TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Roma Imran Tariq
213315IADB

Brauseri pistikprogramm veebilehega seotud informatsiooni jagamiseks

Bakalaureusetöö

Juhendaja: Jaanus Pöial
PhD

Tallinn 2025

Autorideklaratsioon

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

Autor: Roma Imran Tariq

01.12.2024

Annotatsioon

Käesoleva lõputöö eesmärgiks on luua brauseri pistikprogramm, mis võimaldab jagada informatsiooni mistahes veebilehe kohta. Rakendus kombineerib tähtsamad funktsionaalsused ehk võimaldab kasutajatel kommenteerida, suhelda reaalajas ja raporteerida tõrgete esinemisi. Valmiv rakendus ei ole lõplik ning seetõttu on selle väljundiks minimaalne töötav toode.

Valmiv lahendus koosneb esi- ja tagarakendusest. Esirakenduseks on Chromiumi-põhistes brauserites töötav pistikprogramm, mille teostamiseks kasutatakse TypeScript programmeerimiskeelt, React teeki ja Tailwind raamistikku. Tagarakenduse teostamiseks kasutatakse C# programmeerimiskeelt koos ASP.NET Core raamistikuga. Andmeid hoitakse relatsioonilises andmebaasis, mille juhtimissüsteemiks on PostgreSQL.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 37 leheküljel, 7 peatükki, 18 joonist, 3 tabelit.

Abstract

Browser Extension for Sharing Information about a Webpage

The aim of this thesis is to create a browser extension that allows users to share information about any webpage. The application combines the most important features – it allows users to comment, chat in real time and report connectivity issues. The application is not complete and therefore the result will be minimal viable product.

The thesis contains information about analysing the problem, setting requirements, choosing correct technologies, implementing the solution, testing it and thinking of further development.

The application is divided into frontend and backend. Frontend is a Chromium based browser extension that is written with React in TypeScript. Tailwind framework is used for the styling of it. The application ought to provide simple and comfortable user experience. Backend is written in C# with ASP.NET Core. Selected technologies allow backend to be well maintainable and secure. Backend uses PostgreSQL relational database for data storage. Client and server communicate mainly by using REST API over HTTP. However, for real-time conversation WebSocket API with SignalR implementation is used instead.

The thesis is in Estonian and contains 37 pages of text, 7 chapters, 18 figures, 3 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface,* rakendusliides
ASP.NET Core	Raamistik veebirakenduste loomiseks
BLL	*Business Logic Layer,* äriloogika kiht
CSS	*Cascading Style Sheets,* keel veebilehtede kujundamiseks
DAL	*Data Access Layer,* andmetele juurdepääsu kiht
EF	*Entity Framework,* .NET põhine ORM
HTTP	*Hypertext Transfer Protocol,* protokoll teabe edastamiseks
HTTPS	*Hypertext Transfer Protocol Secure*, HTTP krüpteeritud versioon
IP-aadress	arvutivõrgus oleva seadme identifikaator
.NET	Raamistik rakenduste loomiseks
ORM	*Object-relational mapping,* tehnika relatsioonilise andmebaasiga suhtlemiseks kasutades rakenduse objekte
OTP	*One-time password,* ühekordne parool
REST	*Representational State Transfer*, tarkvara arhitektuuri tava
SQL	*Structured Query Language,* keel relatsioonilise andmebaasiga suhtlemiseks
SSO	*Single sign-on,* identifitseerimismeetod, mis võimaldab ühe kontoga siseneda erinevatesse süsteemidesse
URL	*Uniform Resource Locator*, üldine infoallika asukohamääraja
UUID	*Universally Unique Identifier,* universaalselt unikaalne identifikaator

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Veebilehtede külastamisest on saanud inimeste igapäevaelu lahutamatu osa. See on mugav ja tõhus viis informatsiooni tarbimiseks. Internet leiab tänapäeval küll tunduvalt laiemat kasutust, kuid algselt see just teabe jagamiseks loodigi.

Populaarsemaid rakendusi kasutades, kus ka kasutajad saavad teavet jagada, on näha, et mitmesuunaline suhtlus võib kasutamiskogemust parandada. See annab kasutajatele võimaluse teistega suhelda, arvamust avaldada ja küsimusi esitada. Mis omakorda muudab tarbitava teabe mitmekesisemaks ja interaktiivsemaks.

Teiste kasutajate poolt jagatav teave saab erineval moel abiks olla. Otsuse langetamisel võetakse tihti arvesse teiste hinnangut ja tagasisidet. Olgu selleks toote ostmisel valiku tegemine või veebilehe sisu usaldusväärsuse hindamine. Puudulikku, vigast või tahtlikult valelikku teavet on ka raskem levitada, kui kasutajatel on võimalus sellest märku anda ja parandada.

Tuues välja vaid mõned kasutusjuhtumid on juba näha kuivõrd oluline on kasutajate panus teabe kvaliteedi jaoks. Teisalt näitab see seda, et veebilehtedel, kus saab infot ainult tarbida, võib kasutajakogemus märgatavalt kannatada saada.

1.1 Eesmärk ja ülesande püstitus

Lõputöö eesmärgiks on luua universaalne lahendus, mille abil saab teavet jagada mistahes veebilehega seoses ning muuta seeläbi veebi kasutamine veelgi paremaks.

Eesmärgi saavutamiseks tuleb leida erinevaid võimalusi, kuidas kasutajad saaksid üksteisega veebilehega seostuvat teavet jagada ja tarbida. Analüüsida olemasolevaid lahendusi ja nende puudusi. Panna paika loodava rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded. Valida sobivad tehnoloogiad lahenduse loomiseks ning teostada selle põhjal rakendus.

1.2 Metoodika

Ülesande lahenduseni jõudmiseks on jagatud töö järgnevateks sammudeks:

1. Uurida probleemi.

   Uurida probleemi aktuaalsust ja leida, kas see vajab üldse lahendamist.

2. Uurida teabe jagamise viise ja leida olulisemad funktsionaalsused.

   Võrrelda erinevaid võimalusi teabe edastamiseks. Uurida populaarsemate lahenduste pealt, milliseid võiks olla neile vastavad funktsionaalsused. Analüüsida funktsionaalsusi ning valida neist olulisemad.

3. Uurida olemasolevaid lahendusi ja analüüsida nende puudusi.

   Leida alternatiivsed lahendused, analüüsida neid ja leida nende puudused. Puuduste välja selgitamine aitab õppida teiste vigadest ja luua selle võrra parem lahendus.

4. Uurida ja leida lahendusele sobiv sihtgrupp.

   Parema tulemuse saamiseks leitakse kasutajagrupp, kellele loodav lahendus võiks huvi pakkuda.

5. Uurida ja panna paika lahenduse nõuded.

   Tulemuse täpsustamiseks pannakse paika funktsionaalsed ja mittefunktsionaalsed nõuded.

6. Võrrelda ja analüüsida erinevaid tehnoloogiaid.

   Rakendus koosneb erinevatest osadest. Enne nende teostamiseni asumist tuleb erinevaid tehnoloogiaid uurida ning võrrelda. Nii-viisi leitakse sobivad töövahendid esirakenduse, tagarakenduse ja andmebaasi jaoks.

7. Teostada rakendus.

Rakenduse teostamiseks luuakse minimaalne töötav toode, mis võimaldab kasutada põhilisi funktsioone. Lahenduse väljund on mahukam kui prototüüp, kuid siiski sisaldab puudusi ja pole lõplik lahendus.

8. Testida lahendust.

   Kontrollida loodud rakenduse vastavust nõuetele. Viia läbi kasutajatestimine ning analüüsida kasutajate poolt saadud tagasisidet.

Lahenduseni jõudmiseks kasutatakse erinevaid meetodeid.

1.3 Struktuur

Töö on jaotatud seitsmeks peatükiks. Esimeses peatükis tutvustatakse lühidalt tööd. Probleemi analüüsi peatükis leitakse olulisemad funktsionaalsused ning olemasolevate lahenduste puudused. Lahenduse nõuded peatükis pannakse paika sihtgrupp, funktsionaalsed ja mittefunktsionaalsed nõuded, lahenduse skoop ning lahendatavad kasutajalood. Tehnoloogiate analüüsi peatükis leitakse sobilikud vahendid rakenduse loomiseks. Lahenduse teostus peatükis tuuakse välja olulisemad osad rakenduse arendamisest. Tulemused peatükis hinnatakse valminud lahendust, tuuakse välja edasiarenduse võimalused ning viiakse läbi kasutajatestimine. Viimases peatükis võetakse töö lühidalt kokku.

2 Probleemi analüüs

Veebi kasutamine on väga paljude jaoks osa igapäevaelust. Käesoleva aasta seisuga on veebis saadaval enam kui miljard erinevat veebisaiti [1]. Otsingumootorit Google arvesse võtmata, võimaldavad enim külastatud veebisaidid: YouTube, Facebook ja Instagram kasutajatel üksteisega teavet jagada [2]. Tegemist on sotsiaalmeedia platvormidega, kus kasutajatevaheline suhtlus on lausa põhifunktsiooniks. Võib järeldada, et sarnane vajadus võib kasutajatel olla ka veebisaitidel, millel see põhifunktsiooniks ei ole. Tegelikult nii ongi. Klassikaliste meediakanalite põhifunktsiooniks on informatsiooni jagamine kasutajatele, samas on tihti veebiväljaannetes võimaldatud kasutajatel ka üksteisega muljeid jagada.

Mitmetel veebisaitidel on kasutajatel võimalus teineteisega teavet jagada. Samas on ka palju veebisaite, kus seda võimalust ei ole. Lisaks koosneb veebisait enamasti mitmest veebilehest. See tähendab, et isegi kui veebisaidil on võimalus teavet jagada, ei ole garanteeritud, et see funktsionaalsus ei puuduks mõnelt lehelt, kus see olla võiks.

2.1 Olulisemad funktsionaalsused

Tabelis 1 on toodud välja peamised suhtlusmeetodid ja hinnatud nende sobivust antud lahenduse puhul kõige olulisemaks osutuvate kriteeriumite vastu. Võrdlusest selgub, et antud olukorras on parimaks suhtlusmeetodiks tekstipõhine lahendus. Teksti kasutamine on mugav, sobilik reaalajas suhtluseks olenemata kasutajate kogusest, madala andmemahu kasutusega ja lihtsasti modereeritav märksõnade filtreerimisega.

Tabel 1. Suhtlusmeetodite võrdlus.

