada, et kogu hüpertabel pärast korralikult uuesti ka kokku pakiti - andmete otse hüpertabelisse sisestamisel ei pruugitud seda teha.
- Andmete liigutamisel ajutisest tabelist hüpertabelisse mängis rolli hüpertabeli tabelitükkide ajavahemik. Nii oli andmete sisestamise kiirus ühepäevaste tabelitükkide puhul 15% kiirem kui ühenädalaste tabelitükkide puhul.
- *Continuous aggregate* vaadete uuendamine võttis tabeli mahu kasvuga üha rohkem aega. Seega oli vaadete järk-järguline uuendamine ülimalt tähtis, vältides niimoodi ka andmebaasiühenduse katkemist. Mida suuremaks andmete intervallid (5 min, 15 min, 1 tund jne) muutusid, seda suuremaks võis muutuda ka vaate uuendamise intervall.

Töö käigus loodud koormustestide kirjeldused ning graafilises vormis tulemused on saadaval peatükis Lisa 8.

5.3 Tagarakenduse arendamine

Tagarakenduses teostati rakenduse äriloogika, peamiseks ülesandeks oli andmete töötlemine. Rakenduse loomisel lähtuti controller-service-repository mustrist. Ühtlasi kasutati modulariseeritud monoliitset (modularized monolith) ja RESTful API arhitektuuri.

Controller-service-repository mustril põhinev arhitektuur jagab rakenduse kolmeks kihiks [34]:

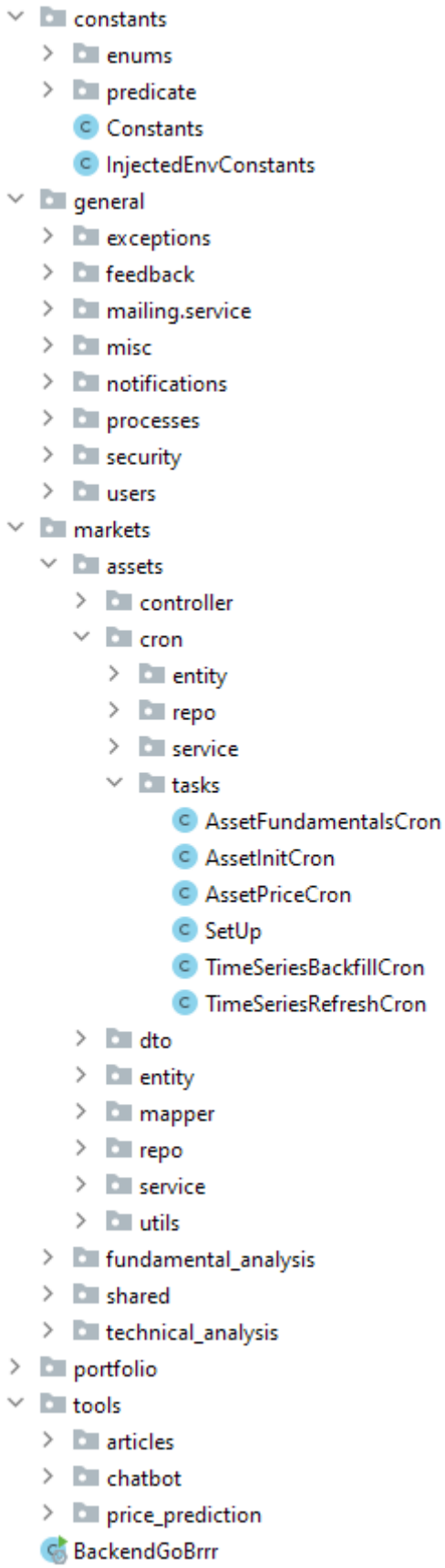
- *Controller* kiht vastutab kasutajaliidese ja tagarakenduse vahelise suhtluse eest. See osa võtab vastu kasutaja päringud ja edastab need *service* kihile.
- *Service* kiht vastutab rakenduse äriloogika eest. Service osa teostab andmete töötlemise ja ärireeglite rakendamise, suheldes vajadusel *repository* kihiga.
- *Repository* kiht teostab andmete pärimist ja salvestamist andmebaasist.

Modulariseeritud monoliit arhitektuuristiil jagab rakenduse mitmeks kaudselt seotud mooduliks, kus iga moodul vastutab konkreetse funktsionaalsuse eest. Moodulid kasutavad ühist andmebaasi. Modulaarne arhitektuur võimaldas arendajatel luua ja hallata rakendust mugavamalt, kuna ühe mooduli äriloogika ei mõjutanud teiste moodulite oma. Selline arhitektuur võimaldas ka rakendust vajadusel kiirelt laiendada [35].

RESTful API arhitektuur võimaldab eesrakendusel suhelda tagarakendusega läbi standardsete HTTP päringute, tagades mugava ja turvalise suhtlusviisi [36].

5.3.1 Struktuur

Tagarakenduse moodulid on nähtavad Joonis 7-l. Igas moodulis on erinevad paketid, mis vastutavad konkreetse funktsionaalsuse eest. Alampakettide nimed on proovitud igas moodulis hoida samana. Standardne moodul koosneb pakettidest: *controller, service, repository, entity, dto, mapper,* ja *utils*.



Joonis 7. *Tagarakenduse pakettide struktuur*

Entity paketi klassid kujutavad endas andmebaasi tabeleid Java objektide kujul (ORM stiil). Selle sätestamiseks kasutati Springi *@Entity* annotatsiooni. DTO paketid vastutavad andmete edastamise eest eesrakendusele. Selline lisakiht aitas kaasa rakenduse turvalisuse hoidmisele. Nii said töö autorid olla kindlad, et salastatud infot, mis võis olla defineeritud tabelis (ning seeläbi *entity* klassis) kasutajaga kogemata ei jagataks. *Mapper* paketid vastutavad andmete teisendamise eest DTO kujule.

Cron paketid vastutavad ajastatud ülesannete eest, näiteks andmete pärimisega välisest API-st. Twelve Data API-ga suhtlemiseks kasutatakse Springi RestTemplate klassi, mis võimaldab teha HTTP päringuid ja saada vastuseid. Ühtlasi on võimalik sätestada kellaaeg, millal salvestatakse andmebaasi hetktõmmis portfellide väärtustest või millal liigutakse ajalooline hinnainfo ühest tabelist teise. Utils paketid vastutavad abifunktsioonide eest.

5.3.2 Turvalisus

Tagarakenduse turvalisuse tagamiseks kasutati Spring Security raamistikku, mis on spetsiaalselt loodud turvalisuse tagamiseks, pakkudes selleks laia valikut tööriistu ja võimalusi. Spring Security võimaldas töö autoritel luua turvalise autentimise ja autoriseerimise süsteemi, mis tagas rakenduse turvalisuse ja kasutajate privaatsuse. Spring Security võimaldas luua kasutajatele erinevaid rolle (näiteks administraator või tavakasutaja) ning määrata neile vastavad õigused ja piirangud.

Konto loomisel kasutati emaili kinnitamise süsteemi, mis tagas kasutaja autentsuse. Konto tegemisel saadeti kasutajale automaatne email, mis sisaldas linki unikaalse kinnituskoodiga. Seda linki tuli klikkida, et konto lõplikult aktiveerida. Email saatmise platvormiks valiti *MailTrap*, mis võimaldas töö autoritel jälgida emaili saatmisel tekkinud logifaile kui ka kasutajate statistikat[37].

5.3.3 Protsesside haldamine

Kuna mitmed protsessid - näiteks uue aktsia ajalooliste aegridade salvestamine - võtsid suures koguses aega, kasutati selliste ülesannete täitmisel *process queue* süsteemi. Selline lahendus võimaldas teostada ajamahukaid ülesandeid taustal oleval eraldi lõimel (*thread*), ilma et ülejäänud programm oleks pidanud ootama ülesande täitmise taga. Process queue lahendus kasutas sisemiselt Springi *TaskExecutor* klassi.

Ülesannete ajastamiseks kasutati Springi *@Scheduled* annotatsiooni, millega sai konfigureerida *cron-stiilis* ülesannete täitmise algusaeg ja intervall. Näiteks oli tänu ajastatud

ülesannetele ning *process queue'le* võimalik aktiveeritud varade ajaloolise aegrealise info salvestamine ajastada kindlale kellaajale, vältides niimoodi rakenduse jõudluse langust.

5.3.4 Kohandatavus

Tagarakenduse arendamisel suunati rõhku ka kohandatavuse peale, et oleks võimalik muuta rakenduse parameetreid lihtsa konfiguratsioonifaili abil. Selline lahendus võimaldas tagarakenduse tööd vastavalt vajadustele kiirelt kohandada, ilma et oleks tarvis olnud muuta sisemist koodi ja rakendust uuesti kompileerida.

Näiteks sätestati rakenduse konfiguratisoonifailis ajastatud ülesannete intervallid, lõimede maksimaalne arv, konfidentsiaalsed API võtmed või varade andmebaasist välja filtreerimise piirangud vahetusplatvormi või valuuta järgi. Pärast konfiguratsioonifaili muutmist tuli rakendus lihtsalt taaskäivitada.

5.4 Eesrakenduse arendamine

Eesrakendus vastutas kasutajaliidese eest, olles kasutajatele otsene kokkupuude antud platvormiga. Eesrakenduse loomisel lähtuti komponentpõhisest arhitektuurist, milleks kasutati Reacti raamistikku.

Eesrakenduse arendamisel olid esikohal kasutajaliidese mugavus ning navigeeritavus. Head stiili aitasid hoida kasutatavuse mustrite (Behavioral Patterns) jälgimine. Nende hulka kuulusid järgnevad punktid [38]:

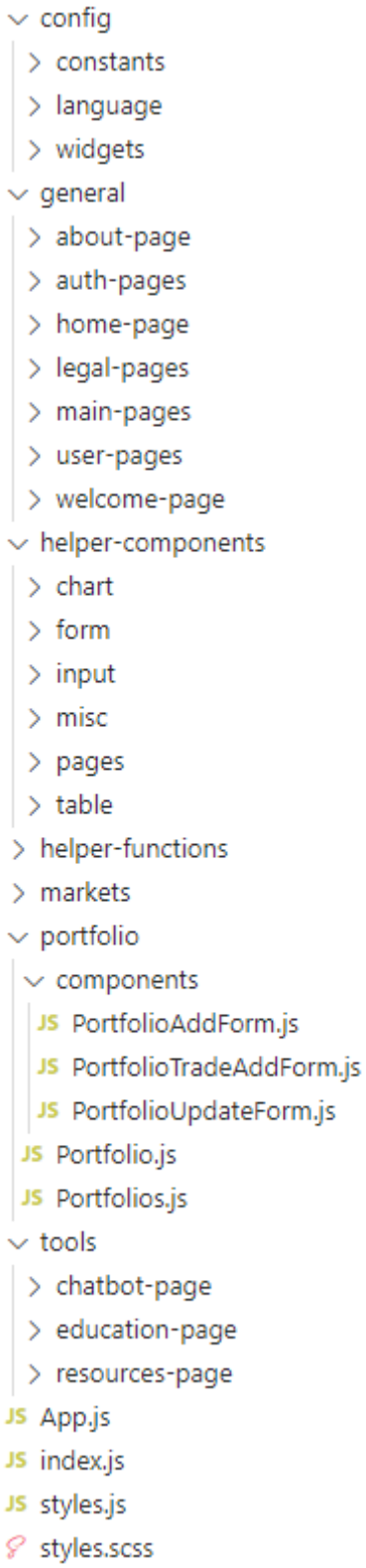
- Turvaline uurimine: kasutaja peaks saama platvormil tegevusi teha, ilma et ta kogemata midagi ära lõhuks.
- Kohene rahulolu: kasutajal peaks olema võimalus ilma täiendavate klikkideta teda huvitava teema juurde jõuda.
- Piisav rahulolu: kasutaja peaks saama kogu vajaliku info kätte ilma, et ta peaks liigselt vaeva nägema ning suuri tekstiplokke lugema.
- Jooksvad muutused: kasutaja peaks saama oma hetketegevust kergelt muuta.
- Ruumiline mälu: kasutajad tihti mäletavad nuppude ja vormielementide asukohta, mitte nende nimetusi. Igal lehel peaks olema standardne paigutus, et kasutajad saaksid kiirelt navigeerida.
- Sujuv kordus: kasutajad peaksid saama kergelt korduvaid tegevusi teha. Sellistele tegevustele peaks leiduma otsetee.

5.4.1 Struktuur

Eesrakenduse struktuuri prooviti hoida tagarakendusega sarnasena (vt Joonis 8). Eesrakenduse kaustad jagati kaheks suuremaks osaks: leheküljed ja alamkomponendid. Leheküljed vastutavad kasutajaliidese kuvamise eest, kasutades väiksemaid alamkomponente. Komponendid on korduvkasutatavad osad, mis vastutavad konkreetse funktsionaalsuse eest. Komponendid kasutavad lehekülgede poolt edastatud andmeid (*props*) info kuvamiseks.

Lisafunktsioonid olid eraldatud kausta *helper-functions*, mis aitasid pärida, vastu võtta ja töödelda andmeid. Näiteks oli võimalik luua HTTP päringuid aktsiatele, mis vastavad kasutaja otsingule või teisendada saabunud info tabeli või graafiku kujule.

Universaalsed komponendid, mida kasutati mitmel lehel, olid kasutas *helper-components*. Näiteks oli võimalik ühe *StyledTable* komponendiga kuvada erinevaid tabeleid.



Joonis 8. *Eesrakenduse kaustade struktuur*

Peale isetehtud komponentide kasutati ka väliseid vidinaid (widget), mis võeti TradingView platvormilt. Selliste valmistehtud vidinate abil oli võimalik kuvada varade graafikuid ja uudiseid, mis sobisid antud prototüübi arendamiseks suurepäraselt[39]. Selliseid vidinaid oli võimalik konfiguratisooniparameetrite abil ka ümber kujundada, et sobiksid antud platvormiga paremini kokku. Näiteid eesrakenduse kujundusest on võimalik näha peatükis 5.4.4.

5.4.2 Turvalisus

Kasutaja sisselogimisel kasutati JWT (JSON Web Token) autentimisstandardit. JWT on turvaline ja lihtne autentimisskeem, mis võimaldab kasutajal ennast krüpteeritud tokeni alusel autentida [40]

JWT tokenid genereeriti ning krüpteeriti tagarakenduses kasutaja andmete põhjal ning saadeti kasutajale vastusena autentimispäringule. Tokenid salvestati kasutaja brauseri lokaalsesse hoidlasse (*localstorage*). Väljalogimisel kustutati brauseri mälust JWT ning ka muud kliendiandmed.