Kriteerium	**Tekst**	**Heli**	**Video**
Ligipääsetavus	Hea	Rahuldav	Halb
Reaalajas suhtlus	Hea	Hea	Rahuldav
Andmemaht	Hea	Rahuldav	Halb
Moderatsioon	Hea	Halb	Halb

Lihtne ja tõhus viis oluliste funktsionaalsuste leidmiseks on antud juhul vaadata, milliseid tekstipõhiseid suhtlusviise kasutatakse enim külastatud sotsiaalmeedia veebisaitidel YouTube, Facebook ja Instagram [2].

Tabelis 2 tehtud võrdlusest selgub, et ühised funktsionaalsused on kommenteerimine, reaalajas suhtlemine, kommentaaridele vastamine ja reageerimine ning postitused. Lahenduses kombineeritakse postitamise ja kommenteerimise funktsioonid, sest sisuliselt on postituse rollis veebileht, mille kohta teavet jagatakse.

Tabel 2. Tekstipõhised suhtlusviisid enim külastatud veebisaitidel.

Funktsionaalsus	**YouTube**	**Facebook**	**Instagram**
Kommenteerimine	Jah	Jah	Jah
Reaalajas suhtlemine	Jah	Jah	Jah
Kommentaaridele vastamine	Jah	Jah	Jah
Kommentaaridele reageerimine	Jah	Jah	Jah
Privaatsõnumid	Ei	Jah	Jah
Postitused	Jah	Jah	Jah
Hinnangud	Ei	Jah	Ei

Võib esineda olukordi, kus veebisait ei ole kättesaadav. Taolisel juhul ei ole ilmselgelt võimalik seda teavet saada veebisaidilt endalt. Lihtne meetod välja selgitamaks, kas probleem on kliendi- või serveripoolne, on sellest teistega arutamine. Kui probleem esineb mitmel kasutajal on vägagi tõenäoline, et veebisait ei olegi kättesaadav ja kasutaja ei pea omalt poolt midagi tegema. Antud teemal saab teistega arutada näiteks kommentaaride kaudu. Küll aga on tegemist võrdlemisi spetsiifilise juhtumiga ja kommentaaride vahelt otsimine ei pruugi olla lihtsaim ettevõtmine. Selle jaoks lisandub lahendusele funktsionaalsus, mis võimaldab kasutajatel veebilehega seotud tõrgetest teavitada ja selle kohta statistikat näha.

2.2 Olemasolevad lahendused

Lõputöös käsitletavale probleemile on mitmeid sarnaseid lahendusi. Antud peatükis käsitletakse neist osasid ja tuuakse välja nende puudused:

- Google Sidewiki – aastal 2009 Google poolt loodud brauseri pistikprogramm, mis võimaldas kasutajatel kommentaaride kaudu jagada mõtteid mistahes veebilehe kohta [3]. Programm pandi kinni aastal 2011 [4].

- Hypothesis – aastal 2011 loodud brauseri pistikprorgamm, mis võimaldab kasutajatel annoteerida mistahes veebilehel olevat teksti [5]. Rakenduse põhivaates on korraga kuvatud palju funktsioone [6], mis teeb selle autori arvates uutele kasutajatele raskesti hõlmatavaks.

- DownDetector – aastal 2012 loodud veebisait, mis kuvab informatsiooni võimalikest tõrgetest vaid populaarsemate veebiteenuste kohta [7].

- Isitdownrightnow – aastal 2012 loodud veebisait, mis kuvab informatsiooni võimalikest tõrgetest mistahes veebilehe kohta. Puudub teineteisega suhtlemise võimalus. [8]

- Disqus – aastal 2007 loodud veebirakendus [9], mis võimaldab kasutajatel üksteisega suhelda kommentaaride kaudu veebilehtedel, kus selle haldaja on lahenduse enda veebilehele lisanud [10].

Lõputöös käsitletavale lahendusele on palju sarnaseid alternatiive, kuid neis kõigis esineb puudusi. Ei ole ühtegi rakendust, mis võimaldaks kasutada mistahes veebilehe kohta kolme olulisemat funktsionaalsust:

- kommenteerimine, sealhulgas kommentaarile vastamine ja reagreerimine;

- reaalajas suhtlemine;

- tõrgetest teatamine ja selle statistika nägemine.

2.3 Kasutusvõimalused

Loodaval rakendusel võib olla palju erinevaid kasutusvõimalusi. Rakendus kombineerib mitut erinevat funktsionaalsust ehk rakendust võib kasutada ka siis, kui ollakse huvitatud vaid ühest. Järgnevalt tuuakse välja funktsionaalsused koos kasutusvõimalustega, mis autori arvates võiks osutuda enim kasutatud juhtudeks.

Esiteks saab kasutaja teavitada sellest, kui veebileht ei ole kättesaadav ja näha kas sama probleem esineb ka teistel. See aitab kasutajal aru saada sellest, kas viga on kliendi- või serveripoolne. Kui probleem esineb ka teistel ehk viga on serveripoolne, ei ole kasutajal vaja aega kulutada lahenduse otsimiseks. Funktsionaalsus võib osutuda kasulikuks ka veebilehe haldajale, sest kasutajatel tekib parem arusaam vea olemusest ja pole vaja murega haldaja poole pöörduda. Veebilehe kättesaamatuse statistika aitab haldajal näha probleemi olemasolu ja võib-olla selle põhjal ka mingeid järeldusi teha.

Teiseks saab kasutaja suhelda teistega reaalajas. Taoline olukord võib tekkida juhul, kui veebilehel voogedastatakse mõnda sündmust ja soovitakse teistega sellest diskuteerida. Populaarsemate sündmuste edastamisel võib korraga olla miljoneid aktiivseid kasutajaid.

Kolmandaks saavad kasutajad üksteisega arutada veebilehega seotut kommentaaride kaudu. Veebilehel võib esineda puudulik või lausa valelik info, mida kasutajad saavad täiendada ja parandada. Veebilehe sisu võib olla maksumüüri taga, vajada sisselogimist või olla liigutatud muule aadressile. Taolise olukorra puhul võib juhtuda, et keegi on sisust kokkuvõtte teinud ja seda teistega jaganud. Joonisel 10 on kuvatud näide, kus juhendile viitav hüperlink tagastab veateate ning loodava rakenduse abil leitakse otsitav teave.

Kasutusvõimalusi on mitmeid ning see, kuidas rakendust kasutatakse sõltub kasutajast endast. Vägagi tõenäoline, et esineb ka kasutusviise, mille peale pole isegi autor tulnud.

3 Lahenduse nõuded

Antud peatükis analüüsitakse valmivat lahendust ja pannakse paika nõuded. Lahenduse nõuete paikapanek aitab paremini mõista, milline valmiv rakendus olema peaks.

3.1 Sihtgrupp

Rakenduse toimimiseks on oluline, et kasutajad ei piirduks ainult lugemisega, vaid postitaksid ka ise. Seetõttu on hea, kui kasutajateks on aktiivsed veebilehtede külastajad, kes soovivad sageli teistega teavet jagada.

Kasutajad peaksid üksteisest arusaamiseks kõnelema ühist keelt ehk oskama inglise keelt, sest tegemist on Internetis kõige enam kasutatud keelega [11].

On raske ette kujutada, et keegi kasutaks vabatahtlikult rakendust, mille kasutamine on ebamugav. Mugav viis veebilehega seotud teabe jagamiseks on, kui mõlemad rakendused on korraga nähtaval. See on võimalik avades mõlemad rakendused eraldi akendes ja paigutades need kõrvuti või kasutades brauseri pistikprogrammi (joonis 10).

Brauseri pistikprogrammi kasutades kulub lahenduse avamiseks minimaalselt üks samm:

1. pistikprogrammi ikoonile vajutamine.

Tavalist veebirakendust kasutades kulub minimaalselt neli sammu:

1. avatud veebilehe aadressiriba aktiivseks muutmine;
2. aadressiribalt veebilehe URL-i kopeerimine;
3. veebirakenduse avamine uues aknas;
4. kopeeritud URL-i pasteerimine.

Pistikprogrammi avamine on antud juhul märgatavalt lihtsam kui tavalise veebirakenduse avamine. Seetõttu on tähtis, et kasutajad oleksid võimelised paigaldama brauseri pistikprogrammi.

Kokkuvõtvalt on sihtgrupiks aktiivsed veebilehtede külastajad, kes on huvitatud teistega teabe jagamisest, omavad baastasemel inglise keele oskust ja on võimelised paigaldama brauseri pistikprogrammi.

Sihtgrupp on võrdlemisi lai, et potentsiaalseid kasutajaid oleks võimalikult palju. See on oluline, sest mida rohkem kasutajaid lahendusel on, seda tõenäolisem, et veebilehe kohta abi leitakse. Enam kui 270 000 kasutajaga Hypothesis pistikprogramm on olemasolevatest lahendustest sarnaseim, kuid see on mõeldud eelkõige annoteerimiseks [12]. Disqus on teine alternatiivne lahendus, millel on ligi 35 miljonit kasutajat, kuid võimaldab kommenteerida vaid valitud veebilehtedel [13]. Kasutajate hulga suurus näitab, et kommenteerimise võimalust veebilehtedel siiski kasutatakse. Seda toetab ka küsitlus, mille järgi on 77,9% vastanutest kommentaare lugenud ja 55,0% ise postitanud [14]. Väljatoodud lahenduste sihtgrupid on kitsamad kui loodava lahenduse sihtgrupp ehk võimalikke kasutajaid on loodaval rakendusel rohkem. Ilmselt leidub väljatoodud lahenduste kasutajate seas neid, kes oleksid huvitatud potentsiaalselt tõhusamast lahendusest, mis oleks mugav, üldisem, võimaldaks kasutada erinevaid viise informatsiooni jagamiseks ja võimaldaks seda teha mistahes veebilehe kohta.

3.2 Funktsionaalsed nõuded

Funktsionaalsed nõuded kirjeldavad funktsionaalsusi, mida käsitletav rakendus peab olema võimeline tegema [15]. Nõuete kirjeldamisel on lähtutud peatükis 2.1 leitud olulisematest funktsionaalsustest.

Järgneb loetelu rakenduse funktsionaalsetest nõuetest:

- rakendus peab võimaldama kasutajatel kommenteerida, sealhulgas kommentaarile vastata ja reageerida ning teha seda mistahes veebilehe kohta;
- rakendus peab võimaldama kasutajatel reaalajas suhelda ning teha seda mistahes veebilehe kohta;
- rakendus peab võimaldama kasutajatel teavitada mistahes veebilehega seotud tõrgete esinemisest ja selle kohta statistika nägemist.

3.3 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded seavad piirangud rakenduse disainile ja teostusele [16]. Nõuete kirjeldamisel on lähtutud peatükis 3.1 defineeritud sihtgrupist.

Järgneb loetelu rakenduse mittefunktsionaalsetest nõuetest:

- esirakenduseks on brauseri pistikprogramm, mis peab olema kättesaadav võimalikult paljudele ehk olema kasutatav Google Chrome brauseris [17];
- tagarakendus peab võimaldama vähese vaevaga alternatiivse esirakenduse loomist;
- rakendus peab võimaldama edasiarendust.

3.4 Skoop

Lõputöös käsitletavat rakendust on võimalik sisuliselt lõputult edasi arendada ja funktsionaalsusi lisada. Selleks, et rakendus õigeks ajaks valmis saaks, tuleb määrata skoop.

Esiteks peab rakendus täitma kõik funktsionaalsed ja mittefunktsionaalsed nõuded esitatud vastavalt peatükkides 3.2 ja 3.3.

Teiseks kuulub enamasti rakenduse juurde ka kasutajate ning tuluteenimise strateegia leidmine. Eriti taolise rakenduse puhul, mis on üles ehitatud kasutajatevahelise suhtluse ümber. Ajapiirangu tõttu jääb see antud töö skoobist välja.

Kolmandaks kuulub rakenduse juurde enamasti põhjalik testimine, seehulgas ka turvalisuse kontrollimine. Antud lahenduse puhul ei käsitleta tundlikke andmeid ning seetõttu jääb see skoobist välja. Ajapiirangu puudumisel loodaks rakendusele minimaalselt automaattestid, mis aitaksid ennetavalt vigu leida.

3.5 Kasutajalood

Hea tarkvara koosneb kihtidest. Kihtideks jaotamiseks on kaks viisi: vertikaalselt ja horisontaalselt. Horisontaalselt jaotades võetakse korraga ette üks kiht ja rakendatakse kõik vajalikud toimingud. See tähendab, et rakendust saab alles siis kasutada, kui kõik

funktsionaalsused on valmis. Vertikaalselt jaotades võetakse ette korraga üks kasutajalugu ja teostatakse igast kihist selle kohta käiv osa. Vertikaalselt jaotades saab peale kasutajaloo valmimist lisatud funktsionaalsust koheselt proovida ja vajadusel varakult parandusi teha. [18]

Toetamaks rakenduse vertikaalset kihitamist, moodustatakse peatükis 3.2 paikapandud funktsionaalsetest nõuetest lähtuvalt kasutajalood.

Teostades tabelis 3 välja toodud kasutajalood prioriteedi järgi, alustades kõrgematest ja liikudes madalamate juurde, valmivad rakenduse olulisemad osad esimesena. Nii valmib tähtaja saabumiseks võimalikult kasutuskõlblik rakendus ja ei ole probleemiks kui mõni vähemtähtsam kasutajalugu jääb teostamata.

Tabel 3. Kasutajalood.

Prioriteet	**Kasutajarollis <roll>**	**soovin <tegevus>,**	**et saaksin <eesmärk>.**
Kõrge	kasutaja	luua konto	sisselogida
Kõrge	kasutaja	sisselogida	postitada
Kõrge	autenditud kasutaja	väljalogida	lõpetada sessiooni
Kõrge	autenditud kasutaja	jääda sisselogituks	vältida sisselogimist
Kõrge	kasutaja	näha kommentaare ja vastuseid	leida teavet
Keskmine	kasutaja	sorteerida kommentaare	leida teavet kiiremini
Kõrge	autenditud kasutaja	positada kommentaare ja vastuseid	jagada teavet
Keskmine	kasutaja	näha kommentaari reaktsioone	kiire ülevaate kommentaari adekvaatsusest
Keskmine	autenditud kasutaja	reageerida kommentaarile	aidata tuua paremad kommentaarid esile
Kõrge	kasutaja	näha reaalajas jagatud sõnumeid	näha teiste hetkemõtteid
Kõrge	autenditud kasutaja	saata reaalajas sõnumeid	jagada enda hetkemõtteid
Kõrge	kasutaja	näha statistikat veebilehe tõrgetest valitud ajaperioodil	parema arusaama probleemi kestvusest
Kõrge	autenditud kasutaja	raporteerida veebilehel esinevaid tõrkeid	teistega jagada oma probleemi olemasolu
Keskmine	kasutaja	muuta teemat	kasutada rakendust nii, et selle välimus ei mõjuks avatud veebilehe suhtes häirivalt
Madal	autenditud kasutaja	saada teavitusi	teada, kui keegi vastab või reageerib mu kommentaarile

4 Tehnoloogiate analüüs

Antud peatükis analüüsitakse valmiva rakenduse tehnoloogiaid. Oluline on valida töövahendid nii, et nad sobiksid probleemi lahendamiseks ja lihtsustaksid rakenduse arendusprotsessi. Võimalusel on hea kasutada levinud töövahendeid, sest mida suurem on selle kogukond, seda suurem on abi leidmise tõenäosus.

Kui võimalik, on enamasti mõistlik kasutada mõnda raamistikku või teeki, mis on vajamineva funktsionaalsus juba lahendanud. Nii vähendatakse arendusele kuluvat aega ja tõenäoliselt valmib ka turvalisem ja töökindlam rakendus. Lisaks, raamistikke kasutavad rakendused peavad vastama raamistiku disainile ja põhjustavad seetõttu rakenduses mustrite teket [19]. Teadaolevate mustrite kasutamine muudab rakenduse programmikoodi loetavamaks ja paremini laiendatavaks.

4.1 Esirakenduse tehnoloogiate valik

Mittefunktsionaalsete nõuete järgi peab olema esirakenduseks brauseri pistikprogramm, mis tähendab, et rakendus tuleb jagada esi- ja tagarakenduseks.

Kõige populaarsemad programmeerimiskeeled esirakenduse arendamiseks on JavaScript ja TypeScript [20]. TypeScript on tugevalt tüübitud keel, mis lisab JavaScriptile süntaksi ja aitab seeläbi varakult vigu leida [21]. TypeScripti kasutamine parandab arenduskogemust ja seetõttu osutub see eelistatud keeleks.

Kõige populaarsemad JavaScript raamistikud on React, Angular ja Vue [22]. Olemuselt on need raamistikud sarnased ja otsus langetatakse tihti meeldivuse järgi. Antud töös valitakse React, sest see on kõige populaarsem teek [22].

Esirakendust on vaja ka kujundada. Enim kasutatud CSS raamistikud on Bootstrap ja Tailwind [23]. Tailwindi kasutamisel on mitu eelist. Projekti ehitamisel lisatakse vaid nii vähe CSS-i kui vaja, mis tagab minimaalse CSS faili suuruse [24]. Eeliseks on ka selle ideaalne sobivus Reactis kasutatavate komponentidega ning paindlikkus tagada disaini unikaalsus järjepidevust kaotamata [24].

4.2 Tagarakenduse tehnoloogiate valik

Tagarakenduse tehnoloogiate valikul on raamistikul väga suur ning oluline osa. Valik tehakse raamistiku mitte programmeerimiskeele põhjal. Professionaalsete arendajate hulgas on enim kasutatud veebirakenduste loomiseks mõeldud raamistikud ASP.NET Core, Next.js ja Express [20].

- ASP.NET Core – Microsofti poolt loodud raamistik, millel on suur ja aktiivne kogukond [25]. Lahendus sisaldab kõike vajalikke funktsioone, mis teeb ta äärmiselt mugavaks ja turvaliseks [25]. Tegemist on populaarsete lahenduste seast kõige kiirema veebiraamistikuga [26].

- Next.js – React-i põhine kiire raamistik. Raamistiku fookuseks on aga serveripoolne renderdamine, mis antud juhul ei ole sobilik lahendus. [27]

- Express – Node.js põhine minimalistlik ja kiire raamistik, mis sobib API loomiseks [28]. Minimalistlik lahendus tagab paindlikkuse ja hoiab kokku rakenduse suuruse pealt. Samas tähendab see seda, et funktsionaalsuste kasutamiseks peab kasutama väliseid teeke, mille haldamine võib osutuda ajakulukaks.

Analüüsi põhjal selgub, et Next.js käesoleva rakenduse lahenduseks ei sobi. Erinevus Expressi ja ASP.NET Core vahel ei ole suur ja ei saa öelda, et üks nendest on ainuõige valik. Käesoleva töö juhul on olulisel kohal arenduse kiirus ning seetõttu langetatakse valik ASP.NET Core kasuks. Valitud raamistikku saab kasutada C# programmeerimiskeelega [25].

4.3 Esi- ja tagarakenduse vaheline suhtlus

Esi- ja tagarakenduse vaheliseks suhtluseks sobib hästi HTTP protokolli kasutamine – klient saadab serverile päringu ja server vastab [29]. Krüpteeritud andmevahetusel edastatud teavet ei saa keegi kõrvaline lugeda ning seetõttu on HTTPS-i kasutamine eelistatum variant. Klient- ja serverrakenduse vaheline suhtlus käib läbi API. Selle kasutamiseks on erinevaid viise, aga antud lahenduse puhul on kõige sobivam, oma lihtsuse tõttu, REST API kasutamine [30].

Rakendus peab võimaldama ka reaalajas suhtlemist. Selle lahendamiseks sobib hästi WebSocket API kasutamine, sest see loob kliendi ja serveri vaheliseks suhtluseks kahesuunalise sessiooni [31].

Rakendus peab võimaldama sisselogimist, et kasutajad saaksid postitada. Autentimiseks on mitmeid meetodeid [32]:

- Kasutajanimi ja parool – kõige levinum ja kindlam lahendus [32]. Turvalisuse tagamiseks vajab paroolide korrektset hoiustamist.
- SSO – sisenemine mõne muu rakenduse kontoga. Turvaline lahendus, kuid muu konto puudumisel ei saa autentida.
- OTP – ühekordne parool, mis saadetakse näiteks e-mailile. Turvaline, aga võib-olla mõnele harjumatu. Nõuab sisselogimiseks muu rakenduse avamist.

Väljatoodud variantidest on turvaliseim ühekordse parooli kasutus, kuid klassikalise kasutajanime ja parooli kasutamine on kõige kättesaadavam ja seetõttu osutub see valitud meetodiks. Teostatav rakendus ei käsitle tundlikke andmeid, mistõttu ei ohverdata mugavust marginaalse turvalisuse võidu vastu.

Rakendus peab ka meeles hoidma, et kasutaja on autenditud ja seda informatsiooni kuskil hoiustama. Mark Shafran on oma lõputöös erinevaid meetodeid võrreldnud ja jõudnud järeldusele, et küpsiste kasutamine on eelistatuim valik [33]. Küpsis on kogum andmeid, mis server saadab kliendile, et klient saaks seda edasiste päringutega kaasa saata [34].