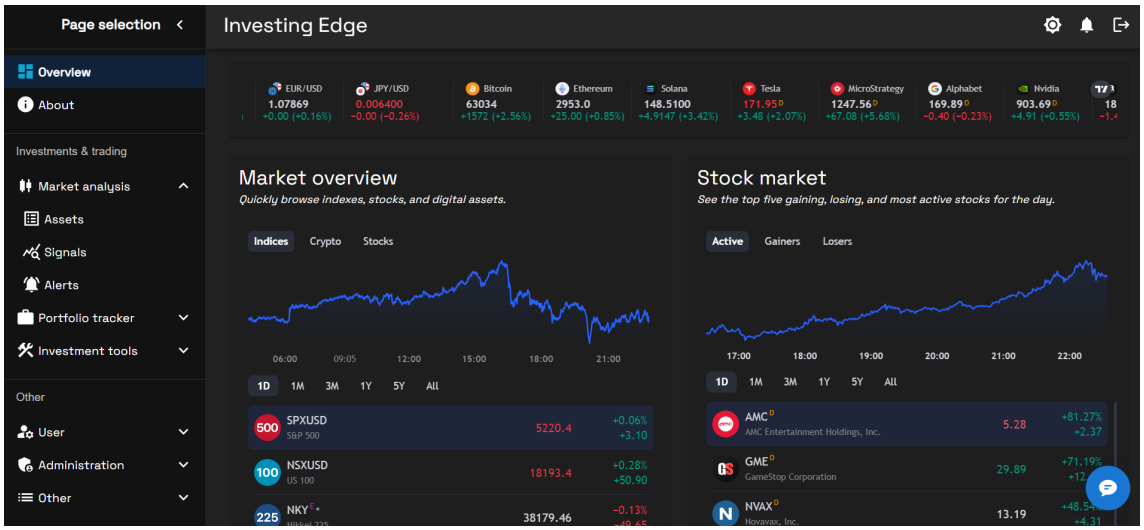
5.4.3 Kasutatavuse mustrite rakendamine

Järgnevalt on kirjeldatud mõned tähtsamad kohad, kus esines kasutatavuse mustrite rakendamist.

Hoides silmas turvalise uurimise mustrit, loodi platvormi võimalusi tutvustav leht, ilma, et oleks vaja luua kasutajakontot. See leht tuleb kasutajale automaatselt ette, kui ta esmakordselt platvormi külastab. Ühtlasi lisati kõikidele vormielementidele juurde valideerimine ning potentsiaalsed veateated.

Kohese rahulolu mustri põhjal oli kasutajal võimalik luua konto vaid mõne klikiga. Konto loomiseks oli vaja kasutajanime, e-posti aadressi ja parooli. Kasutaja autentsuses veendumiseks oli vajalik ka e-posti kinnitamine.

Järgides piisava rahulolu mustrit, näidati kasutajale esilehel turgude üldpilti, andes sügavamale navigeerimiseta kätte kiire ülevaate turuolukorrast (vt Joonis 9).



Joonis 9. *Platvormi kodulehe vaade vasakpoolse menüüribaga*

Jooksvate muutuste mustrit järgides oli kasutajal võimalik kogu aeg näha navigeerimisriba, mis sisaldas erinevaid selgitavaid ikoone koos linkidega vastavatele lehtedele (vt Pilt 9). Lisaks ilmus hiirega menüüriba peale liikudes alamlehe lühikirjeldus.

Ruumilise mälu mustrit järgides oli lehtede komponentidel standardne paigutus. Ühtlasi olid elementidel nagu tabelid nupud samasuguses asukohas. Investeerimisvõimalused on esitatud kolmes eraldi kategoorias, vastavalt funktsionaalsete nõuete loetelule: turuanalüüs (market analysis), portfell (portfolio tracker) ning investeerimistööriistad (investment tools) (vt Joonis 9).

Sujuva korduse mustri põhjal muudeti tabeleid kasutajasõbralikumaks. Näiteks oli võimalik valida korraga mitu rida, et neid kustutada või uuendada.

Lõputöö lõpus viidi kliendiga läbi kasutajaliidese kasutatavuse ning mugavuse analüüs, millest räägitakse lähemalt peatükis 6.1.

5.5 Vestlusroboti arendamine

Generatiivsel tehisintellektil (GenAI) põhineva vestlusroboti arendamisel keskenduti kõigepealt erinevatele lähenemisviisidele, kuidas sellist vestlusrobotit oleks võimalik luua.

Esimene võimalus oleks olnud alustada nullist ning iseseisvalt tehisintellekti keelemudel luua ja treenida. Teine võimalus oleks olnud kasutada juba valmis mudelit ning seda antud töö vajaduste põhiselt kohandada.

Kõige tõhusamaks lähenemisviisiks osutus kasutada juba valmis keelemudelit. Prototüübi raames ei olnud autoritel piisavalt aega ega ressursse, et ise GenAI-d nullist treenida ja seeläbi ka aksepteeritav kvaliteet saavutada.

5.5.1 Struktuur

Töös valiti kasutamiseks OpenAI Assistant API, kuna see võimaldas GenAI käitumist kohandada vastavalt platvormi nõuetele. Töös välistati teised variandid, kuivõrd võimaldatavad funktsionaalsused olid neil puudulikud. Näiteks ei olnud teistel lahendustel võimalust lisada erifunktsioone nagu andmebaasiga suhtlemine.

Kuna töötati GenAI-ga, pidi vestlusrobotil olema võimekus vastuste formuleerimisel kas otsida ja lisada andmeid ise või siis tugineda olemasolevale teadmiste põhjale (*knowledge base*). Käesolevas töös puudus mõistlik dünaamiliste andmete lisamise võimalus, peamiselt seetõttu, et kui andmeid taheti muuta, tuli vestlus-sessioon taaskäivitada, mis omakorda kaotas kasutaja vestluse ajaloo. Selletõttu valisid töö autorid teise lahenduse, kasutades teadmiste põhja.

Eesrakenduse liideseks valiti Voiceflow platvorm, millega oli võimalik mugavalt integreerida vestlusrobot kasutajaliidesega. Voiceflow võimaldas autoritel kohandada vestlusroboti välimust ja käitumist. Suhtlus kasutaja, Voiceflow platvormi ning Pythoni programmi vahel toimus HTTP päringute abil.

5.5.2 Suhtlusloogika

Pythoni koodi loogika koosnes erinevatest funktsioonidest, mis vastutasid erinevate kasutajaküsimuste ja -käskude töötlemise eest. Kui vestlusrobot tuvastas mõned funktsiooniga seotud võtmesõnad, käitus ta vastavalt funktsiooni lähtekoodis sätestatule: kontrollis parameetreid ja vajadusel küsis kasutajalt puuduva info täpsustamist. Näiteks kui vestlusrobot tuvastas kasutaja soovi saada viimase hinnainfo mingi aktsia kohta, kontrollis ta kõigepealt, kas kõik vajalikud parameetrid on juba sellel sessioonil olemas. Kui mõni infokild puudus (näiteks aktsia nimi), küsis ta kasutajalt selle täpsustamist, enne kui jätkas funktsiooni täitmist.

Algselt valisid autorid Pythoni veebiserveriks Flaski raamistiku, kuna see võimaldas kiiresti luua ja hallata HTTP-päringuid GenAI-ga suhtlemiseks. Pärast prototüübi piisavat testimist otsustasid autorid üle minna Tornado serverile. Tornado pakkus paremat jõudlust ja skaleeritavust, mis oli oluline reaalajas vestlusroboti toimimiseks. Samuti tagas Tornado

suurema vastupidavuse suurema kasutajate arvu korral, võimaldades platvormil sujuvalt töötada isegi suure koormuse korral. Seega võimaldas Tornado kasutuselevõtt autoritel täiustada vestlusroboti töökindlust ja reageerimiskiirust, samal ajal tagades platvormi jõudluse.

5.6 Bränd

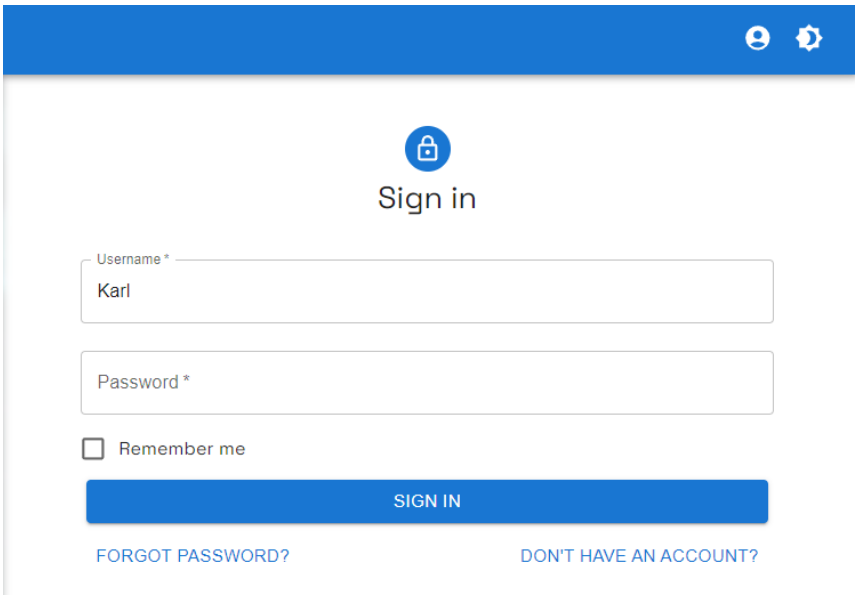
Koostöös kliendiga sooviti luua platvormile ühtne bränd, mida saaks tulevikus laiemale avalikkusele esitledes ära kasutada.

Domeeninimi on oluline osa brändi identiteedist. See on esimene asi, mida kasutajad peavad teadma, kui nad välja töötatud platvormi soovivad brauseris avada. Seetõttu pidid töö autorid valima nime, mis oleks meeldejääv, ainulaadne ja kajastaks platvormi väärtusi. Selle määramiseks loodi tabelarvutustarkvaras nimekiri potentsiaalsetest nimedest ning demokraatlikul hääletusel valiti selleks investingedge.pro. Nimekiri ja hääletuse tulemused on saadaval peatükis Lisa 5.

Investeerimismaailmas viitab mõiste "serv" (edge) unikaalsele eelisele, mida investor turul teiste ees omab. See võib olla eripärane vaatenurk, spetsiifiline strateegia või mingi oskuste komplekt, mis võimaldab investoril teha informeeritumaid otsuseid ja saavutada seeläbi paremaid tulemusi[41].

Samuti arutati platvormi värviskeemi üle. Otsustati luua kaks värvikombinatsiooni, üks valge taustaga (RGB 255, 255, 255) ning teine tumeda taustaga (RGB 18, 18, 18). Platvormi sekundaarvärviks valiti sinine (RGB 25, 118, 210), sümboliseerimaks lojaalsust ning loogilisust (vt Joonis 10)[42].

Värviteema valitakse esmakordsel platvormile sisenemisel automaatselt vastavalt seadme sätetele, kuid kasutajal on võimalik ka käsitsi vastavalt soovile valida heleda või tumeda värviskeemi vahel.



Joonis 10. *Platvormi hele värviteema näide*

5.6.1 API-de valik ja juriidiline pool

Platvormi arendamisel oli API-de valikul oluline tagada nende sobivus mitte ainult tehnilises mõttes, vaid ka juriidiliste nõuete ja piirangute osas. Iga API teenusepakkuja määrab kindlaks oma tingimused, millele tuleb vastata, et tagada juriidiline vastavus ja vältida võimalikke õiguslikke probleeme (vt Lisa 4).

Platvormi API-de valimisel juhindusid autorid järgmistest kriteeriumitest:

- Tingimuste läbivaatamine: Iga API kasutustingimused ja litsentsilepingud loeti hoolikalt läbi. See hõlmas tingimuste mõistmist, eriti seoses andmete kasutamise, säilitamise ja kuvamisega.
- Vastavuse tagamine: Veenduti, et kõik kasutatavad API-d oleksid vastavuses nende pakkujate määratud tingimustega. Näiteks TwelveData API võimaldas andmete kuvamist ja kasutamist ärilisel eesmärgil vastavalt Pro+ taseme plaanile, mis tagati nõuetekohaste litsentside hankimisega.

Nende sammude järgimine tagas, et platvormi arendamine toimus vastavalt kõikidele juriidilistele nõuetele, pakkudes turvalist ja usaldusväärset keskkonda tulevastele kasutajatele ja investoritele.

5.6.2 Platvormi kasutustingimuste ja privaatsuspoliitika loomine

Töö autorid pidasid platvormi arendamisel oluliseks ka kasutustingimuste ja privaatsuspoliitika loomist. Need dokumendid tagasid, et platvormi kasutamine oli reguleeritud ja kasutajate andmed olid kaitstud vastavalt kehtivatele seadustele ja parimatele tavadele.

Alustuseks uurisid autorid mitmete sarnaste veebiplatvormide kasutustingimusi ja privaatsuspoliitikaid. See võimaldas mõista, milliseid punkte on oluline kaasata ja kuidas teised platvormid on lahendanud erinevad õiguslikud küsimused. Analüüsi käigus pöörati tähelepanu järgmistele aspektidele:

- Kasutustingimuste struktuur ja sisu: Vaadeldi, kuidas teised platvormid olid oma kasutustingimused struktureerinud ja milliseid olulisi punkte need sisaldasid. Erilist tähelepanu pöörati kasutajate õiguste ja kohustuste määratlemisele, vastutuse piirangutele ja intellektuaalomandi kaitsele.
- Privaatsuspoliitika üksikasjad: Uuriti, kuidas teised platvormid käsitlesid andmete kogumist, kasutamist, säilitamist ja jagamist. Eriti olulised olid meetmed, mida rakendati kasutajate privaatsuse kaitsmiseks ja andmete turvalisuse tagamiseks.

Teiste veebiplatvormide dokumentide põhjal loodi oma kasutustingimused ja privaatsuspoliitika, järgides järgmisi samme. Tuginedes teiste platvormide parimatele tavadele, koostati esialgsed versioonid nii kasutustingimustest kui ka privaatsuspoliitikast. Põhitähelepanu oli tagada, et dokumendid oleksid selged, arusaadavad ja hõlmaksid kõiki vajalikke punkte. Seejärel kohandati ja täiendati esialgseid dokumente vastavalt platvormi spetsiifilistele vajadustele ja omadustele. Näiteks määratleti täpselt, kuidas platvormil kogutud andmeid kasutatakse ja millised on kasutajate õigused ning kohustused

Lõplikuks sammuks oli dokumentide juriidiline läbivaatus, et tagada nende vastavus kehtivatele seadustele ja regulatsioonidele. Selleks tehti koostööd õigusteaduse taustaga spetsialistiga, kes vaatas dokumendid põhjalikult läbi ja kinnitas nende õigsuse.

6. Tulemused, analüüs ja edasised arendusvõimalused

Käesoleva lõputöö tulemusena loodi toimiv veebipõhine finantsanalüüsi prototüüp, mis sisaldab kasutajasõbralikku eesrakendust ja modulaarset tagarakendust koos laiendatava andmebaasiga[43].

Veebirakendus võimaldab sisse logitud kasutajatel suhelda vestlusrobotiga, küsides sellelt investeerimisalaseid nõuandeid ja infot erinevate varade kohta. Samuti pakub rakendus võimalust jälgida varade ajaloolist hinnainfot ja tutvuda aktsiate fundamentaalnäitajatega.

Lisaks saavad kasutajad rakenduses teostada varadele tehnilist analüüsi, konfigureerides ostu- või müügisignaale ja nendega seotud teavitusi. Kasutajatel on ühtlasi võimalik ka teistelt kasutajatelt signaale tellida. Selliseid ostu- või müügisignaale on võimalik omavahel kombineerida ning seeläbi luua juba uusi investeerimisstrateegiaid, suurendades niimoodi tulusust ja eemaldades vajaduse pidevalt hinnagraafikuid jälgida.

Rakendus võimaldab samuti kasutajatel luua ja hallata investeerimisportfelle. Portfelli hinnainfo lisamisel salvestatakse vara hind nii vara enda kursi kui ka USA dollari väärtuse järgi. Pärast hinnainfo lisamist salvestatakse portfelli info, võimaldades kasutajatel jälgida portfelli väärtust mis tahes kursis. Portfelli hetkeväärtust salvestatakse perioodiliselt andmebaasi, võimaldades kasutajatel oma portfelli arengut jälgida. See funktsioon on kasulik ka neile, kes soovivad saada praktilist kogemust finantsturgudelt, kuid ei julge veel oma raha investeerida, võimaldades neil simuleerida aktsiate ostmist ja müüki.

Järgnevalt kirjeldatakse prototüübi tööpõhimõtet ja kasumlikkust, ühtlasi vaadatakse üle töö vastavus kirja pandud funktsionaalsetele ning mittefunktsionaalsetele nõutele. Lisaks analüüsitakse kliendilt saadud tagasisidet ja rakenduse tehnoloogilisi ja arhitektuurilisi otsuseid. Lõpetuseks pakuvad töö autorid välja mõtteid kuidas võiks rakendust edasi arendada.

6.1 Prototüübi tööpõhimõte

Platvormi prototüüp koosneb neljast osast: eesrakendusest, tagarakendusest, andmebaasist ja vestlusrobotist, mis on integreeritud tagarakendusega.

Rakenduse üldine tööpõhimõte on järgmine:

1. Tagarakenduse esmakordsel käivitamisel luuakse *Liquibase'i* abil andmebaasi tabelite struktuur ning sisestatakse tähtsad üldkasutatavad väärtused, näiteks kasutajarollid.
2. Tagarakendus initsialiseerib Twelve Data API kaudu kauplemisplatvormide, valuutade ning saadaval olevate varade nimekirjad, salvestades need andmebaasis vastavatesse tabelitesse.
3. Kasutaja sirvib ja filtreerib eesrakenduse kaudu erinevaid saadaolevaid varasid ning saadab tagarakendusele vara aktiveerimise päringu.
4. Tagarakendus teostab *process queue* abil päringud Twelve Data API-le, salvestades andmebaasi aktiveeritud vara ajaloolise hinnainfo ning ka fundamentaalnäitajad.
5. Pärast edukat suuremahulise aegrealise info salvestamist uuendatakse *continuous aggregate* vaateid, võimaldades kiiret ja efektiivset andmete pärimist.
6. Vastavalt konfiguratsioonifailis sätestatud väärtustele hakkab tagarakendus välisest API-st perioodiliselt aktiveeritud varade hiljutist hinnainfot ning aktsiate puhul ka muud statistikat küsima.
7. Pärast edukat vara aktiveerimist on võimalik kasutajal sirvida vara detailseid näitajaid, lugeda uudiseid ning lisada varaga seotud tehinguid oma portfelli.
8. Portfelli vaates kuvatakse kasutajale tema investeeringute hetkeseis, tema soovitud kursis, võimaldades lisaks kasutajal jälgida oma investeeringute ajaloolist kasvu või kahanemist.
9. Kasutajad, kes on tuttavad TradingView platvormiga, saavad oma tehnilise analüüsi indikaatorite signaale siduda antud platvormiga.
   (a) Näiteks, olles sidunud platvormiga 2 indikaatorit Apple'i aktsia kohta, üks trendi järgiv ja teine volatiilsust järgiv indikaator, saab kasutaja luua platvormi kaudu uue kombineeritud signaali, mis teavitab kasutajat siis, kui mõlemad signaalid on näiteks ühe tunni jooksul aktiveerunud.
   (b) Selline süsteem võimaldab kasutajatel mõelda välja uusi strateegiaid, ühtlasi vähendades pidavat graafikute jälgimise vajadust.
   (c) Juhul kui kasutaja soovib oma strateegiaid teistega jagada, saab ta need avalikuks või salasõnaga kaitstult poolavalikuks muuta.
10. Teised kasutajad saavad sirvida, tellida ning kombineerida teiste poolt jagatud ostu- või müügisignaale, luues selleläbi tugeva investeerimiskogukonna.
11. Vestlusrobot on kättesaadav kõigile sisselogitud kasutajatele, kelle käest saab küsida investeerimisalast nõu ja andmeid või saada platvormi tehnilist tuge.

6.2 Prototüübi kasumlikkus

Kasumlikkuse all mõtlevad töö autorid prototüübi võimekusi, mis aitavad kasutajatel teha targemaid investeerimisotsuseid. Kasumlikkus on oluline, kuna see mõjutab otseselt

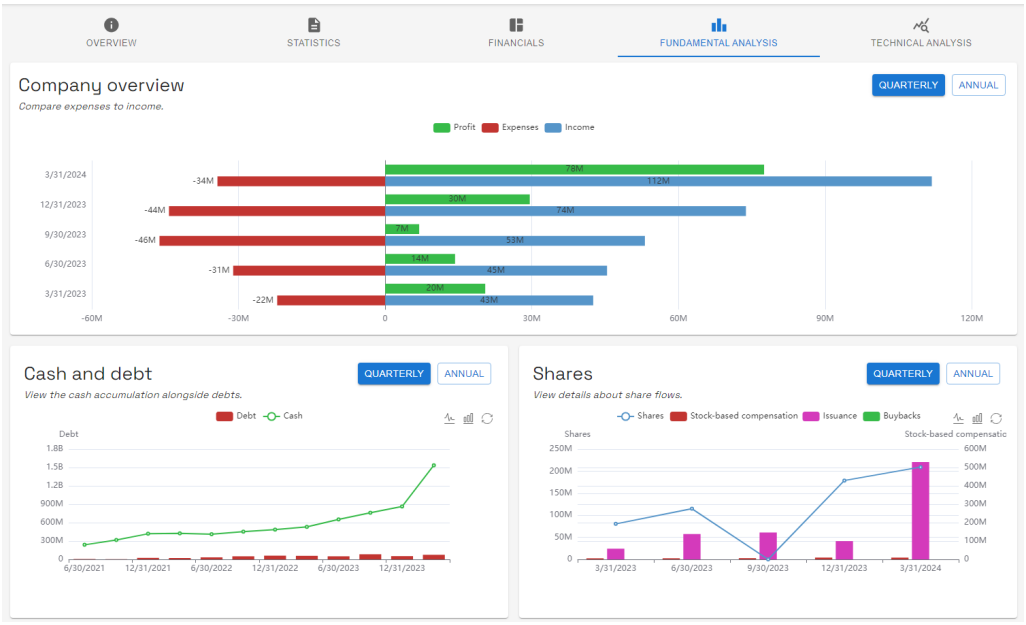
kasutajate rahulolu ja platvormi kasutamise sagedust. Kui kasutajad leiavad, et platvorm aitab neil teha paremaid otsuseid, siis on nad tõenäolisemalt valmis platvormi soovitama ka teistele.

6.2.1 Fundamentaalanalüüs

Valminud prototüüp võimaldab kasutajatel tutvuda erinevate varade fundamentaalnäitajatega, mis aitavad neil teha teadlikumaid investeerimisotsuseid. Fundamentaalnäitajad on olulised, kuna need annavad kasutajatele ülevaate varade tervisest ja tulevikupotentsiaalist.

Näiteks saavad kasutajad tutvuda varade finantsnäitajatega nagu kasumimarginaal ja võlakohustused. Andmete visualiseerimiseks ja kiire ülevaate saamiseks on vara detailvaatelehel välja toodud paar graafikut, mis kajastavad ajaloolisi finantsnäitajaid (vt Joonis 11).

Antud näites on näha bitcoini-kaevandaja *CleanSpark*'i finantsnäitajaid. Firma on olnud pidevalt kasumis, võlakoormus on madal, vaba raha on suures koguses ning firma on vastutustundlikult käitunud, piirates aktsiapõhise hüvitise (*stock-based compenstation*) väljamaksmist. Ainus murettekitav näitaja on aktsiate emissioon, mis on hiljuti tõusnud. Arvestades aga antud valdkonna tihedat konkurentsi ning sellega seonduvat vajadust kaevandusmasinaid pidevalt uuendada, on see kõrge näitaja arusaadav – arvestades ka firma hiljutist kasumlikkust, tundub aktsiate emissioon olnud juhatusel õige valik. Seda tõestab ka firma aktsiahinna graafik, mis on alates 2024. aasta algusest üle 60% tõusnud (24. mai 2024 seisuga).



Joonis 11. *CleanSpark (CLSK) finantsnäitajad graafilises vormis*

6.2.2 Tehniline analüüs

Peale fundamentaalanalüüsi on olulisel kohal ka tehniline analüüs, mis keskendub varade hinnaliikumisele ning püüab ennustada, kuhu hind tulevikus liikuda võib. Hinnakõikumise alusel on võimalik luua erinevaid indikaatoreid, mis omakorda võimaldavad luua kauplemisstrateegiaid ostu- ja müügisignaalide näol.

Valminud prototüüp pakub võimalust selliseid signaale kokku koguda ning neid omavahel suurema tulemuslikkuse raames kombineerida, vähendades ühtlasi pidevat hinnagraafikute jälgimise vajadust. Allpool on illustreeriv näide, kuidas kombineeritud signaalid võivad aidata kasutajal teha paremaid investeerimisotsuseid.

Joonisel 12 on kajastatud peale Tesla hinnagraafikule kaks tehnilise analüüsi indikaatorit: esimene on näidatud kollase-sinise taustaga ning teine *„buy/sell“*(ostu/müügi) kolmnurgakestega. Kasutajal on võimalik prototüübi kaudu seadistada uus kombineeritud signaal, mis saadab kasutajale teavituse siis, kui mõlemad indikaatorid on aktiveerunud näiteks viimase tunni jooksul. Antud näites tähendaks see nii tõusvat kollase taustaga trendi-indikaatorit kui ka *„buy“*(ostu) kolmnurga olemasolu. Selline olukord vähendab „valehäirete“ tekkimist (näiteks esimene *„buy“*(ostu) signaal, millele ei järgnenud tõusvat trendi ning seega annab kasutajale suurema kindlustunde ostu- või müügihetkel.

Joonis 12. *Tesla (TSLA) hinnagraafik koos kahe tehnilise indikaatoriga.*

Missuguseid signaale kombineeritakse, on kasutaja enda otsus. Kui aga kasutaja on leidnud tulusa strateegia, saab ta seda kergelt teistega jagada. Antud töö raamesse ei mahtunud tagasiulatuva tulususe testimise (*backtest*) tööriista loomine, mis hõlbustaks signaalide kombineerimise valikut. Sellise tööriista loomine on see-eest töö autoritel tulevikuplaanides.

6.3 Kliendi tagasiside

Allpool on kirjeldatud kliendi tagasisidet, mis saadi pärast esimese ja teise prototüübi versiooni esitlemist. Kliendi tagasiside põhjal arendati prototüüpi edasi, lisades uusi funktsionaalsusi ning parandades olemasolevaid vigu.

6.3.1 Esimene tagasiside

Esimene ametlik versioon rakendusest esitleti kliendile pärast kõikide tiimiliikmete aktiivset neljakuulist arendustegevust. Ehkki klient oli autorite tööga jooksvalt kursis, oli see esimene isiklik kokkupuude valminud prototüübiga. Kliendil paluti uurida platvormi „värske pilguga“, dokumenteerides platvormile konto loomist, lehel navigeerimise arusaadavust, varade sirvimist, varade detailide vaatamist ning ka ostu-müügisignaalide tellimist. Selle läbi lootsid töö autorid testida peale platvormi funktsionaalsuste ka kasutajaliidese loogilisust (UI) ja kasutajakogemust (UX).

Klient oli esimese versiooniga üldjoontes rahul, kuid leidis mõningaid probleeme ja tähelepanekuid, jagades tagasisidet ka töö autoritega. Peamised täiendust vajavad kohad olid kodulehel, vara detailvaate lehel ning rakendust tutvustaval infolehel, mis kõik vajasid täiendusi.

Avalehel oli küll võimalik pealiskaudselt tutvuda turu hetkeseisuga, kuid klient leidis, et leht oli liiga üldine ning ei pakkunud piisavalt informatsiooni. Ühtlasi oli avalehele jäänud demoandmetega komponent. Ka infolehel oli pealiskaudne jutt, mis rakenduse hetkeversiooniga enam kokku ei sobinud ja seega vajas uuendusi.

Lisaks soovis klient suuremat detailsust ka varade lehele, et oleks võimalik peale hetkehinna ning tehnilise analüüsi signaalide näha arusaadavas vormingus ka vara fundamentaalnäitajad.

Peale selle leidis klient puuduseid vestlusrobotiga rääkides, kelle käest ei olnud tol hetkel võimalik küsida platvormi-teemalisi küsimusi (tehniline abi) ega andmebaasist pärida infot.