4.4 Andmebaasi juhtimissüsteemi valik

Andmebaasi juhtimissüsteemid võib jaotada laialt kaheks: relatsiooniline- ja mitterelatsiooniline andmebaasi juhtimissüsteem [35]. Relatsiooniline andmebaasi juhtimissüsteem sobib antud lahendusega paremini, sest see on optimeeritud struktureeritud ning omavahel seotud andmete kasutamiseks [36].

Enim kasutatud andmebaasi juhtimissüsteem on PostgreSQL [20]. PostgreSQL on tasuta, avatud lähtekoodiga ja võimekas objekt-relatsiooniline andmebaasi juhtimissüsteem [37]. Väljatoodud variandi tugevaks eeliseks on selle hind ja lai kogukond ning seetõttu sobib see hästi loodava rakenduse teostamiseks.

4.5 Tehnoloogiate analüüsi kokkuvõte

Tehnoloogiate valimisel oli tähtis nende populaarsus, et vajadusel oleks võimalik lihtsa vaevaga abi leida.

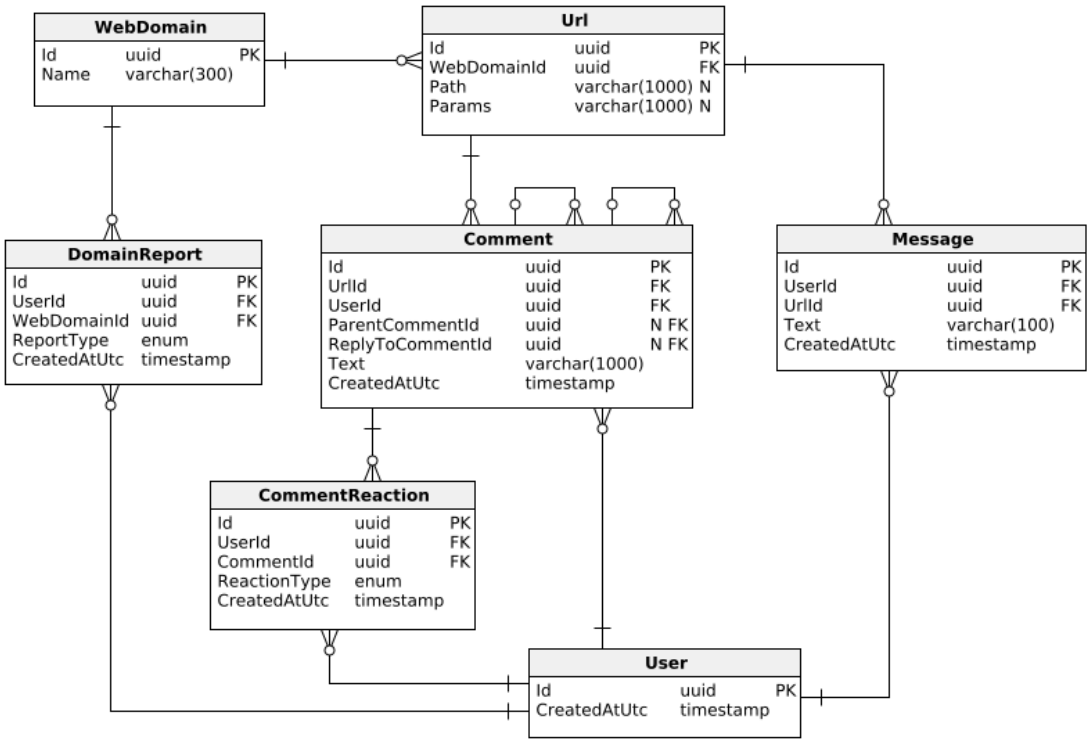
Esirakenduse loomiseks kasutatakse TypeScripti koos React teegi ja Tailwind raamistikuga. Tagarakenduse loomiseks kasutatakse C# programmeerimiskeelt koos ASP.NET Core raamistikuga. Klient- ja serverrakenduse vaheliseks suhtluseks kasutatakse peamiselt REST API-t, aga ka WebSocket API-t reaalajas suhtlemiseks. Rakendustevahelise suhtluse turvalisuse tagamiseks kasutatakse HTTPS protokolli. Sisselogimine on võimalik kasutajanime ning parooli alusel ja teavet sellest hoitakse kliendi brauseris oleva küpsisega. Andmeid hoiustatakse PostgreSQL andmebaasis.

5 Lahenduse teostus

Käesolevas peatükis kirjeldatakse rakenduse loomist. Esi- ja tagarakendusele vastavat lähtekoodi hoitakse GitHub repositooriumites (lisa 2).

5.1 Andmebaasi andmemudel

Rakenduse toimimiseks tuleb kasutatavaid andmeid kuskil hoiustada. Andmeid hoiustatakse andmebaasis. Andmebaas tuleb disainida nii, et see vastaks rakenduse vajadustele. Arvesse tuleb ka võtta, et peatükis 4.4 valiti sobivaks vahendiks objekt-relatsioonilise andmebaasi kasutamist. Olemi-suhete diagrammi loomine annab hea ülevaate planeeritavast andmebaasist ja rakenduse olemusest. Diagrammi kasutamine dokumentatsioonina ei pruugi olla parim idee, sest üldiselt andmebaas võib rakenduse elu jooksul muutuda ja diagrammi pidev uuendamine võib muutuda ajakulukaks. Esialgne andmemudel on näha joonisel 1.



Joonis 1. Esialgne olemi-suhete diagramm.

Tagarakenduses on kasutajate haldamiseks kasutusel ASP.NET Core Identity teek, mis muudab andmemudelit. Joonisel 1 kuvatud diagrammil on teegi poolt lisatud tabelitest välja toodud vaid poolik kasutaja tabel.

Vajab mainimist, et rakenduses on primaarvõtmete andmetüübiks UUID. UUID on 128-bitine universaalselt unikaalne identifikaator, mis eemaldab praktikas võrdsete väärtuste genereerimise võimaluse ehk on kasulik hajussüsteemide puhul [38]. Teisalt on UUID pikem kui *integer* andmetüüpi väli, mis teeb selle kasutamise veidi aeglasemaks ja välimuselt raskemini eristatavaks. Kuna PostgeSQL võimaldab UUID kasutamist [39] ja rakenduse nõudeks oli selle lihtne laiendatavus, valitakse *integer* andmetüübi asemel UUID. Selle kasutamine aitab ära hoida ka identifikaatori äraarvamist, mis tõstab rakenduse turvalisust.

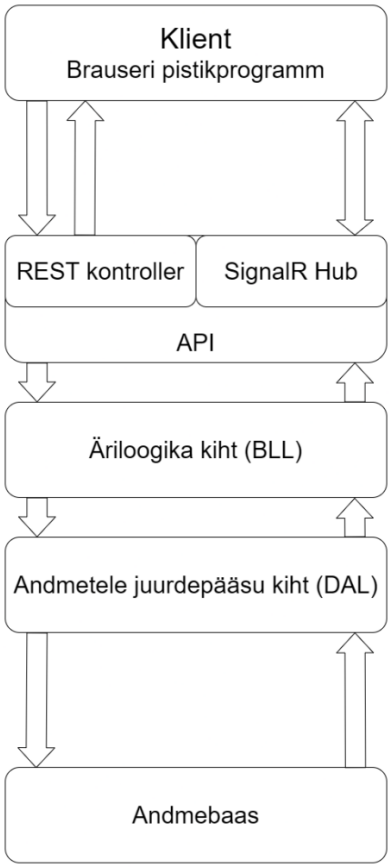
Lisaks on joonisel 1 näha *enum* andmetüübi kasutamist. Tegelikkuses vastab sellele *integer* andmetüüp. *Enumid* asendavad antud juhul teatmik-tüüpi olemite kasutust.

5.2 Tagarakendus

Tagarakenduse loomiseks kasutatakse C# programmeerimiskeelt ja ASP.NET Core raamistikku. Tagarakendus peab ühtlasi suhtlema ka PostgreSQL andmebaasiga.

5.2.1 Arhitektuur

Et rakendus oleks lihtsasti laiendatav, on mõistlik jagada see kihtideks, kus igal kihil on oma ülesanne. See parandab programmikoodi loetavust ja testitavust. Kihtide omavahelist suhtlust illustreerib joonis 2.



Joonis 2. Rakenduse arhitektuur.

Tagarakendus on jaotatud järgmisteks kihtideks:

- API – kiht esirakendusega suhtlemiseks;
- äriloogika kiht – kiht andmete töötlemiseks;
- andmetele juurdepääsu kiht – kiht andmebaasiga suhtlemiseks;
- domeen – kiht andmebaasi defineerimiseks.

5.2.2 Andmebaasiga suhtlemine

Tagarakendus vajab toimimiseks andmebaasi kasutamist. Taolise rakenduse disainimiseks on peamiselt kaks varianti [40]:

1. genereerida andmebaas domeenimudelite põhjal;
2. genereerida domeenimudelid andmebaasi põhjal.

Antud rakenduse teostamisel kasutatakse esimest varianti. Rakenduse algfaasis võib andmemudel tihti muutuda ning pidev andmebaasi skeemi manuaalne muutmine on ajakulukas.

EF Core võimaldab domeenimudelite põhjal neile vastava andmebaasi genereerimist [41]. Entity Framework (EF) Core on ORM, mis võimaldab andmebaasiga domeenimudelite kaudu suhelda [42].

Tööriista kasutamiseks on vaja domeenimudelid ning *DbContext* klass korrektselt üles seada, et Entity Framework oskaks nende põhjal korrektsed tabelid ja nendevahelised seosed luua. Joonisel 3 on toodud näide korrektsest domeenimudelist, kus on kasutatud erinevaid võtteid:

- koodi kordavuse vähendamiseks on laiendatud baasklassi, mis sisaldab mudelite ühiseid välju. Sealhulgas on ka identifikaator, mille kasutamine on vajalik;
- välja piirangute seadmiseks on lisatud maksimaalne sõne pikkuse limiit;
- on loodud seosed mudelitega, mis on üks-mitmele seose esimeses rollis;
- on loodud seosed mudelitega, mis on üks-mitmele seose teises rollis;
- on kasutatud selgitavaid atribuute seoste loomiseks. Vajalik juhul kui EF ei saa andmetüübi ja nime põhjal otsust langetatud.

```
public class Comment: DomainEntity
{
    [MaxLength(1000)]
    public string Text { get; set; } = default!;

    public Guid UserId { get; set; }
    public AppUser? User { get; set; }

    public Guid UrlId { get; set; }
    public Url? Url { get; set; }

    public Guid? ParentCommentId { get; set; }
    public Comment? ParentComment { get; set; }

    public Guid? ReplyToCommentId { get; set; }
    public Comment? ReplyToComment { get; set; }

    public ICollection<CommentReaction>? CommentReactions { get; set; }

    [InverseProperty(nameof(ParentComment))]
    public ICollection<Comment>? ParentCommentReplies { get; set; }

    [InverseProperty(nameof(ReplyToComment))]
    public ICollection<Comment>? DirectCommentReplies { get; set; }
}
```

Joonis 3. Koodilõik domeenimudelist.