Kasutajaliidesega seoses soovitas klient suuremate ekraanimõõtudega seadmetel automaatselt vaskpoolne menüüriba lahti hoida, vältides nii kasutaja ebavajalikku hiireklikki. Ühtlasi soovitas ta vestlusrobotile suuremat visuaalset tähelepanu tõmmata.

Kliendi tagasiside põhjal arendasid töö autorid prototüüpi edasi, kirjutades ümber nii avalehe kui ka varade detailvaate lehe, näitamaks detailset varade kirjeldust, suhtarve ja muid tähtsaid näitajaid. Lisaks integreeriti andmebaas vestlusrobotiga, mille abil oli võimalik robotil pärida ning vastata kasutajale informatiivsemalt. Pärast muudatuste sisseviimist paluti kliendilt uue versiooni kohta tagasisidet.

6.3.2 Teine tagasiside

Rakenduse teine versioon avaldas kliendile muljet: veebilehe vaated olid arusaadavad ning tekst hästi sõnastatud. Komponendid nagu uudised kohandusid paremini erinevate ekraanimõõdetega.

Kliendile meeldis eriti nii harivate materjalide ja artiklite lehekülg kui ka eraldiseisev lehekülg vestlusroboti suhtlusaknaga, mis oli suuremate mõõtmetega ning seega mahutas teksti paremini.

Kodulehekülg oli kliendi sõnul suurepärane, andes kiirülevaate turgudel toimuvast. Ühtlasi soovitas klient pärast kasutaja sisselogimist suunata ta kohe kodulehe peale, mitte „igavale“ kasutajakonto lehele.

Peale mõne lause sõnastuse paranduse oli klient rakendusega vägagi rahul. Tagasiside põhjal kavatsevad töö autorid kui ka klient prototüüpi edasi arendada, seehulgas ka väljaspool antud lõputöö raame.

6.4 Analüüs

Järgmistes alampeatükkides on analüüsitud arendusprotsessi tulemusi ning arutatud ettetulnud probleemide üle.

6.4.1 Projekti arendusprotsess

Lõputöö prototüübi arendusprotsess kestis kokku 21 kuud. Selle perioodi alguses oli arendustöö küll suhteliselt katkendlik. Kogu arendusperioodi jooksul tegeles tööga Janar Takis. Sven Erko Jaroševitš ja Karl Erik Seeder liitusid aktiivse arendustööga viimasel viiel kuul.

Projekti algatasid Janar Takis ja Karl Erik Seeder 2022. aasta septembris aine Veebirakenduse projekt ITI0302 raames, mille käigus valmis algeline veebirakendus. Tol ajal oli

visioon suunatud lühiajalise kaupleja vaatele, et oleks võimalik näha platvormil tehnilise analüüsi signaale. Rakendus oli oma funktsionaalsuste poolest ülimalt piiratud. Ühtlasi oli kasutajaliides lihtsakoeline.

Hiljem otsustati rakendust lõputöö raames edasi arendada, kaasates nii klient kui ka kolmas tiimiliige, Sven Erko Jaroševitš. Projekt viidi lõpule 2024. aasta mais lõputöö esitamisega.

Projekti juhendas ülikooli poolt Gert Kanter, kes pakkus nõu lõputöö kirjutamise osas. Koodipoolsed küsimused suunati Siim Rebasele. Lisaks korraldati aeg-ajalt demoseansse ning küsiti perioodiliselt ka kliendilt nägemust ja tagasisidet.

6.4.2 Tiimitöö

Enne 2024. aasta algust tegeleti projektiga mõned päevad nädalas ilma kindla ajakavata. Alates jaanuarist muutus arendustöö süstemaatilisemaks: korraldati koosolekuid ja kasutati struktureeritumat lähenemist ülesannete haldamisel. Kuna kõik kolm töö autorit ka koolitöö kõrvalt aktiivselt töötasid, siis oli alguses raskusi leida ühist aega, mis kõigile sobiks, tehes arenduse algusetapi ebastabiilseks. Samas töö käigus muutus meeskonnatöö sujuvamaks ja tihti leiti ajapuudusele lahendus une arvelt. Tagasi vaadates oleksid võinud autorid hakata läbimõeldud kava järgi tööga aktiivsemalt tegelema juba varem.

Vaatamata konarlikule algusele oli projekti arendusprotsess edukas. Suhtlus kliendi ja tiimiliikmete vahel oli pidev ning ettetulnud probleemid lahendati kiiresti. Tiimiliikmete vahel sujus koostöö ladusalt ja saavutati põhieesmärk: luua veebirakendus, mis pakub ligipääsu varade fundamentaal- ja hinnainformatsioonile ning võimaldab kasutajatel hõlpsalt teostada tehnilist ja fundamentaalanalüüsi, olles samas kasutajasõbralik ka uutele investoritele. Ka klient oli lõpptulemusega rahul ja hindas valminud prototüüpi positiivselt.

Rakenduse arendamise käigus suhtlesid töö autorid tihti juhendajatega, mis võimaldas autoritel uurida nende lähenemisviise sarnastele probleemidele. Samuti toimus tihe suhtlus tiimiliikmete vahel, kus arutati üheskoos, kuidas kõige mõistlikumalt erinevaid takistusi ületada.

Koodiülevaatusi teostades oli võimalik autoritel süvendada oma teadmisi kasutatavatest raamistikest ning saada ülevaade, kuidas teised tiimiliikmed probleemidele lähenenud ja neid lahendanud.

Kui töö autorid kavatsesid platvormile uusi funktsionaalsusi juurde lisada või olemasolevaid parandada, siis kohtuti koosolekutel ja viidi üksteist olukorraga kurssi. Meeskonnatöö

tulemusena võeti vastu otsuseid, kuidas kõige optimaalsemal kujul funktsionaalsusi lisada.

Arendusprotsessi peamiseks murekohaks oli tiimiliikmete vahel ebavõrdne ajapanuse jaotumine. Nimelt panustas tiimiliige Janar Takis märkimisväärselt rohkem aega projekti kui ülejäänud tiimiliikmed. Antud murekoht oli arusaadav – Janar oli idee algne autor, arendades projekti edasi ka pärast veebirakenduse aine lõppu ning selleläbi panustades kokkuvõttes ka kõige rohkem töötunde.

6.4.3 Nõuete täitmine

Allpool kirjeldatud funktsionaalsed nõuded (peatükk 3.3) said valminud prototüübiga täidetud:

- Turuanalüüs
  - aktsiate fundamentaalnäitajate näitamine,
  - varade hinnagraafikute näitamine,
  - varade ja turgudega üldisemalt seotud uudiste näitamine,
  - ostu- ja müügisignaalide seadistamine ning nende jagamine teistega.
- Portfellihaldus
  - kasutajal on võimalik luua portfelle,
  - kasutajal on võimalik portfelli sisestada käsitsi tehinguid,
  - kasutajal on võimalik vaadata portfelli ülevaadet, tootlust ning jaotust,
  - kasutajal on võimalik näha portfelli hinda soovitud kurssiga.
- Investeerimistööriistad
  - platvormil on saadaval tehisintellekti vestlusrobot,
  - kasutajal on võimalik küsida vestlusrobotilt investeerimisega seotud küsimusi,
  - platvormil on saadaval harivad materjalid, et investeerimisteadmisi arendada.

Järgnevad funktsionaalsed nõuded jäid täitmata:

- Turuanalüüs
  - aktsiate hindamine, leides potentsiaalseid investeerimisvõimalusi,
  - tehniliste analüüsi indikaatorite platvormisisene loomine ja kasutamine.
- Portfellihaldus
  - platvorm oskab hinnata kasutaja portfelli riskitolerantsi ning selle abil anda soovitusi portfelli strateegia muutmiseks.
- Investeerimistööriistad
  - kasutajal on võimalik saada vestlusrobotilt uudiste kokkuvõtteid,

– kasutajal on võimalik saada vestlusrobotilt turgude kiirülevaateid.

6.4.4 Nõuete analüüs

Eelnevalt kirjeldatud nõuete täitmatuse põhjuseks oli ennekõike ajapuudus. Töö käigus valmis küll robustne andmebaasistruktuur ning võimekas tagarakendus, kuid nende arendamine võttis rohkem aega kui algselt planeeritud.

Prototüübi arendamisel oli autoritel võimalik valida kahe lähenemise vahel: kas keskenduda funktsionaalsuste hulgale või nende kvaliteedile. Kuigi suurem funktsionaalsuste hulk oleks võinud pakkuda laiapindsemat kasutajakogemust, otsustasid töö autorid siiski keskenduda rakenduse – eriti tagarakenduse – kvaliteedile ning tulevikupotentsiaalile. See tähendas, et keskenduti sellele, et olemasolevad funktsionaalsused oleksid läbimõeldud ja töötaksid tõrgeteta.

Seega autorite otsus keskenduda rakenduse kvaliteedile tõi kaasa selle, et mõned funktsionaalsed nõuded jäid täitmata. Lisaks tekkisid välisest API-st info küsimisel ja salvestamisel mõned probleemid, mis põhjustasid arendustegevuses viivitusi (vt ptk 6.5.6).

6.4.5 Olemasolevate lahenduste võrdlemine

Varasemalt teostatud olemasolevate lahenduste analüüsi põhjal võivad autorid öelda, et töö eesmärk sai sellegipoolest täidetud. Loodud platvormi on mugav kasutada nii algajatel kui ka edasijõudnutel investoritel. Platvormi kaudu saab kokku koguda ka erinevaid tehnilise analüüsi signaale ja neid teistega jagada, saades koheselt ka nende kohta teavitusi. Lisaks võimaldab platvorm teostada fundamentaalanalüüsi, pakub ligipääsu turu-uudistele ning ka abistavale vestlusrobotile, säilitades kõige selle juures kasutajatele tasuta ligipääsu.

Paljudel olemasolevatel lahendustel on väiksemamahulised võimekused. Peamised turul pakutavate lahenduste puudused on seotud just kauplemissignaalide ning tehisintellekti funktsionaalsustega. Populaarseimad platvormid on tasulised, jäädes hinavahemikku 5 – 549 USA dollarit kuus (vt Lisa 3).

6.4.6 Tehnoloogiliste valikute analüüs

Tehnoloogilised valikud, mis tehti töö alguses, osutusid üldjuhul õigeteks. React, mida kasutati kasutajaliidese loomiseks, võimaldas komponentide loomist lihtsustada ja kiirendada, eriti koos MUI raamistikuga. Spring, mida kasutati tagarakenduse loomiseks, näitas end

võimeka ja usaldusväärse raamistikuna, mis suutis toime tulla ka suure koormusega. Eelnevale lisaks tegi ka PostgreSQL-i skaleeritavus sellest hea valiku andmebaasisüsteemiks.

Prototüübi loomisel osutus üheks suurimaks väljakutseks aga suuremahuliste ajalooliste andmete haldamine. Kuigi TimescaleDB oli efektiivne uute andmete sisestamisel ja *continuous aggregate* vaadete loomisel, muutus see suure hulga ajaloolise hinnainfo sisestamisel ikkagi aeglaseks. Probleem tulenes sellest, et vaadete uuendamisel pidi SQL kogu tabeli läbi vaatama ning vaate nullist looma. Mida rohkem varasid süsteemis aktiveeriti, seda pikemaks venis ka vaadete uuendamise aeg. See nõudis tagarakenduses täiendava loogika väljaarendamist, mis võimaldaks ajastada vaadete uuendamise madala aktiivsusega kellaajale, vältides kasutajate tegevuse liigset segamist andmebaasi intensiivse töö ajal.

Töö autorid kaalusid iga uue aktiveeritud vara hinnainfo jaoks uue tabeli loomist, kuid see oleks võinud tekitada veel suuremaid probleeme, muutes ühtlasi tabelite haldamise keerulisemaks: paarikümne tabeli asemel oleks olnud vaja hakkama saada mitme tuhande tabeliga. Ühtlasi võis nii suure arvu tabelite loomine, haldamine ja perioodiline uuendamine võtta serverilt nii palju mälu, et see oleks võinud seiskuda. Antud probleemile ei leitudki head lahendust – ainus võimalus tundub olevat serveri jõudluse – eesotsas mälu – suurendamine.

Ehkki TimescaleDB hüpertabeli *timescaledb.compress* funktsioon vähendas ajaloolise hinnainfo tabeli mahtu pea neli korda, tuli uute andmete sisestamisel tabeli vastav osa lahti pakkida ning pärast sisestamist uuesti kokku pakkida. See protsess võttis aega ja oli ressursimahukas, eriti kui sooviti mitu uut vara korraga sisestada. Probleemi lahendamiseks kasutasid autorid eraldi tabelit, kuhu ajutiselt salvestati uute varade hinnainfo. Pärast hinnainfo esialgset sisestamist tõsteti uued andmed ühe korraga õigesse hüpertabelisse. Sellise lahendusega oli andmebaasil vaja ainult ühe korra hüpertabelit lahti ja kokku pakkida.

Töö käigus tekkis murekoht ka andmebaasipäringute osas: kuidas filtreerida, pagineerida või otsida andmeid tõhusalt vastavalt erinevatele parameetritele ja kasutaja sätestatud nõuetele. Filtreerimisprobleemi lahendasid töö autorid oma implementatsiooniga Springi *CriteriaBuilder'ist*, mis suutis vastavalt konfiguratsioonifailile infot filtreerida, grupeerida, valides soovitud väljad ning lõpuks soovitud objektid tagastada (vt Lisa 6). Arvutuste sooritamiseks lõid töö autorid ka eraldi klassi nimega ConfigField, mille korrektselt seadistamisel sai kalkuleerida pea kõiki soovitud matemaatilisi avaldisi, mida SQL lubas.

Sarnase lahenduse arendasime ka TimescaleDB hüpertabelitele, mis ei sobitunud puuduva

primaarvõtme tõttu kokku Java ORM implementatsiooniga. Selleks pidid töö autorid looma omaenda loogika selliste predikaatide ehitamiseks. See otsustati teha võimalikult sarnaseks Springi *CriteriaBuilder'le*, eesmärgiga hoida funktsionaalsus ning kasutusprotsess tuttav. Kuid tulemus oli siiski märgatavalt primitiivsem kui Springi enda implementatsioon – peamine puudujääk oli tabelite liitmise (*join*) võimalus, kuid kuna väljatöötatud päringute süsteem lahendas probleemi väga hästi, ei näinud töö autorid vajadust seda edasi arendada.