Erinevad andmebaasi juhtimissüsteemid toimivad erinevalt. Selleks, et EF valitud andmebaasiga suhelda oskaks, on vaja lisada vastav teek. Populaarsemate valikute jaoks, seehulgas PostgreSQL, on teegid tasuta kasutamiseks olemas [43]. Siin on hästi näha, miks levinud tehnoloogiate kasutamine on üldiselt hea otsus.

Arenduse ajal kasutatakse PostgreSQL andmebaasi asemel mälupõhist andmebaasi. Mälupõhine andmebaas ei kasuta genereeritud andmebaasi loomise faile ehk peale igat domeeni muudatust ei ole vaja uusi migratsiooni faile genereerida. Lisaks hoiab see aega kokku andmebaasi käivitamise arvelt. Rakenduse käivitamisel andmebaasi valimist vastavalt keskkonnale on näidatud joonisel 4.

```
if (builder.Environment.IsDevelopment())
{
    builder.Services.AddDbContext<AppDbContext>(options =>
        options.UseInMemoryDatabase("dev"));
}
else
{
    builder.Services.AddDbContext<AppDbContext>(options =>
        options.UseNpgsql(connectionString));
}
```

Joonis 4. Koodilõik andmebaasi valimisest sõltuvalt keskkonnale.

5.2.3 Kasutajate haldamine

Rakendus peab võimaldama kasutajaks registreerimist ja sisselogimist. Selleks on kasutusel ASP.NET Core Identity teek. Teegi kasutamine vähendab muidugi ka arendusele kuluvat aega, aga selle peamiseks eeliseks enda loodud lahenduse ees, on turvalisuse tagamine. Võib kindel olla, et kui mõni turvaoht peaks tekkima, parandatakse see mõne uuendusega üsna pea. Eriti oluline on see seetõttu, et rakendus peab võimaldama sisselogimist kasutajanime ja parooli alusel ehk paroole tuleb andmebaasis hoiustada. Paroole ei tohiks ilmselgelt hoiustada lihtsalt niisama, vaid nendest peaks esmalt räsi looma. Ainult parooli räsiks muutmisest ka ei piisa. Samad paroolid annavad võrdse räsi ehk räsitabelist on võimalik räsi kaudu parool teada saada. Selle vältimiseks tuleb räsile lisada sool. Identity teek just seda paroolidega teebki.

Informatsiooni sellest, et kasutaja on sisselogitud, hoitakse küpsises. Teegi poolt on küpsise seadistamine tehtud äärmiselt lihtsaks. Joonisel 5 on näha, kuidas esialgset küpsist on vastavalt rakenduse vajadusele muudetud.

```
builder.Services.ConfigureApplicationCookie(options =>
{
    options.Cookie.Name = "auth";
    options.Cookie.HttpOnly = true;
    options.Cookie.SecurePolicy = CookieSecurePolicy.Always;
    options.Cookie.SameSite = SameSiteMode.None;
    options.Cookie.IsEssential = true;
    options.SlidingExpiration = true;
    options.ExpireTimeSpan = TimeSpan.FromDays(30);
    options.Events.OnRedirectToLogin = redirectContext =>
    {
        redirectContext.Response.StatusCode =
            StatusCodes.Status401Unauthorized;
        return Task.CompletedTask;
    };
});
```

Joonis 5. Koodilõik küpsise seadistamisest.

Küpsiseid hoiustatakse kliendi brauseris ehk vaikimisi on neile võimalik JavaScriptiga ligi pääseda. See on aga turvarisk, mille vältimiseks tuleb kasutada *HttpOnly* atribuuti, mis lubab küpsist muuta vaid serveril. Esirakenduseks on brauseri pistikprogramm ehk selle domeen erineb alati tagarakenduse omast. Sellisel juhul peab *SameSite* välja väärtuseks olema *None*, sest päring ei tule kunagi samalt domeenilt, kus küpsis seatakse. Kui aga *SameSite* välja väärtuseks on *None*, peab kindlasti kasutama *Secure* atribuuti. *Secure* atribuut tähendab, et küpsist saab saata ainult üle HTTPS protokolli. [34]

5.2.4 Sama veebilehe tuvastamine

Et kasutajad saaksid mistahes veebilehe kohta informatsiooni jagada, tuleb veebilehti teineteisest eristada. Seda saab teha kasutades veebilehe URL-i.

Ühele veebilehele võib aga vastata mitu erinevat URL-i. Selleks, et seda probleemi vähekenegi parandada, tuleb URL-i töödelda. Täielikult töötavat lahendust sellele probleemile ei ole, sest see kuidas URL-i veebilehel kasutatakse sõltub konkreetsest veebirakendusest. Veebilehe sisu põhjal ei saa ka otsust langetada, sest sisu võib muutuda ja ei pruugi kõigile kättesaadav ollagi.

URL koosneb erinevatest osadest (joonis 6). Järgneb loetelu osadest ja sellest, kuidas neid veebilehtede grupeerimisel töödeltakse:

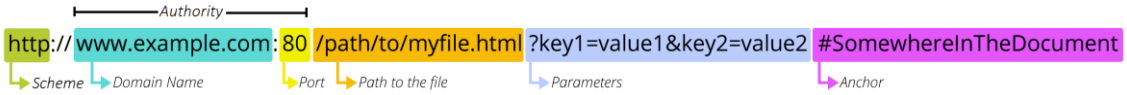
- Protokoll – veebilehtedel on selleks enamasti kas HTTP või HTTPS [44]. Üldiselt peaksid need mõlemad tagastama sama sisu ja seetõttu grupeerimisel seda osa ei kasutata.

- Autoriteet – enamasti domeeni nimi IP-aadressi asemel ning üldiselt standard porte 80 ja 443 ei kuvata [44]. Domeeni nimi ei ole tõstutundlik [45] ja seda võetakse arvesse ka grupeerimisel. Ajalooliselt on kasutusele jäänud www alamdomeeni kasutamine ning paljud veebilehed on kättesaadavad nii seda kasutades kui ka mitte kasutades. Grupeerimisel ei arvestata www prefiksit.

- Asukoht – määrab teekonna ressursini. Kasutus sõltub veebirakendusest ja seetõttu jääb muutumatuks.

- Parameetrid – võti-väärtus paarid. Üldjuhul ei tohiks parameetrite järjekord veebilehel kuvatavat sisu mõjutada ja seetõttu järjestatakse töötlemisel paarid tähestiku järgi. Osad parameetrid ei pruugi veebilehe sisu mõjutadagi, kuid seda ei ole võimalik puhtalt URL-i alusel välja selgitada ja seetõttu ühtegi parameetrit arvestusest välja ei jäeta.

- Ankur – dokumendi sisene asukoht, kuhu brauser automaatselt veebilehte avades kerib [44]. Töötlemisel seda osa ei kasutata.



Joonis 6. URL-i osad [44].

URL-i töötlemine aitab grupeerimisel suurendada samade veebilehtede hulka, kuid ei ole täielik ega veakindel lahendus.

5.2.5 Esirakendusega suhtlemine

Esirakendusega suhtlemiseks kasutatakse peamiselt REST API-t. Selle jaoks on ASP.NET Core raamistikul API kontrollerid. Kontrolleri tööks on kontrollida sisendit, edastada valideeritud andmed äriloogika kihile ja tõlkida tagastatud vastus õigele kujule. Kontrollerid ja äriloogika kiht kasutavaid erinevaid objekte, et tagarakenduse muutudes, jääks API muutumatuks. Vastasel juhul võib esirakendus saada ootamatul kujul vastuseid. Näidist API kontrolleri meetodist, mis valideerib atribuutide abil sisendit on näha joonisel 7.

```
[HttpGet]
public async Task<ResponseWithPaging<Comment>> GetAll(
    [FromQuery] string url,
    [FromQuery] [Range(1, int.MaxValue)] int pageNr = 1,
    [FromQuery] [Range(1, 100)] int pageSize = 25,
    [FromQuery] ESort sort = ESort.Top)
{
    Guid? userId = User.IsAuthenticated() ? User.GetUserId() : null;

    var (comments, pageCount) = await _uow.CommentService
         .GetAll(userId, url, sort, pageNr, pageSize);

    return new ResponseWithPaging<Comment>
    {
        Data = comments.Select(_commentMapper.Map)!,
        PageNr = pageNr,
        PageSize = pageSize,
        PageCount = pageCount
    };
}
```

Joonis 7. Koodilõik API kontrolleri meetodist.

Selleks, et vea korral rakendus ei lõpetaks töötamist ja kasutajatele tagastataks arusaadav veateade, tuleb tegeleda ka veahaldusega. Vältimaks koodi korduvust, teostatakse veahaldus kontrollerite asemel eraldi kihis. Päring läbib enne kontrollerini jõudmist mitu erinevat kihti. Raamistik võimaldab sinna ka enda kihte lisada. Joonisel 8 on näha, et vea tekkimisel kirjutatakse tegelik viga logisse ning rakenduse lõppemise asemel tagastatakse kasutajale arusaadav ja lihtne veateade.

```
    private async Task HandleExceptionAsync(
        HttpContext context, Exception exception)
    {
        var response = exception switch
        {
            CustomUserBadInputException e =>
                new ErrorResponse(HttpStatusCode.BadRequest, e.Message),
            _ => new ErrorResponse(HttpStatusCode.InternalServerError,
                "Internal server error. Please try again later.")
        };

        if (response.StatusCode == HttpStatusCode.InternalServerError)
        {
            logger.LogError(exception, "An unexpected error occurred.");
        }
        else
        {
            logger.LogWarning("An error occured: {}", exception.Message);
        }

        context.Response.ContentType = MediaTypeNames.Application.Json;
        context.Response.StatusCode = (int)response.StatusCode;
        await context.Response.WriteAsJsonAsync(response);
    }
```

Joonis 8. Koodilõik veahalduse teostamisest.