Kuna platvorm vajas ligipääsu detailsele finantsinfole, siis teostati analüüs võrdlemaks erinevaid API teenusepakkujaid (vt ptk 4.3). Autorid leidsid, et kuigi paljudel API-del oli iseenesest väga palju ja kvaliteetseid andmeid, siis ei lubatud neid andmestikke kas hoiustada või jagada kolmandate osapooltega. Töö raames ei sobinud selliseid API-sid kasutada. Oli ka variante, kus andmeid lubati jagada kolmandate osapooltega, aga sellised variandid olid väga kulukad. Viimaks leidsid töö autorid Twelve Data API, mis pakkus mõistlikku hinnaga kommertslitsentsi, varade valik oli lai ning lubas ka teha mitusada päringut enda API pihta minutis. Selline lahendus sobis antud projektiga ideaalselt.

Kuid ka aktsiainfo töötlemisel esines probleeme. Twelve Data API-st tuli vahel ebaühtlast infot kursside kohta, mõnikord näiteks sentides, kuigi hinna teisendamist sentideks ei eksisteerinud. See muutis hinna otsimise ebaselgeks. Lisaks pidid töö autorid aktsiate hinna ja fundamentaalinfo esmalaadimisel looma erinevad protsessid, kuna API päringute arv oli piiratud ja päringute krediit sai pidevalt otsa.

Vestlusroboti loomine tõi kaasa mitmeid väljakutseid. Algselt plaanisid autorid välja arendada täiesti uue AI-mudeli, kuid varsti sai selgeks, et mõne kuuga ei ole võimalik saavutada soovitud kvaliteeti. Järgmine idee oli kasutada pooltreenitud mudeleid, kuid need olid kas liiga algelised või treenitud liiga spetsiifiliste andmetega, mis muutis nende kasutamise keeruliseks. Lõpuks otsustasid autorid ümber treenida juba väljaõpetatud mudeli.

Enne kliendi soovide kirjapanemist uurisid autorid iseseisvalt huvi pärast erinevaid AI platvorme: Zapier ja ChatGPT Plus. Kui klient esitas oma soovitud funktsionaalsused, ei sobinud need platvormid, kuna neil puudus eraldiseisvate funktsionaalsuste lisamine. Hiljem avastasid autorid, et kliendi vajalikke funktsionaalsusi on võimalik täita OpenAI Assistants API vahendusel.

Vestlusroboti poolel tekitas probleeme ka Pythoni programmi Flask veebiserver. Selle raamistiku sisseehitatud veebiserverit ei soovitata kasutada suure koormusega avalikes toodetes. Seetõttu liikusid töö autorid üle võimsamale Tornado veebiserverile.

Enne kliendi soovide kirjapanemist uurisid autorid iseseisvalt huvi pärast erinevaid AI platvorme: Zapier ja ChatGPT Plus. Kui klient esitas oma soovitud funktsionaalsused, ei sobinud need platvormid, kuna neil puudus exaraldiseisvate funktsionaalsuste lisamine. Hiljem avastasid autorid, et kliendi vajalikke funktsionaalsusi on võimalik täita OpenAI Assistants API vahendusel.

Vestlusroboti poolel tekitas probleeme ka Pythoni programmi Flask veebiserver. Selle raamistiku sisseehitatud veebiserverit ei soovitata kasutada suure koormusega avalikes toodetes. Ehkki antud prototüübi puhul ei olnud raamistiku valik probleemne, tuleb see tulevikus välja vahetada.

Oluline osa ajast kulus ka vana koodi ümber kirjutamisele: uute ja paremate lahenduste õppimisel soovisid autorid need ka koheselt implementeerida. Ehkki selle tulemusel on koodibaas arusaadavam, universaalsem ning seega edasiarenduste jaoks valmis, jõuti sellepärast vähem funktsionaalsuseid valmis.

Samuti tekitas probleeme e-kirjade edastamise platvormide algne keeldumine anda töö autoritele SMTP serverile juurdepääsu. See nõudis aega ja suhtlust teenusepakkujaga, et lõpuks vajalik juurdepääs saada.

Ehkki autorid olid teadlikud töö mahust, osutus lõputöö arvatust veelgi keerukamaks. Samuti oleks tiimiliikmetel võinud olla parem ajaplaneerimine, vältimaks viimase hetke kiirustamist ning ennetades ootamatuid probleeme.

6.5 Edasised arendusvõimalused

Veebirakenduse prototüübi arendamisel loodi tugev alus suure hulga varade andmete haldamiseks. Tulevikus võiks arendustöö keskenduda funktsionaalsetele nõuetele, mida antud töö autoritel ei õnnestunud täielikult realiseerida. Näiteks võiks hinnata kasutaja riskitaluvust algoritmide või närvivõrkude abil ning pakkuda soovitusi portfelli muutmiseks. Peale selle võiks ka vestlusrobotit täiendada, lisades võimaluse koostada uudiste kokkuvõtteid ja pakkuda turgudest kiirülevaateid.

Lisaks võiks kasutada masinõpet, et hinnata varadega seotud uudiste mõju. Selleks võiks luua mudeli, mis analüüsib, kas uudis on negatiivne ja kuidas see võiks mõjutada vara hinda. Kuna andmebaasis on olemas vajalik info varade kohta, võiks masinõppe abil luua mudeli, mis arvutab varadele riskiskoori. Selleks seoses oleks aga vaja leida eraldiseisev andmestik, mis sisaldaks analüütikute hinnanguid ning kasutada seda treeningandmestikuna.

Tehnilise analüüsi poole pealt on võimalik prototüübi andmebaasis sisalduva hinnainfo põhjal luua indikaatoreid. See võimaldaks kasutajatel otse platvormi siseselt hallata ostu- ja müügisignaale, ilma et neid peaks saatma väliselt platvormilt nagu TradingView.

Lisaks eeltoodule on veel palju võimalusi varade andmete uurimiseks ja analüüsimiseks. Näiteks võiks luua kasutajale võimaluse siduda oma investeerimisportfell investeerimistehingute platvormidega. See võimaldaks automatiseerida tehingute tegemist ning ühtlasi näha nende tehingute mõju portfellile.

7. Kokkuvõte

Antud lõputöö eesmärk oli luua finantsanalüüsi platvorm, mis oleks intuitiivne ja kasutajasõbralik nii algajatele kui ka kogenud investoritele ja kauplejatele. Platvorm võimaldaks kasutajatel teostada varadele nii tehnilist kui ka fundamentaalanalüüsi ning kasutada erinevaid investeerimistööriistu.

Töö käigus uuriti olemasolevaid lahendusi, tuvastati nende puudused ja koostöös kliendiga Darbinvest OÜ sätestati platvormi nõuded. Arendusprotsessi tulemusena valmis veebirakenduse prototüüp koos tehisintellekti-vestlusrobotiga.

Töö alguses esitatud probleemile, milleks oli heade investeerimingute ülevaate ja analüüsimis platvormi puudumine, sai lahendus loodud. Tehtud töö tulemusena valmis robustne tagarakendus ning laiendatav andmebaas, mis lubab ligipääsu nii fundamentaalsele kui ka tehnilisele analüüsile, hoides ühtlasi rakenduse kasutajaliidese kasutajasõbralikuna. Rakenduse loomisel on lähtutud kasutatavuse mustritest, et platvormi oleks lihtne kasutada nii algajatel kui ka edasijõudnutel.

Prototüüp täidab valdava enamuse esialgselt püstitatud nõuetest, võimaldades sisse logitud kasutajatel suhelda vestlusrobotiga, küsides sellelt investeerimisalaseid nõuandeid ja infot erinevate varade kohta. Samuti pakub rakendus võimalust jälgida varade ajaloolist hinnainfot, tutvuda aktsiate fundamentaalnäitajatega ning konfigureerida ja jagada ostu- või müügisignaale.

Prototüüp ei ole veel suuteline looma automaatset hinnangut aktsiatele, platvormisiseseid tehnilisi analüüsi indikaatoreid ega andma soovitusi portfellile. Kuigi mõned nõuded jäid täitmata, vastab loodud tarkvara sellegipoolest lõputöö algsetele eesmärkidele: rakendus ühendab turuanalüüsi, portfellihalduse ja tehisintellekti võimalused ning on ka kliendile meelepärane.

Antud tööd kavatsevad autorid edasi arendada, lisades uusi funktsionaalsusi ja parandades olemasolevaid. Tulevikus on plaanis lisada kasutajate riskitolerantsi hindamine ja soovituste andmine portfelli koostamiseks, uudiste analüüsimiseks masinõppe mudel ning kasutajate portfellide sidumine väliste investeerimisplatvormidega.

Kasutatud kirjandus

[1] Derek Saul. *Retail Trading Just Hit An All-Time High. Here's What Stocks Are The Most Popular*. [Võrgumaterjal]. [Kasutatud 11.05.2024]. 2023. URL: https://www.forbes.com/sites/dereksaul/2023/02/03/retail-trading-just-hit-an-all-time-high-heres-what-stocks-are-the-most-popular/.

[2] Danielle Zanzalari. *Why Is Investing Important?* [Võrgumaterjal]. [Kasutatud 11.05.2024]. 2022. URL: https://www.thebalancemoney.com/why-is-investing-important-5222360.

[3] *Business Finance, Stock Market, Quotes, News*. [Võrgumaterjal]. [Kasutatud 18.02.2024]. URL: https://finance.yahoo.com.

[4] *Stock Market News - Financial News*. [Võrgumaterjal]. [Kasutatud 18.02.2024]. URL: https://www.marketwatch.com.

[5] *Free Stock Charts, Stock Quotes and Trade Ideas*. [Võrgumaterjal]. [Kasutatud 09.05.2024]. URL: https://www.tradingview.com.

[6] Derek Saul. *Pine Script™ v5 User Manual*. [Võrgumaterjal]. [Kasutatud 09.05.2024]. 2023. URL: https://www.tradingview.com/pine-script-docs/en/v5/Introduction.html.

[7] *10 amazing TradingView statistics and facts*. [Võrgumaterjal]. [Kasutatud 11.05.2024]. 2017. URL: https://www.tradingview.com/blog/en/10-amazing-tradingview-statistics-facts-5469.

[8] Katre-Helena Käppa. *Tallinna börsi ettevõtete finantsandmete veebirakendus*. [Võrgumaterjal]. [Kasutatud 09.05.2024]. 2021. URL: https://digikogu.taltech.ee/et/Item/826e0097-6288-4b13-9cec-851254c02c0a.

[9] *MQL5 Algo Trading community*. [Võrgumaterjal]. [Kasutatud 09.05.2024]. URL: https://www.mql5.com/en/about.

[10] *MetaTrader 5 Trading Signals with Automatic Execution*. [Võrgumaterjal]. [Kasutatud 09.05.2024]. URL: https://www.mql5.com/en/signals/mt5.

[11] *Quotas for Google Services*. [Võrgumaterjal]. [Kasutatud 09.05.2024]. URL: https://developers.google.com/apps-script/guides/services/quotas.

[12] *About GitLab*. [Võrgumaterjal]. [Kasutatud 11.05.2024]. URL: https://about.gitlab.com/.

[13] Henri Keerutaja. *Veebirakendus eraisiku investeeringute halduseks*. [Võrgumaterjal]. [Kasutatud 03.01.2024]. 2023. URL: https://digikogu.taltech.ee/et/Item/f54d4374-774f-42fa-ae0b-80e28484145c.

[14] Liis-Marie Kütt. *Algkooliõpilaste rahaliste vahendite haldamise rakenduse arendus*. [Võrgumaterjal]. [Kasutatud 03.01.2024]. 2021. URL: https://digikogu.taltech.ee/et/Item/f54d4374-774f-42fa-ae0b-80e28484145c.

[15] *PostgreSQL ++ for time series and events*. [Võrgumaterjal]. [Kasutatud 18.02.2024]. URL: https://www.timescale.com/.

[16] *Meet our customers*. [Võrgumaterjal]. [Kasutatud 07.02.2024]. URL: https://www.timescale.com/case-studies.

[17] *Hypertables*. [Võrgumaterjal]. [Kasutatud 07.01.2024]. URL: https://docs.timescale.com/use-timescale/latest/hypertables.

[18] *About time buckets*. [Võrgumaterjal]. [Kasutatud 07.01.2024]. URL: https://docs.timescale.com/use-timescale/latest/time-buckets/about-time-buckets/.

[19] *Hierarchical continuous aggregates*. [Võrgumaterjal]. [Kasutatud 07.01.2024]. URL: https://docs.timescale.com/use-timescale/latest/continuous-aggregates/hierarchical-continuous-aggregates/.

[20] Simon Maple. *JVM Ecosystem report 2018 - About your Platform and Application*. [Võrgumaterjal]. [Kasutatud 09.05.2024]. 2018. URL: https://snyk.io/blog/jvm-ecosystem-report-2018-platform-application/.

[21] *Why Spring?* [Võrgumaterjal]. [Kasutatud 09.05.2024]. URL: https://spring.io/why-spring.

[22] *Introduction to Liquibase*. [Võrgumaterjal]. [Kasutatud 09.05.2024]. URL: https://docs.liquibase.com/concepts/introduction-to-liquibase.html.

[23] *Twelve Data Services Agreement*. [Võrgumaterjal]. [Kasutatud 22.11.2023]. URL: https://twelvedata.com/terms.

[24] *JavaScript developer ecosystem*. [Võrgumaterjal]. [Kasutatud 21.09.2023]. URL: https://www.jetbrains.com/lp/devecosystem-2023/javascript/.

[25] *React*. [Võrgumaterjal]. [Kasutatud 10.05.2024]. URL: https://react.dev/.

[26] *Sass: Syntactically Awesome Style Sheets*. [Võrgumaterjal]. [Kasutatud 09.05.2024]. URL: https://sass-lang.com/.

[27] *Material UI - Overview*. [Võrgumaterjal]. [Kasutatud 11.05.2024]. URL: https://mui.com/material-ui/getting-started/.

[28] Marquel Ellis. *Python AI: A Beginner's Guide*. [Võrgumaterjal]. [Kasutatud 11.05.2024]. 2024. URL: https://blog.hubspot.com/website/python-ai.

[29] *The all-in-one platform to build AI Agents*. [Võrgumaterjal]. [Kasutatud 11.05.2024]. URL: https://www.voiceflow.com/features/platform-overview.

[30] *What is a container?* [Võrgumaterjal]. [Kasutatud 11.05.2024]. URL: https://www.docker.com/resources/what-container/.

[31] *What Is NGINX?* [Võrgumaterjal]. [Kasutatud 11.05.2024]. URL: https://www.nginx.com/resources/glossary/nginx/.

[32] *What is Scrum?* [Võrgumaterjal]. [Kasutatud 10.05.2024]. URL: https://www.scrum.org/resources/what-scrum-module.