Reaalajas sõnumite saatmiseks kasutatakse REST API asemel WebSocket API-t. Selleks on soovitatud kasutada SignalR teeki, mis on vajadusel võimeline tagama soovitud tulemuse ka WebSocket API-ta [46].

SignalR-i kasutades tuleb etteantud baasklassi laiendada ja sinna soovitud funktsionaalsused lisada. Joonisel 9 on välja toodud meetod, mis käivitatakse, kui kasutaja avab reaalajas suhtlemise vaate.

```
public async Task JoinChat(string url)
{
    var urlId = await GetOrCreateUrlId(url);
    var messages = await _uow.MessageService.GetPreviousMessages(urlId);

    await Groups.AddToGroupAsync(Context.ConnectionId, urlId.ToString());
    await Clients.Caller.SendAsync(ReceiveGroupId, urlId);
    await Clients.Caller.SendAsync(ReceiveMessages,
        messages.Select(_mapper.Map));
}
```

Joonis 9. Koodilõik SignalR teegi abil reaalajas suhtlemise alustamisest.

5.3 Esirakendus

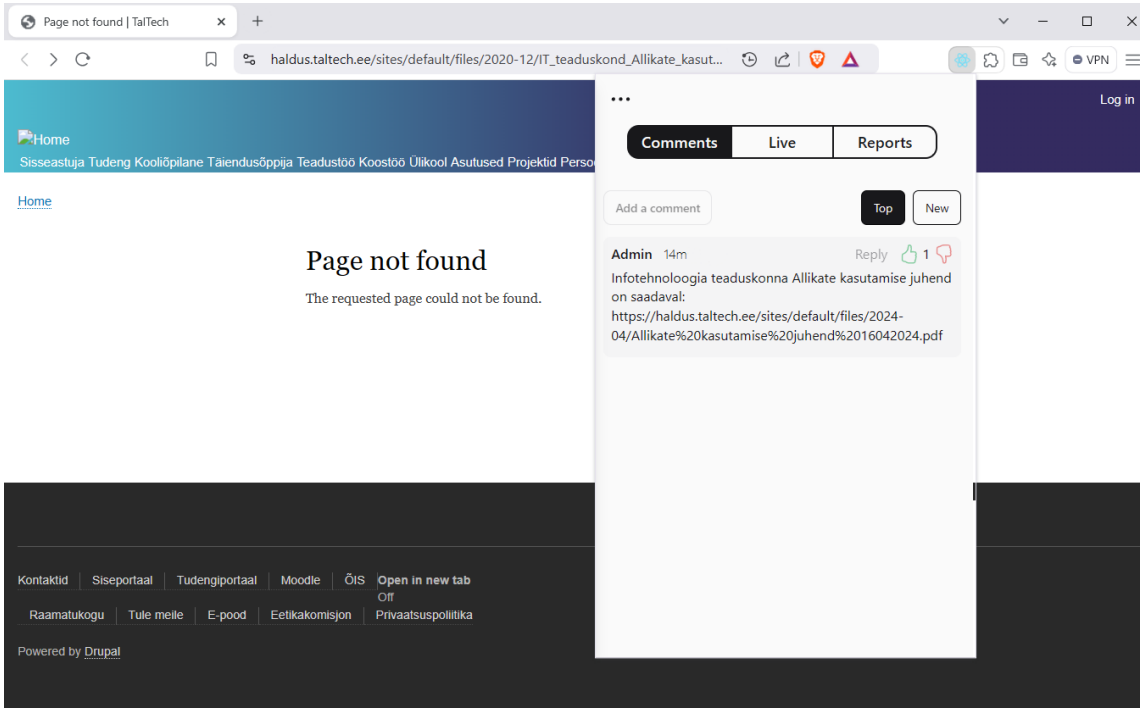
Esirakenduseks on brauseri pistikprogramm ja selle loomiseks kasutatakse TypeScript programmeerimiskeelt, React teeki ja Tailwind raamistikku.

5.3.1 Arhitektuur

Brauseri pistikprogramm on ülesehituselt sarnane tavalise veebirakenduse esirakendusega. Esinevad vaid mõned erinevused.

Esiteks asub rakendus kasutaja masinas. Selleks, et brauser teaks milliseid õigusi rakendus vajab ning oskaks seda õigesti avada ja kasutada, tuleb luua konfiguratsiooni fail [47]. Konfiguratsiooni failid on üldjuhul erinevates brauserites sarnased. Loodav rakendus peab töötama Google Chrome brauseris ehk tuleb kasutada Chromiumile vastavat konfiguratsiooni faili. See tähendab, et loodav rakendus töötab lisaks Chrome-ile ka teistes Chromiumi-põhistes brauserites nagu Microsoft Edge, Opera ja Brave.

Teiseks, rakenduse kuvamiseks on erinevaid viise. Kolm varianti, mis sobivad hästi antud rakenduse puhul on hüpikaken, külgpaneel ja rakenduse süstimine avatud veebilehe sisse. Veebilehte süstimine jääb esialgu kõrvale, sest see võib osutuda liiga pealetükkivaks. Külgpaneel on oma käitumise poolest parim variant, sest see on veidi suurem kui hüpikaken ning seda ei suleta automaatselt. Teisalt lisati külgpaneel Chrome brauserile alles 2022. aasta keskel, mis teeb selle võrdlemisi uueks funktsionaalsuseks [48]. Kasutajatel võib puududa varasem kokkupuude kõrvalpaneelide kasutamisega ning seetõttu kuvatakse pistikprogrammi hüpikaknana. Hüpikaknana kuvatud pistikprogrammi paiknemist avatud veebilehe suhtes on näha joonisel 10.



Joonis 10. Pistikprogrammi paiknemine avatud veebilehe suhtes.

Kolmandaks vajab pistikprogramm teavet avatud lehe kohta. Avatud veebilehe URL-i leidmiseks tuleb kasutada Chrome API-t [49]. Chrome API kasutamiseks tuleb lisada ka vastavad TypeScripti tüübid. Joonisel 11 on toodud välja programmikood sellest, kuidas Chrome API kaudu saadakse avatud veebilehe URL.

```
const queryOptions = { active: true, lastFocusedWindow: true };
chrome.tabs.query(queryOptions, tabs => {
    setUrl(tabs[0].url);
});
```

Joonis 11. Koodilõik avatud veebilehe URL-i pärimisest Chrome API kaudu.

Neljandaks, erinevalt tavalistest veebirakendustest, ei kasuta pistikprogramm vaadete vahel navigeerimiseks aadressiriba. Reactil on selle jaoks pakkuda *MemoryRouter*, mis hoiab URL-i aadressiriba asemel mälus [50]. See on mugav lahendus, sest rakenduse disaini see ei muuda. Joonisel 12 on välja toodud programmikood mälupõhise vaadetevahelise navigeerimise lisamisest

```
root.render(
    <MemoryRouter initialEntries={["/comments"]}>
        <Routes>
            <Route path={"/"} element={<Root />}>
                <Route path={"comments"} element={<CommentsPage />} />
                <Route path={"live"} element={<LivePage />} />
                <Route path={"reports"} element={<ReportsPage />} />
                <Route path={"login"} element={<LoginPage />} />
                <Route path={"register"} element={<RegisterPage />} />
            </Route>
        </Routes>
    </MemoryRouter>
);
```

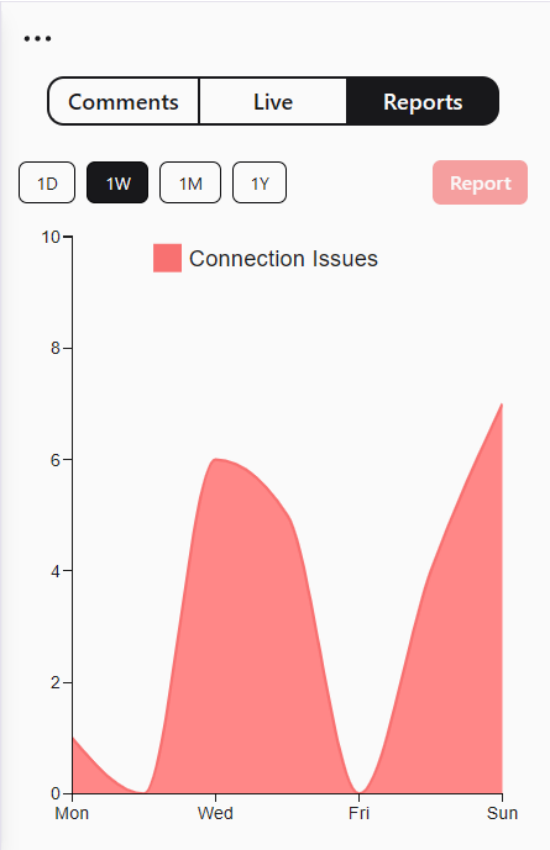
Joonis 12. Koodilõik mälupõhise navigeerimise lisamisest.

5.3.2 Kujundus

Esirakenduse teostamisel on oluline roll ka selle kujundusel. Alternatiivseid lahendusi analüüsides selgus, et ühe populaarse rakenduse negatiivseks pooleks oli selle keeruline disain. Antud lahenduses üritatakse selle vea vältimiseks luua lihtne ja minimalistlik kujundus. Minimalistlik disain on eriti oluline brauseri pistikprogrammi puhul, sest rakendus hõlmab vaid väikese osa ekraanist ja põhifookuseks on veebileht ise.

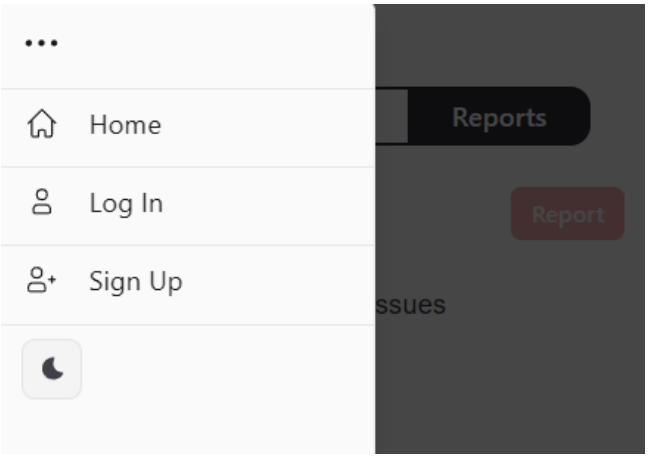
Tuleb tagada järjepidev, ühtlane ja arusaadav kujundus. React teeki kasutades koosneb rakendus komponentidest, mis soodustab elementide taaskasutamist. Järjepidevat disaini soodustab ka Tailwind raamistiku kasutamine, sest kujundamiseks on saadaval eeldefineeritud värvid. Mitmes kohas on kasutatud näiteks valikut muutvat nuppu. Sellele vastav programmikood, kus on näha ka eeldefineeritud värvide kasutust, on toodud välja joonisel 16.