[33] *About configuration in TimescaleDB*. [Võrgumaterjal]. [Kasutatud 12.05.2024]. URL: https://docs.timescale.com/self-hosted/latest/configuration/about-configuration/.

[34] Tom Collins. *Controller-Service-Repository*. [Võrgumaterjal]. [Kasutatud 11.05.2024]. 2021. URL: https://tom-collings.medium.com/controller-service-repository-16e29a4684e5.

[35] Priyank Gupta. *Understanding the modular monolith and its ideal use cases*. [Võrgumaterjal]. [Kasutatud 02.01.2024]. 2020. URL: https://www.techtarget.com/searchapparchitecture/tip/Understanding-the-modular-monolith-and-its-ideal-use-cases.

[36] *What is REST API?* [Võrgumaterjal]. [Kasutatud 11.05.2024]. 2020. URL: https://www.redhat.com/en/topics/api/what-is-a-rest-api.

[37] *Email Sending that just works*. [Võrgumaterjal]. [Kasutatud 12.05.2024]. URL: https://mailtrap.io/email-sending/.

[38] *9 Behavioural patterns for interactive design*. [Võrgumaterjal]. [Kasutatud 11.05.2024]. 2021. URL: https://www.harmonycb.com/patterns-for-effective-interactive-design/2021/08/.

[39] *TradingView Widgets*. [Võrgumaterjal]. [Kasutatud 12.05.2024]. URL: https://www.tradingview.com/widget-docs/widgets/.

[40] *Introduction to JSON Web Tokens*. [Võrgumaterjal]. [Kasutatud 12.05.2024]. URL: https://jwt.io/introduction/.

[41] Mark LaMonica. *What is your investing edge?* [Võrgumaterjal]. [Kasutatud 12.05.2024]. 2018. URL: https://www.morningstar.com.au/insights/personal-finance/164063/what-is-your-investing-edge.

[42] Bailey Maybray. *Color Psychology: How To Use it in Marketing and Branding*. [Võrgumaterjal]. [Kasutatud 12.05.2024]. 2023. URL: https://blog.hubspot.com/the-hustle/psychology-of-color.

[43] *Lõputöö tulemusel valminud veebileht*. [Võrgumaterjal]. [Kasutatud 18.02.2024]. URL: https://investingedge.pro/.

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

Meie Janar Takis, Sven Erko Jaroševitš, Karl Erik Seeder

1. Anname Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Veebipõhine Finantsanalüüsi platvorm", mille juhendajad on Siim Rebane and Gert Kanter
   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. Oleme teadlikud, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autoritele.
3. Kinnitame, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

27.05.2024

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

Lisa 2 - Ülevaade meeskonnaliikmete panusest ning peamistest tegevustest

See peatükk annab ülevaate iga meeskonnaliikme panusest ja tegevustest projekti jooksul. Projekti teostamise käigus jaotusid vastutusalad järgnevalt. Janar võttis enda peale suurema osa kasutajaliidese arendusest, kuigi aeg-ajalt, vastavalt vajadusele, arendas kasutajaliidest ka Sven. Tagarakenduse arenduse osas töötasid Sven ja Janar tihedas koostöös, jagades ülesandeid ja vastutust. Vestlusroboti arenduse ning platvormi legaalse poole eest vastutas Karl.

Janar

Mina olin investeerimisteemalise veebirakenduse idee algne autor. Minu huvi investeerimise ja finantsturgude valdkonnas viis mind selleni, et soovisin luua rakenduse, mis integreeriks tehnilise analüüsi ja lühiajalise kauplemise funktsioonid. See idee oli inspireeritud minu isiklikust vajadusest sellise tööriista järele, eriti seoses signaalide omavahelise kombineerimise ja jagamisega.

Võtsin enda kanda ka meeskonna juhi rolli. See tähendas, et minu ülesandeks oli koordineerida suhtlust meeskonna ja kliendi vahel ning juhtida üldist arendusprotsessi. Lisaks sellele osalesin ka aktiivselt rakenduse arendamises.

Minu peamised tööülesanded olid seotud andmebaasistruktuuri ja TimescaleDB haldamisega. See võimaldas meil tõhusalt hallata ja analüüsida ajapõhiseid andmeid, mis oli oluline osa meie rakenduse funktsionaalsusest.

Lisaks töötasin ma turuanalüüsi funktsionaalsete nõuetega. Arendasin selles kategoorias tehnilise analüüsi ostu- ja müügisignaalide äriloogikat.

Eesrakenduse arendamine oli samuti minu vastutusalas, arendades üle kolmveerandi vaadetest ning funktsionaalsustest. Üks suurimaid väljakutseid oli leida tasakaal algajatele suunatud mugavuste ja edasijõudnutele mõeldud funktsionaalsuste vahel. Selleks kasutasin kasutatavuse mustreid ning kliendi tagasisidet. Ma püüdsin luua sellise kasutajaliidese, mida oleks intuitiivne ja lihtne kasutada, kuid samas pakuks ka piisavalt võimalusi edasijõudnutele. Ma usun, et see tuli mul hästi välja, arvestades ka kliendi positiivset tagasisidet. Kasutajaliides täitis oma eesmärgid ning mul on ka hea arusaam, kuidas selle arendamisega

edasi liikuda.

Oma koodi arendamisel olin ma perfektsionist. Püüdsin teha nii ees- kui tagarakenduse komponendid nii universaalseteks kui võimalik, proovides vähendada koodi kordamist ja parandada koodi kvaliteeti. See tähendas ühtlasi teiste tiimiliikmete koodistiili parendamist. Kuigi see võttis aega, usun, et see investeering tasus end ära, kuna see aitab meil tulevikus rakendust lihtsamini laiendada ja hooldada. Kahjuks tähendas see aga seda, et lõputöö raames ei suutnud ma kõike endapoolseid nõudeid täita.

Minu kõige suuremad õppetunnid olid seotud andmebaasidega. TimescaleDB hingeelu tundmaõppimine oli keeruline, kuid samas ka väga silmi avardav. Sain aru, kuidas see andmebaas töötab, kuidas see andmeid salvestab ja kuidas kiiresti ja tõhusalt ajapõhiseid päringuid teha. Neid teadmisi ja oskusi saan ka tulevikus ära kasutada.

Lisaks TimescaleDB hingeelu tundmaõppimisele sain ma ka palju praktilisi kogemusi andmebaaside haldamisega. Õppisin, teha komplekssseid päringuid ja seda, kuidas optimeerida andmebaasi jõudlust. See oli minu jaoks uus kogemus, kuna ma ei olnud varem sellises mahus andmebaasidega tegelenud.

Meeskonnatöö oli meie projektis olulisel kohal. Suhtlus tiimiliikmete vahel oli hea ja suutsime koos kiirelt lahendada tekkinud probleemid. Olen oma tiimiliikmete üle uhke – ehkki algus oli arendus kiiruse mõttes konarlik, usun, et lõpptulemus vastas nii minu kui tiimiliikmete kõrgetele standarditele.

Sven

Mul oli ka varasem kogemus investeerimisega, kuid puudusid teadmised aktiivselt kauplemisest. Olin ka varem teiste veebirakenduste raames loonud investeerimisportfellide funktsionaalsusi. Kahjuks ei leidnud ma head lahendust portfelli analüüsimiseks (hajutatus, ajalooline muutus võrreldes minu ostuhetkega). Pangad ja erinevad investeerimisplatvormid pakkusid küll ajaloolise hinna vaatlusvõimalusi, kuid seal sai vaadata vaid varasid, mis olid ostnud nende platvormil. Seega puudus täiemahuline ülevaade oma investeeringutest ning nende käekäigust.

Projekti arendades võtsin ma suurema rolli tagarakenduse loomisel, sest olin Springiga rohkem kokku puutunud kui Reacti ja JavaScriptiga. Peamine roll oli minul portfellihaldusloogika loomisel, kus sain mõelda portfelli äriprotsesside peale ning ka sellest, kuidas esirakendusel oleks mõistlik portfelle kuvada ja kuidas käsitleda erinevate aktsia hinnakursside konverteerimist nii, et kasutaja saaks hea mugava ülevaate oma portfellist.

Tagarakendust luues proovisin komponente luua ka võimalikult dünaamiliselt, vältides arendamisel koodi duplikeerimist. Tänu sellele oli üks esimesi asju, mis ma meie koodibaasis tegin see, et lõin ühise filtreerimise- ja pagineerimissüsteemi. Kuna kõik meie kasutajaliideses olevad tabeli vaated olid sama loogikaga, siis oli tagarakendusse võimalik luua ühine filtreerimissüsteem üsnagi hõlpsalt. Algul oli muidugi filtreerimissüsteem palju piiritletum kui hiljem, kuna nõuded läksid aina laiemaks. Mina lõin ka süsteemi selliste andmete filtreerimiseks, mis polnud traditsioonilised Spring Entity'd ehk näiteks ajalooline hinninfo.

Mul oli ka suur roll tagarakenduses varade initsialiseerimisprotsesside kirjutamisel, mis oli lõpuks märgatavalt keerulisem, kui me algselt ootasime. Samuti tekitasin veel tagarakenduses teavituste süsteemi, meilisaatmise ja kasutaja kontokinnitussüsteemi ning abistasin tagarakenduses ka muude süsteemidega, millega teised tiimiliikmed abi soovisid. Proovisin ka kasutajaliidesesse osadele süsteemidele tuge arendada.

Minu suurim õppetund oli see, et isegi kui ülesanne kõlab idee poolest lihtsalt, siis tasub see põhjalikult läbi mõelda ja ei tasu alahinnata erinevaid alamülesandeid, sest need võivad olla märgatavalt keerulisemad kui esmapilgul tunduvad.

Mina tunnen, et meie meeskonnatöö sujus tiimis hästi ning olime alati üksteise jaoks olemas kui oli vaja nõu küsida. Tänu sellele tiimitööle minu arvates tuli ka lõpp-tulemus palju parem ning läbimõeldud, kui seda individuaalselt tehes.

Karl Erik

Kui olin veel teise kursuse tudeng, võtsin koos Janariga ette veebirakenduse projekti aines ITI0302. Meie tiimisisene koostöö sujus hästi ja aine lõpus pakkus Janar välja, et võiksime projekti edasi arendada lõputööna. Selleks valideerisin idee erinevate õppejõududega ja saime peagi positiivse tagasiside, mis võimaldas meil ametlikult hakata tunde kirja panema.

Alguses mõtlesin, et on vara hakata lõputööga tegelema ning keskendusin teistele ülesannetele. Kuigi Janar hoidis mind pidevalt kursis arendustöödega ning arutasime koos lähenemisviise ja olulisi nüansse, jäi mul lõputöö algus veidi tagaplaanile.

Minu peamised tööülesanded olid seotud AI-ga ja legaalse poolega. Kui aktiivselt hakkasin lõputööga tegelema, olid meeskonna rollid juba suures osas jaotatud: Janar tegeles rohkem eesrakendusega ja Sven tagarakendusega. Kliendi soov AI-tehnoloogia järele pani mind keskenduma sellele valdkonnale.

Lisaks tegelesin enamasti lõputöö dokumendi koostamise ja esitlustega. Samuti GitLabi ülesannete kujundamise ja dokumenteerimisega. Legaalsesse poole peal panustasin *Terms and Conditions*'i ja *Privacy Policy* kaudu, et tagada, et meie tulevane tegevus oleks seaduslik ja vastaks Eesti õigusnõuetele.

GenAI-ga seotud probleemid tekkisid peamiselt vestlusroboti testimisega. Kuna mul puudus varasem kogemus sellise arendusega, osutus probleemide silumine väga keeruliseks ja aeganõudvaks. Ma ei suutnud alati aru saada, kas süsteem töötab soovitud viisil või on midagi valesti. See probleem tuli esile, kui oli vaja testida väiksemaid detailseid aspekte, eriti seoses teadmiste põhjaga.

Kogu selle protsessi käigus õppisin, et tiimitöö ja suhtlus on võtmetähtsusega ning tuleb alati prioritiseerida prototüübi valmimist – mõned asjad võivad rohkem tagaplaanile jääda, et suuremas pildis saaksime edasi liikuda.

Mina tunnen, et meie meeskonnatöö sujus hästi ning olime alati üksteise jaoks olemas. Kõik tiimiliikmed olid abivalmid ja vastuvõtvad kriitikale ja väidetele, mis aitas meil teha palju parema ja stabiilsema lõpp tulemuse.

Lisa 3 – Olemasolevate lahenduste funktsionaalsused

Järgnevalt on välja toodud tabel võrdlemaks konkureerivate lahenduste funktsionaalsuseid välja arendatud prototüübiga. Käesoleva töö käigus valminud rakendus suutis konkureerivatest lahendustest olla esikohal.