Joonisel 13 on välja toodud tõrgete esinemise vaade, kus on näha erinevaid meetodeid järjepideva, ühtlase ja arusaadava kujunduse tagamiseks. Kuvatud vaate selgitamiseks tuuakse valitud varianti esindavad nupud teiste seast esile. Arusaadavuse tagamiseks kasutatakse limiteeritud hulk värve. Vaate oluliseim osa, tõrgete esinemise statistika ja sellest teavitamine, tuuakse värviga esile. Funktsionaalsuste kättesaamatuse kohta antakse märku taustaga sulanduvate toonidega. Kättesaamatuid funktsionaalsusi ei peideta, et kasutaja ei hakkaks funktsionaalsusi mujalt otsima. Lisateavet kuvatakse hiirega elemendi peal olles, et hoida kujundus minimalistlik. Lisateave aitab vähendada kasutajas segadust, selgitades näiteks miks funktsionaalsus ei ole saadaval või mis on konkreetse graafi punkti täpne väärtus.



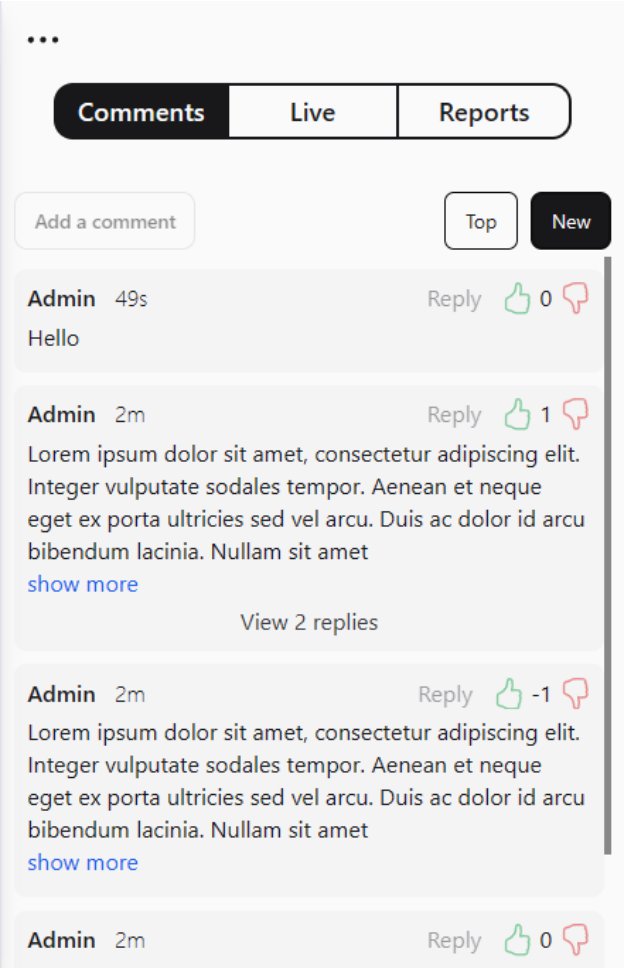
Joonis 13. Pistikprogrammi tõrgete esinemise vaade.

Mõnikord saab kasutajatele edastada teavet kiiremini ja mugavamalt kasutades teksti asemel visuaalseid elemente. Rakenduses on navigeerimise lihtsustamiseks lisatud menüü elementidel teksti kõrvale ikoonid (joonis 14). Lisaks kuvatakse tõrgete statistikat parema ülevaate andmiseks joonisena. Oluline on ka kasutada sobivat joonist. Joongraaf sobib hästi numbriliste andmete ja nende muutuste kuvamiseks [51]. Graafi horisontaalsel ehk x-teljel peaks olema aeg ja vertikaalsel ehk y-teljel sellele vastavad andmed [51]. Vältimaks graafil müra tekkimist, tuleb valida ka sobiva vahemikuga punktid.



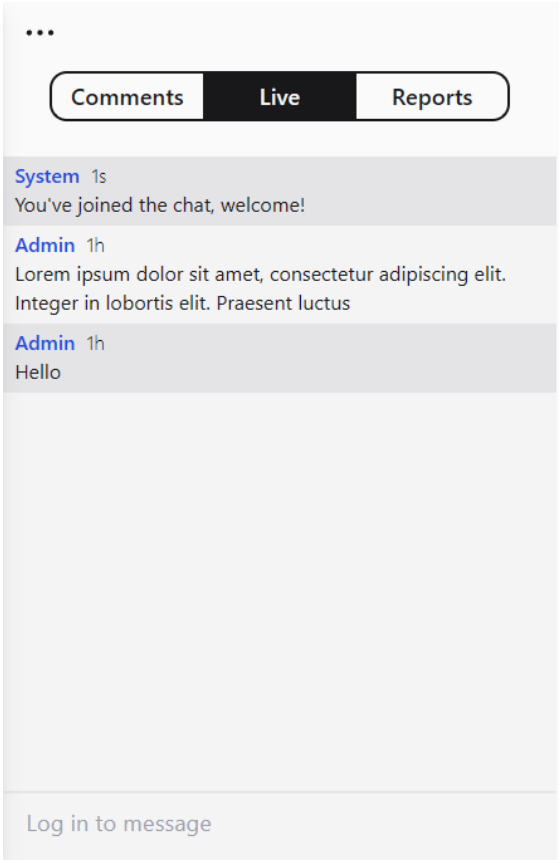
Joonis 14. Pistikprogrammi menüü.

Joonisel 15 on välja toodud kommentaaride vaade. Vaatel on näha, et kommentaaride sorteerimiseks taaskasutatakse tõrgete esinemise vaatel kasutatud elemente. Kujundus on jätkuvalt minimalistlik ning tagasihoidlik. Hetkel mittesaadaval olevad funktsionaalsused tuuakse jällegi välja taustaga sulanduvate toonidega. Pikkade kommentaaride sisu ja vastuseid ei kuvata vaikimis täies mahus, et mitte hõlmata kogu vaadet. Kommentaarid on küll jaotatud lehekülgedeks, aga lehti vahetatakse automaatselt alla kerides. Nii-viisi ei pea kasutaja manuaalselt lehti vahetama ja ei koormata vaadet üleliigsete nuppudega.



Joonis 15. Pistikprogrammi kommentaaride vaade.

Joonisel 16 on välja toodud reaalajas suhtlemise vaade. Kujundus on jätkuvalt minimalistlik. Juhul kui funktsionaalsus pole kättesaadaval kasutatakse taustaga sulanduvaid toone ning lisatakse ka lühike põhjendus. Selleks, et sõnumite lugemine oleks mugavam, eristatakse need naabritest erineva värviga. Kasutajale kuvatakse vaadet avades süsteemi poolt sõnum, mis kinnitab kas ruumiga ühinemine õnnestus või mitte. Lisaks laetakse hiljuti saadetud sõnumid, et lisada vestlusele eelnev kontekst. Sõnumeid võib koguneda palju. Mugavaks kasutamiseks, keritakse vaikimisi automaatselt saabunud uue sõnumi juurde. Erandiks on olukord, kus kasutaja on manuaalselt mingi sõnumi juurde kerinud.



Joonis 16. Pistikprogrammi reaalajas suhtlemise vaade.

Et rakendus sobiks kokku avatud veebilehega, kasutatakse tagasihoidlikke värve. Parema tulemuse saamiseks võimaldatakse kasutajatel kuvatud teemat vahetada. Tailwind raamistikku kasutades on teemapõhine kujundus lihtne. Tuleb lisada teema prefiks kujunduse klassidele, mida soovitakse kasutada vaid valitud teema korral. Joonisel 17 on näha, et osadel klassidel on *dark:* prefiks ehk need klassid kehtivad ainult tumeda teema kasutamisel.

```
interface IProps<T> {
  value: T;
  selectedValue: T;
  setValue: (value: T) => void;
  displayText: string;
}

const OptionButton = (props: IProps<any>) => {
  return (
    <button
      onClick={() => props.setValue(props.value)}
      className={`px-3 py-1.5 rounded-md text-xs transition border
        border-zinc-950 dark:border-zinc-50
        ${props.value === props.selectedValue
          ? "bg-zinc-900 text-zinc-50 dark:bg-zinc-100 dark:text-zinc-900"
          : "hover:bg-zinc-100 dark:hover:bg-zinc-700"
        }`}>
      {props.displayText}
    </button>
  );
};
```

Joonis 17. Koodilõik taaskasutatavast React komponendist.

5.3.3 Tagarakendusega suhtlemine

Tagarakendusega REST API kaudseks suhtlemiseks kasutatakse Axios teeki. Teegil on mitmeid eeliseid olemasoleva meetodi kasutamise ees, millest üheks on JSON-i kujul esitatud andmete ja andmetele vastava objekti vaheline tõlkimine [52]. Axios küpsiseid automaatselt serverile ei edasta ja selle võimaldamiseks tuleb Axios vastavalt konfigureerida [53]. Teostamiseks luuakse ühist loogikat sisaldav baasklass, mille laiendamisel saab mugavalt HTTP päringuid luua. Joonisel 18 on näha kommentaari vastuseid päriva meetodi sisu.

```
try {
    const response = await this.axios.get(`replies`, {
        params: {
            parentCommentId: parentCommentId,
            pageNr: pageNr,
            pageSize: pageSize,
            sort: sortBy
        }
    });

    return response.data as IResponseWithPaging<IComment[]>;
} catch (e) {
    return undefined;
}
```

Joonis 18. Koodilõik Axios teegi abil HTTP päringu tegemisest.

Reaalajas suhtlemiseks saab kasutada sama teeki, mida kasutatakse tagarakenduse poolel. Teostamisel luuakse jällegi koodi paremaks hallatavuseks SignalR seadistamist sisaldav baasklass. Küpsis lisatakse SignalR-i kasutades automaatselt igale päringule kaasa [54]. Joonisel 19 on näha koodilõikku reaalajas suhtlemise alustamisest.

```
public async startSession (groupName: string) {
    if (!this.isDisconnected() || this.isDisconnecting()) return;

    await this.connection.start();
    await this.connection.send("JoinChat", groupName);
}
```

Joonis 19. Koodilõik SignalR teegi abil reaalajas suhtlemise alustamisest.