		General features				*Extra features*					*Stock-specific features*		
Investing platform	Score	Asset quotes	Charts and graphs	Market news	User friendly UI	Watch lists	Research and education	Paper trading simulator	Advanced analysis	AI features	Company profiles	Financial statements	Options analysis
InvestingEdge	14	+	+	+	+	~	+	~	~	+	+	+	-
https://www.macroaxis.com/investment-tools	12	+	+	+	+	-	-	-	+	+	+	+	+
https://capitalise.ai/	12	+	+	-	+	+	-	+	+	+	-	-	-
https://wundertrading.com/en	11	+	+	-	+	-	-	+	-	+	-	-	-
https://www.gurufocus.com/widgets	11	+	+	-	-	-	+	+	+	-	+	+	-
https://www.marketwatch.com/	11	+	+	+	-	+	+	+	-	-	+	+	-
https://alertatron.com/	11	+	+	-	+	-	-	-	+	-	-	-	-
https://finance.yahoo.com/	10	+	+	+	-	+	-	-	-	-	+	-	-
https://www.investopedia.com/	10	+	-	+	+	-	+	+	-	-	+	-	-
https://www.google.com/finance/	10	+	+	+	+	+	-	-	-	-	+	+	-
https://stockcharts.com/freecharts/	10	+	+	+	-	+	+	-	-	-	+	+	-
https://blackboxstocks.com/features/	10	+	+	+	-	+	-	-	-	-	+	-	+
https://www.cnbc.com/pre-markets/	10	+	+	+	+	+	+	-	-	-	+	+	-
https://www.invrs.com/home	10	+	+	+	+	+	+	-	-	-	+	-	-
https://tradersync.com/	8	+	+	-	-	+	-	-	-	+	-	-	-
https://krown-trading.teachable.com/courses/	7	-	-	-	+	-	+	-	+	-	-	-	-
https://metasignals.io/	6	-	-	-	+	-	-	-	+	-	-	-	-
https://alarumist.com/	6	-	-	-	+	-	-	-	-	-	-	-	-
https://www.wolframalpha.com/input?key=&i=TSLA	5	+	+	-	-	-	-	-	-	-	+	+	-
https://github.com/alleyway/add-tradingview-alerts-tool	3	-	-	-	-	-	-	-	-	-	-	-	-

Joonis 13. *Valminud prototüübi ja olemasolevate platvormide funktsionaalsused 1*

		Trading tools					*Portfolio features*		*Social*		*Pricing*		
Investing platform	Score	Integration to external charting platform	Combining buy and sell signals	Sharing signals with others	Integration to trading platform	Algorithmic trading	Risk management tools	Portfolio management	Community features	Customer support	Basic features free?	Advanced features free?	Price / mo
InvestingEdge	14	+	+	+	-	~	~	+	~	+	+	+	$0.00
https://www.macroaxis.com/investment-tools	12	-	-	-	-	-	+	+	-	-	+	-	$39.00
https://capitalise.ai/	12	+	-	-	+	+	+	-	-	-	+	+	$0.00
https://wundertrading.com/en	11	+	-	+	+	+	+	-	-	+	-	-	$5.00
https://www.gurufocus.com/widgets	11	-	-	-	-	-	+	+	+	+	-	-	$41.00
https://www.marketwatch.com/	11	-	-	-	-	-	+	-	-	+	+	-	$10.00
https://alertatron.com/	11	+	+	+	+	+	+	-	+	-	-	-	$30.00
https://finance.yahoo.com/	10	-	-	-	+	-	+	+	+	-	+	+	$0.00
https://www.investopedia.com/	10	-	-	-	-	-	+	+	+	+	-	-	$0.00
https://www.google.com/finance/	10	-	-	-	-	-	+	-	+	-	+	+	$0.00
https://stockcharts.com/freecharts/	10	-	-	-	-	-	+	+	-	-	+	-	$20.00
https://blackboxstocks.com/features/	10	-	-	-	+	-	+	-	+	+	-	-	$50.00
https://www.cnbc.com/pre-markets/	10	-	-	-	-	-	-	-	-	+	+	+	$0.00
https://www.invrs.com/home	10	-	-	-	+	-	-	-	+	-	+	-	$7.00
https://tradersync.com/	8	-	-	-	+	-	+	+	-	+	-	-	$30.00
https://krown-trading.teachable.com/courses/	7	+	-	-	-	+	+	-	-	+	-	-	$200.00
https://metasignals.io/	6	+	-	-	-	+	+	-	-	+	-	-	$549.00
https://alarumist.com/	6	+	-	-	+	+	-	-	+	-	+	-	$10.00
https://www.wolframalpha.com/input?key=&i=TSLA	5	-	-	-	-	-	-	-	-	-	+	-	$10.00
https://github.com/alleyway/add-tradingview-alerts-tool	3	+	-	-	-	+	-	-	-	-	+	+	$0.00

Joonis 14. *Valminud prototüübi ja olemasolevate platvormide funktsionaalsused 2*

Lisa 4 – Väliste API-de võrdlus

Finantsinfot otsustati pärida Twelve Data API-st. Antud API pakub laia valikut andmeid, sealhulgas aktsiate fundamentaalnäitajaid ning ajaloolist hinnainfot nii aktsiatele, valuutadele kui ka digitaalvaradele. Peale laiale infovalikule osutus Twelve Data kasuks mõistlik ja arusaadav hind, detailne dokumentatsioon ning õigus infot oma andmebaasi salvestada, kasutajatele kuvada ning ka rakenduses sisemiselt kasutada [23]. Teised API-d olid kas liiga kallid, ei lubanud andmeid sisemiselt kasutada või puudus litsentsis vastav selgitav klausel.

API provider	ToS link	Notes	*Commercial usage*			
			Price / mo	Store	Display	Create derivatives
AlphaVantage	https://www.alphavantage.co/terms_of_service/	You can install, use, access, display, and run the software for personal, non-commercial use. However, if you plan to use it for commercial purposes, including investment analysis, research, testing, monitoring, or any other commercial activity, you need to contact them for further arrangements.	?	Unknown	Unknown	Unknown
TwelveData	https://twelvedata.com/terms https://support.twelvedata.com/en/articles/5332349-commercial-and-personal-usage	Where explicitly stated on Twelve Data, the User may download, copy and/or share some content available through Twelve Data for its sole personal, commercial, and non-commercial use and provided that the copyright attributions and all the other attributions requested by us are correctly implemented. Pro+ Tier Plans You (individual or a company) may use data supplied by Twelve Data for commercial purposes by displaying the data for third parties, redistributing it, or for internal use.	100	Yes	Yes	No
EODHD	https://eodhd.com/financial-apis/terms-conditions/	Students can take advantage of our 50% discount on all our subscriptions for the development of internal projects and any educational or scientific purposes. Please contact sales to request a quote at support@eodhistoricaldata.com Commercial subscribers can potentially be granted permission to sell, redistribute, display, provide, or grant access to any EOD Historical Data Information or the Services in a repackaged form, but this requires prior	?	Yes	Yes	Unknown
FMP	https://site.financialmodelingprep.com/terms-of-service	Without a specific agreement with Financial Modeling Prep, customers are prohibited from showcasing Financial Modeling Prep Services or Data on platforms, irrespective of whether such usage is complimentary or paid, and whether it pertains to internal or external organizational purposes.	99	Unknown	Unknown	Unknown
iex cloud	https://iexcloud.io/terms/ https://iexcloud.io/documentation/faqs.html	In general, displaying IEX Cloud data is permitted per the guidelines set in the Third Party Requirements, Attribution, and Terms articles. On a paid plan, you can use IEX Cloud data for commercial use, including displaying publicly to users.	100	Yes	Yes	Unknown
Finazon	https://finazon.io/terms	Professional use doesn't categorize us... but do we want to take the chance?	2000	Unknown	Unknown	No

Joonis 15. *Finantsinfo API-de võrdlus*

Lisa 5 – Domeeninimede valik

Domeeninime määramiseks loodi tabelarvutustarkvaras nimekiri potentsiaalsetest nimedest ning demokraatlikul hääletusel valiti selleks investingedge.pro.

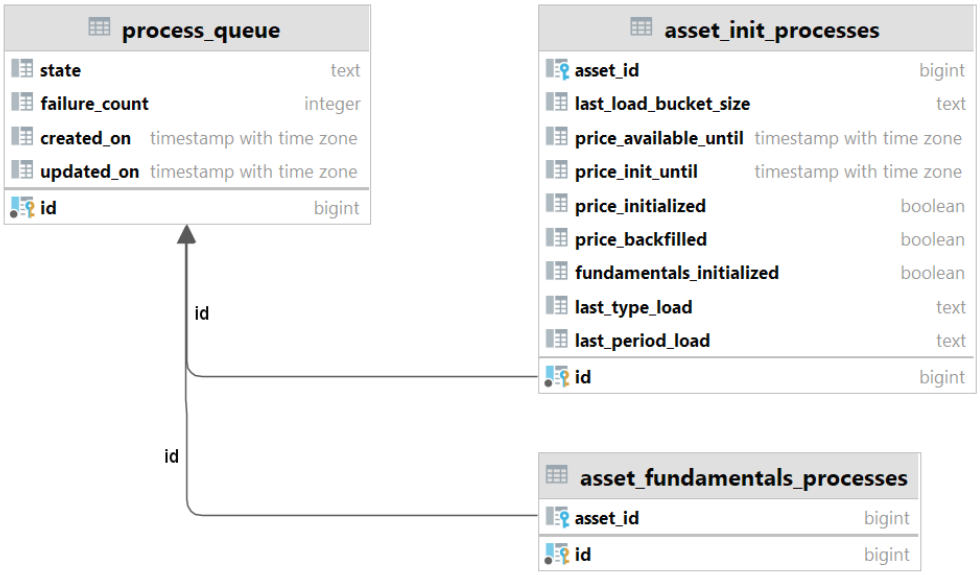
Name	Price (1y)	Price (2y)	Notes	Rating (Janar)	Rating (Sven)	Rating (Karl)	Rating (Client)	Composite score
investingedge.pro	2.74€	21.08€	How much of a market edge does our platform have?					19
intelligentinvesting.pro	2.74€	21.08€	I like it! Straight to the point, no fuss.					18
aphelion.finance	11.00€	58.72€	A good short "brand" name and theme, super! Point in the orbit where it is farthest from the Sun. I.e. uncovering hidden / unknown investments....					18
intelligentinvestor.me	4.57€	34.85€	A good choice as well.					18
marketsense.pro	2.74€	21.08€	Good, short, understandable.					18
helpr.investments	11.00€	103.71€	Kind of pricey...					17
financenavigator.pro	2.74€	20.18€	Slogan: "Charting Your Path to Financial Success"					17
helper.finance	11.00€	58.72€	Good, short, understandable.					16
investingwise.pro	2.74€	21.08€						16
investify.markets	5.49€	18.33€						15
printingpress.finance	11.00€	58.72€						15
intelligentinvestor.io	41.30€	82.59€	This, or intelligentinvestor.me ?					15
intelligentinvest.ing	12.01€	24.01€	Super!					15
smartinvestedge.com	7.33€	12.84€						14
investmentz.io	41.30€	82.59€						14
printingpressportfolio.com	7.33€	18.88€						14
companion.markets	5.49€	18.33€						14
investingcompanion.io	41.30€	82.59€						14
brrr.markets	5.49€	18.33€	"Money printer go brrr", funny, but do people understand it?					13
brrr.financial	11.00€	55.97€	Kind of funny, but do people understand it?					13
printergobrrr.money	8.24€	37.61€						13
aphelioninvestments.com	7.33€	18.88€						13
printingpressinvestments.com	7.33€	18.88€	As in "money printer"					13
aphelionfinance.pro	2.74€	21.08€						13
printing-press-investments.com	7.33€	18.88€						13
investify.today	2.74€	21.08€						13
market-edge.io	41.30€	82.59€						13

Joonis 16. *Domeeninimede valikud*

Lisa 6 – Filtreerimiseks loodud failide liidesed

```
public interface CriteriaConfig {

    Map<FieldValueType, List<ConfigField>> getSearchFields();

    Map<String, Map.Entry<ConfigField, FieldValueType>>
    getValidWhereFilterFields();

    Map<String, Map.Entry<ConfigField, FieldValueType>>
    getValidHavingFilterFields();

    Map<String, ConfigField> getAllValidSortFields();
}

public interface CustomSelectConfig {
    List<ConfigField> getSelectFields();

    List<ConfigField> getGroupBy();
}
```

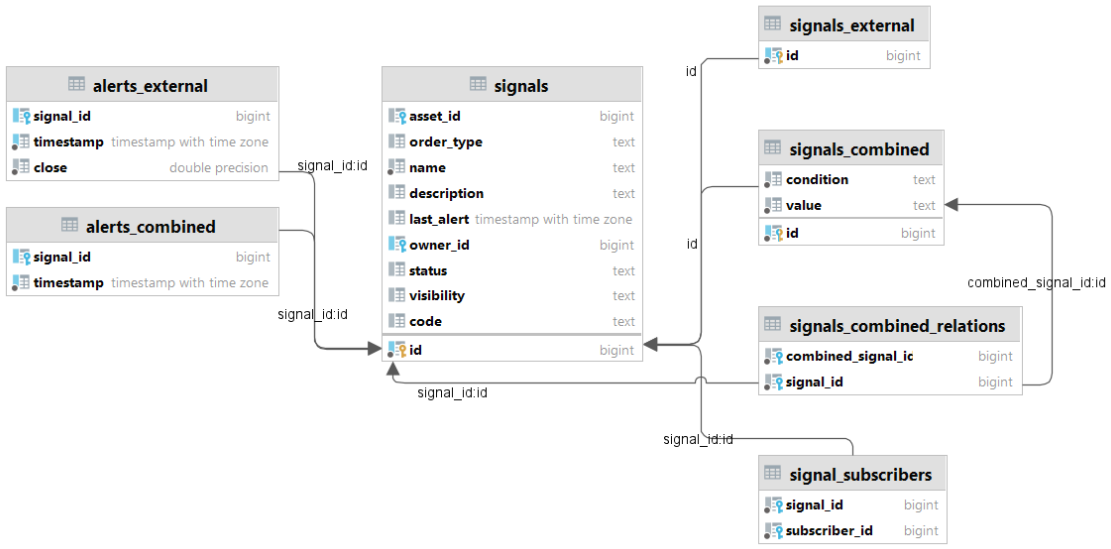
Lisa 7 – Andmebaasi tabelite struktuur



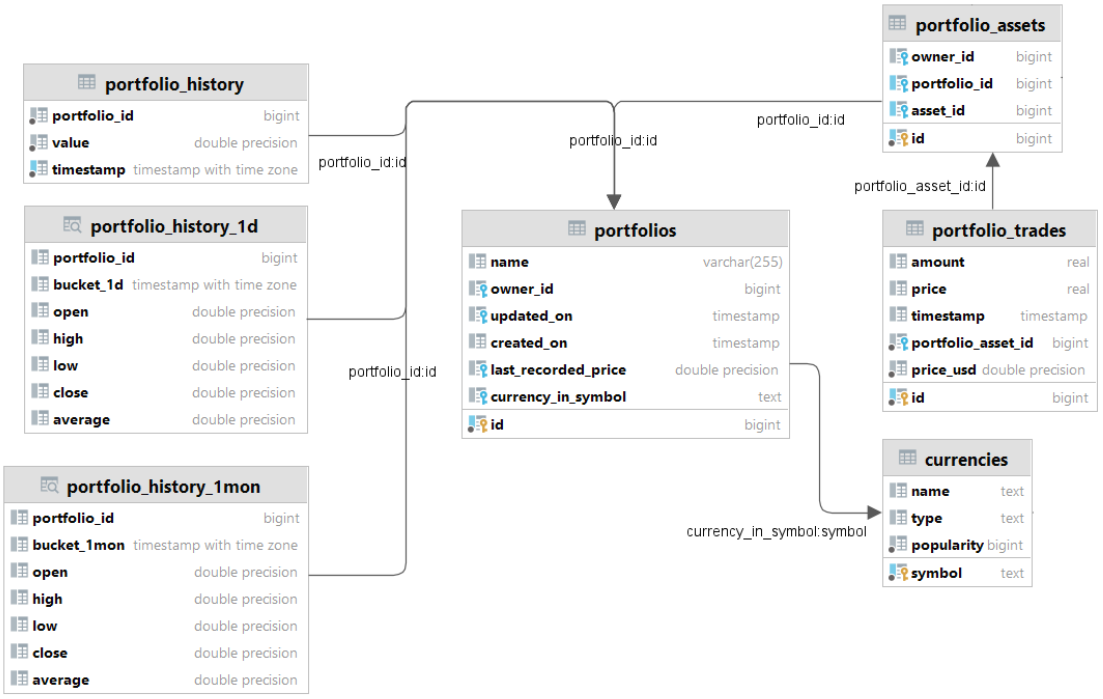
Joonis 17. *Process-queue tabelid*

Joonis 18. *Fundamentaalnäitajate tabelid*

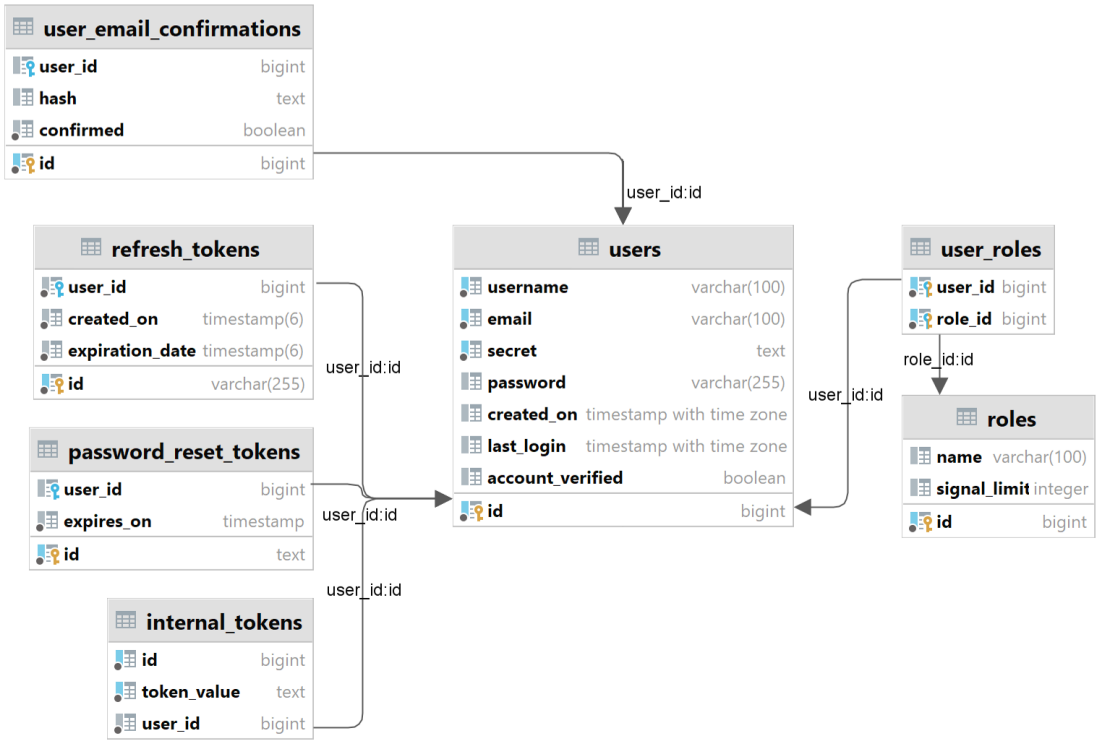
Joonis 19. *Varade, hinnainfo tabelid*



Joonis 20. *Ostu- ja müügisignaalide tabel*



Joonis 21. *Portfellidega seotud tabelid*



Joonis 22. *Kasutajate ja autentimisega seotud tabelid*

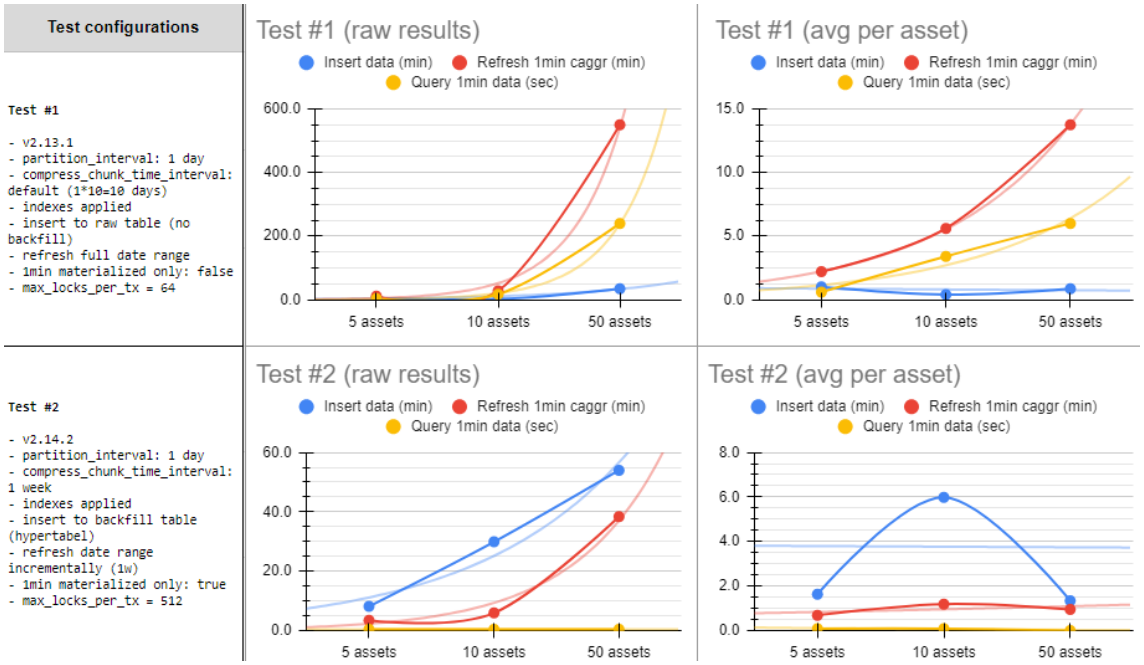
Lisa 8 – TimescaleDB hüpertabelite koormustestid

Testide käigus selgus, et kuna sisestati mitu vara korraga, oli mõistlik uued andmed kõigepealt ajutiselt sisestada eraldi tabelisse (*time_series_backfill*) ning siis ühe korraga liigutada kõik andmed ümber hüpertabelisse (*time_series_raw*).

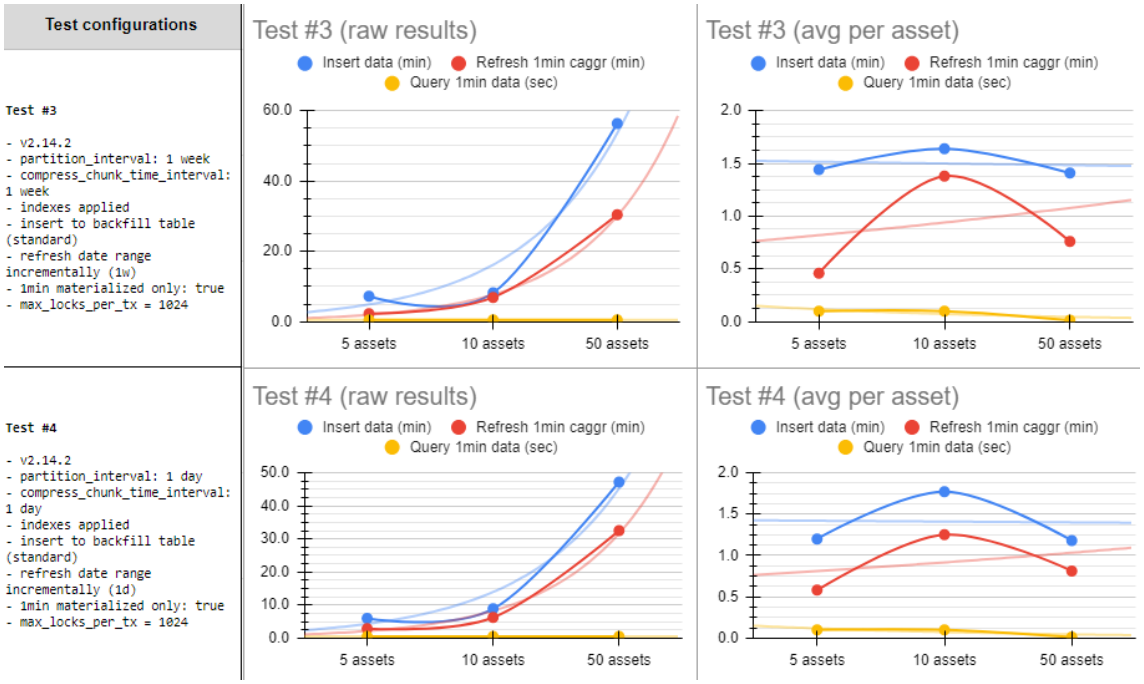
Riistvarapiirangute tõttu pidime tegema järeleandmisi – jätkates küll 1-minutiliste andmete pärimist API-st, kuid esimene *continuous aggregate* vaade loodi viieminutilise intervalli põhjal. Suuremate intervallidega kulub vaadete uuendamisele vähem aega. Kui tulevikus rakendus kasvab ja riistvara võimekus paraneb, saab seda kompromissi muuta.

General info			*Time taken for SQL call*			*Average time taken per asset*		
Test #	Total assets	Assets inserted	Inserting data (minutes)	Refreshing 1min caggr (minutes)	Querying 1min data (seconds)	Inserting data (minutes)	Refreshing 1min caggr (minutes)	Querying 1min data (seconds)
1	5 assets	5	5.0	11.0	3.0	1.0	2.2	0.6
	10 assets	5	2.0	28.0	17.0	0.4	5.6	3.4
	50 assets	40	35.0	550.0	240.0	0.9	13.8	6.0
2	5 assets	5	8.2	3.5	0.5	1.6	0.7	0.1
	10 assets	5	29.9	6.0	0.5	6.0	1.2	0.1
	50 assets	40	54.0	38.4	0.5	1.4	1.0	0.0
	55 assets	5	2.0	46.7	0.5	0.4	9.3	0.1
	60 assets	5	32.0	45.0	0.5	6.4	9.0	0.1
3	5 assets	5	7.2	2.3	0.5	1.4	0.5	0.1
	10 assets	5	8.2	6.9	0.5	1.6	1.4	0.1
	50 assets	40	56.3	30.4	0.5	1.4	0.8	0.0
4	5 assets	5	6.0	2.9	0.5	1.2	0.6	0.1
	10 assets	5	8.9	6.3	0.5	1.8	1.3	0.1
	50 assets	40	47.2	32.5	0.5	1.2	0.8	0.0

Joonis 23. *Koormustestide tulemuste andmestik*



Joonis 24. *Koormustestide konfiguratsioon ning graafilised tulemused (esimene ja teine test)*

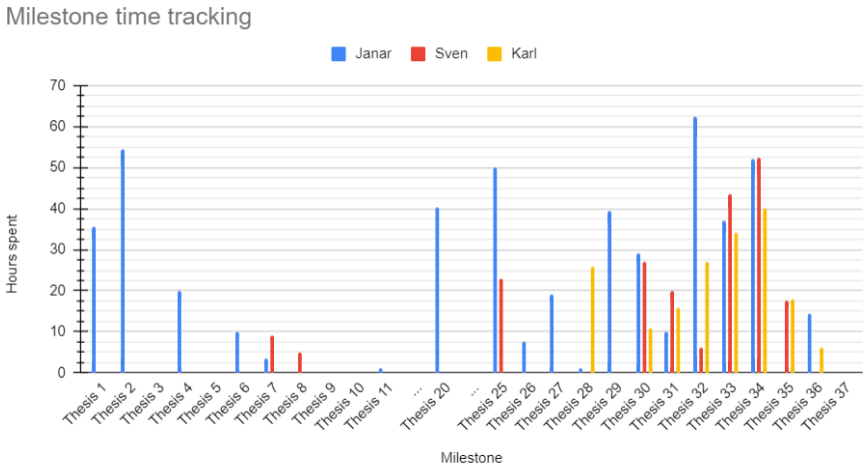


Joonis 25. *Koormustestide konfiguratsioon ning graafilised tulemused (kolmas ja neljas test)*

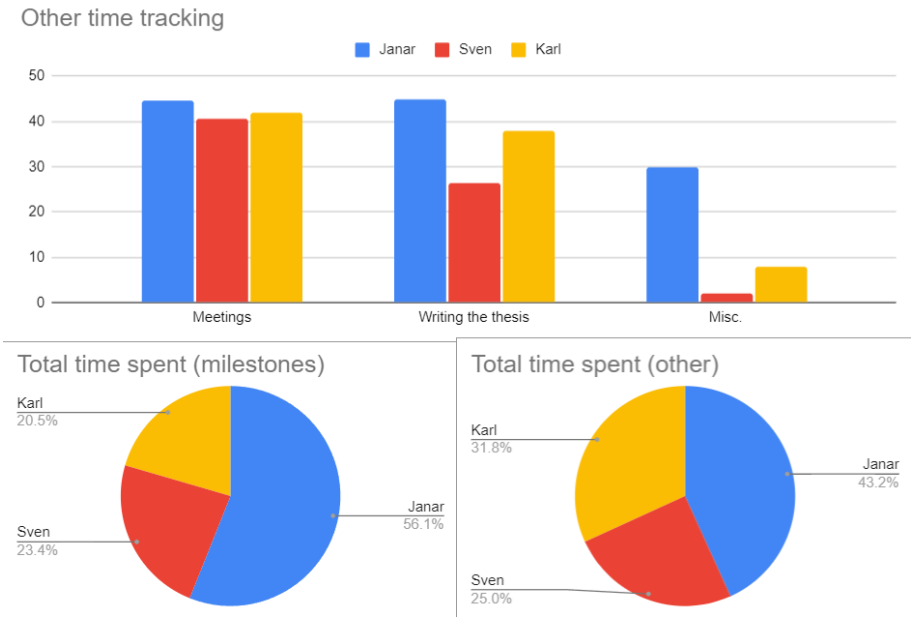
Lisa 9 – Meeskonnaliikmete ajakulu

Tiimiliikmete ajakulu oli järgnev (27. mai 2024 seisuga):

- Janar - 606h 15min
- Sven - 272h 30min
- Karl Erik - 266h



Joonis 26. *Meeskonnaliikmete ajakulu sprintide raames*



Joonis 27. *Meeskonnaliikmete ajakulu kategoriseeritult*