6 Tulemused

Lõputöö raames valmis rakendus, mis võimaldab kasutajatel üksteisele teavet jagada erinevate veebilehtede kohta. Valminud rakendus täidab kõik funktsionaalsed ja mittefunktsionaalsed nõuded. Arendusprotsessi optimeerimiseks loodud kasutajalugudest said teostatud need, mille prioriteediks oli määratud kõrge või keskmine. Lahendust on võimalik proovida järgides esirakenduse repositooriumi juures olevat paigaldamise juhendit (lisa 2).

Lahenduse teostamiseks valiti sobilikud töövahendid. Olgugi, et tavalise veebirakenduse loomine oleks osutunud lihtsamaks ja rohkematele kättesaadavaks, loodi parema tulemuse saamiseks brauseri pistikprogramm. Alternatiivsetest lahendustest kõige sarnasema rakenduse suureks miinuseks oli selle keeruline disain. Seetõttu pandi pistikprogrammi loomisel rõhku ka lihtsa ja minimalistliku kujunduse peale. Olenemata sellest, et tegemist on alles minimaalse töötava tootega.

6.1 Võimalikud edasiarendused

Rakenduse üheks nõudeks oli selle edasiarendatavus. Nõue sai täidetud ning edasiarendamise võimalusi on mitmeid. Alustada võiks teostamata jäänud madala prioriteediga kasutajalugudest või lõputöö skoobist väljajäänu põhjal uute kasutajalugude loomisest.

Kindlasti oleks mõttekas luua rakendusele automaattestid leidmaks varakult esinevaid vigu. Hea oleks seadistada arenduskeskkond nii, et peale igat uuendust käivitatakse versioonihaldus keskkonnas automaattestid. Nii-viisi saab olla kindel, et lisatud muudatused ei põhjusta olemasolevas probleeme. See parandab rakenduse kvaliteeti ja muudab ka rakendusega töötamise arendaja jaoks meeldivamaks, sest võib julgelt muudatusi sisse viia.

Pistikprogrammi võiks muuta kättesaadavaks ka Chromiumi välistes brauserites. Selle teostamine peaks olema võrdlemisi lihtne, sest konfiguratsiooni failid on sarnased ja veebilehega suhtlemiseks tuleb vaid mõnda muud API-t kasutada. Lisaks, hetkel on

pistikprogramm kuvatud hüpikaknana, et ta oleks oleks maksimaalselt tagasihoidlik. Võimalik oleks lisada erinevaid viise rakenduse kuvamiseks ja lasta kasutajal valida eelistatud variant. Näiteks saaks pistikprogrammi kuvada ka veebilehe sisu hulgas või külgpaneelina. Nende eeliseks oleks, et rakendust ei suleta automaatselt. Esirakenduseks saab luua ka tavalise veebirakenduse, mis oleks otsingumootori poolt indekseeritav ja kättesaadav ka mobiilil. Selle teostamine oleks lihtne, sest tagarakenduse liides jääks muutumatuks ning suurt osa olemasoleva esirakenduse programmikoodist saaks taaskasutada.

Lahendusele saaks lisada erinevaid funktsionaalsusi. Rakendus peaks võimaldama erinevaid viise autentimiseks. Näiteks e-mailile saadetud ühekordse parooli või Google konto kaudu sisselogimist. Lisada saaks ka tasulise kasutajarolli, millel oleks tavakasutajaga võrreldes rohkem funktsionaalsusi. Tasuline kasutaja võiks saada kasutada näiteks erinevaid ikoone või emotikone. Muidugi oleks tarvis lisada ka administraator kasutajaroll manuaalseks modereerimiseks ning mõned lihtsamad märksõna filtrid töö automatiseerimiseks.

Lõpuks tuleks optimeerida andmebaasi päringute tegemist. Näiteks kommentaaride reageerimised arvutatakse hetkel iga päringuga uuesti. Võimalik oleks lisada kommentaari tabelile väli, kus hoitakse viimati arvutatud tulemuse väärtust ja ajatemplit, et vähendada ressursikulu. Üle peaks vaatama tegelikult kõik päringud ja lisama indeksid väljadele, kus need olla võiks. Hetkel on indekseeritud kõik välisvõtmed, sest selle teeb Entity Frameworki vaikimis ise automaatselt ära [55].

6.2 Kasutajatestimine

Esmase tagasiside saamiseks paluti neljal inimesel, kes enam-vähem sobisid sihtgruppi, proovida valminud rakendust. Tagasiside kogumiseks koostati Google Forms keskkonnas küsitlus, millele kasutajad testimise järgselt vastata said. Küsimustiku töötlemata kujul vastused on toodud välja lisas 3.

Kasutajatele tehti lühike ülevaade rakenduse põhimõttest detailidesse minemata. Selle eesmärgiks oli simuleerida realistlikku olukorda, kus kasutaja loeb enne rakenduse kasutamist vaid lühikokkuvõtet ning päris täpselt kõigest aru ei saa. Testimise ajal jälgis

autor kasutajate tegevusi, et saada parem ülevaade kasutajakogemusest ja märgata ka väiksemaid probleeme, mis küsimustikust välja ei pruugi tulla.

Kasutajaliidest hinnati 5-punkti skaalal, kus 1 on kõige kehvem hinnang ning 5 kõige parem, hindega 4,25. Kasutajakogemuse keskmiseks hindeks märgiti 4. Positiivse poole poolt toodi välja, et rakenduse kasutamine on lihtne, selge ja arusaadav. Teisalt toodi välja, et kasutajaliides võiks veelgi selgem olla lisades vaadetele nende kirjeldus. Soovitati ka parooli nõudeid lihtsustada – kasutusel olid raamistiku vaikimis valikud, mis võivad tõepoolest üpriski tülikaks osutuda. Jälgides täheldas autor, et kasutajatel jäi segaseks, mille kohta pistikprogrammis suhtlemine käib. Küsimustikust seda aga välja ei tulnud ehk võib-olla võttis sellest arusaamine lihtsalt veidi aega. Tõenäoliselt aitaks probleemi vastu kasutusjuhendi lisamine, mida kuvada esmasel kasutamisel. (lisa 3)

Üldiselt jäädi rakendusega rahule. Keskmiselt pakkus kasutajatele kolmest peamisest funktsionaalsusest huvi vaid kaks. Huvitaval kombel jaotusid need täpselt võrdselt ehk funktsionaalsused said töö jooksul hästi valitud. Mainiti ka erinevate funktionaalsuste lisamist, mida saab kindlasti arvesse võtte edasiarendusel. Vastanutest ei olnud keegi ühtegi alternatiivset lahendust varasemalt kasutanud. Samas olid neist 75% huvitatud rakenduse kasutamisest ka tulevikus ning valmis soovitama seda teistelegi. (lisa 3)

7 Kokkuvõte

Töö käigus uuriti ja analüüsiti probleemi, et suur osa veebilehtedest toetab vaid ühesuunalist teabe liikumist. Leiti, et selle parandamine võib tõsta kasutajakogemust võimaldades kasutajatel küsimusi esitada, arvamust avaldada ning teistega suhelda.

Töö eesmärgiks sai rakenduse loomine, mille abil saaksid kasutajad iga veebilehe kohta teavet jagada. Lahendus peaks parandama veebi kasutamist muutes info liikumine mitmesuunaliseks ka lehtedel, kus see vaikimis võimalik ei ole.

Eesmärgini jõudmiseks võrreldi erinevaid suhtlusmeetodeid ning leiti, et kõige sobivam on informatsiooni edastada teksti kaudu. Olulisemate tekstipõhiste funktsionaalsuste leidmiseks uuriti ja võrreldi populaarsemaid platvorme. Leiti, et kindlasti peaks võimaldama kommenteerimist, kommentaaridele vastamist ja reageerimist ning reaalajas suhtlemist. Neile lisandus tõrgetest teavitamise funktsionaalsus. Seejärel uuriti ja analüüsiti teisi sarnaseid lahendusi. Leitu põhjal pandi paika nõuded, töö skoop ning kasutajalood arenduse lihtsustamiseks.

Probleemi lahendamiseks täpsustati rakenduse arhitektuur ning valiti sobivad tehnoloogiad. Esirakenduse teostamiseks valiti TypeScript programmeerimiskeel, React teek ja Tailwind raamistik. Tagarakenduse jaoks kasutati C# programmeerimiskeelt koos ASP.NET Core raamistikuga. Andmete hoiustamiseks valiti PostgreSQL andmebaas ja rakenduste vaheliseks suhtlemiseks kasutati lisaks REST API-le ka WebSocket API-t, et võimaldada reaalajas suhtlemist. Sisselogimist võimaldatakse nime ja parooli alusel ning teavet sellest hoitakse küpsises.

Lahendusena valmis hüpikaknana kuvatav Chromium brauseri pistikprogramm, mille saab avada vaid ühe nupu vajutusega olles mistahes veebilehel. Rakenduse teostamiseks täideti kõrgema prioriteediga kasutajalood, mille tulemusena valmis minimaalne töötav toode, mis võimaldab kasutada peamisi funktsionaalsusi. Hea kasutajakogemuse tagamiseks võeti arvesse, et pisikprogramm peaks veebilehte vaid toetama ja olema seetõttu kujunduselt tagasihoidlik, lihtne, mugav ning arusaadav. Valminud lahenduse

hindamiseks viidi läbi kasutajatestimine, mille tagasiside oli valdavalt positiivne ja kindlasti kasulik edasiseks arenduseks.

Kasutatud kirjandus

[1] „August 2024 Web Server Survey,“ Netcraft, [Võrgumaterjal]. Available: https://www.netcraft.com/blog/august-2024-web-server-survey/. [Kasutatud 5. oktoober 2024].

[2] „Top Websites Ranking,“ Similarweb, [Võrgumaterjal]. Available: https://www.similarweb.com/top-websites/. [Kasutatud 5. oktoober 2024].

[3] S. Pichai ja M. Cierniak, „Official Google Blog: Help and learn from others as you browse the web: Google Sidewiki,“ Google, 23. september 2009. [Võrgumaterjal]. Available: https://googleblog.blogspot.com/2009/09/help-and-learn-from-others-as-you.html. [Kasutatud 6. oktoober 2024].

[4] A. Eustace, „Official Google Blog: A fall spring-clean,“ Google, 2. september 2011. [Võrgumaterjal]. Available: https://googleblog.blogspot.com/2011/09/fall-spring-clean.html. [Kasutatud 6. oktoober 2024].

[5] „About Hypothesis,“ Hypothesis, [Võrgumaterjal]. Available: https://web.hypothes.is/about/. [Kasutatud 6. oktoober 2024].

[6] „Annotation Basics: Hypothesis,“ Hypothesis, [Võrgumaterjal]. Available: https://web.hypothes.is/help/annotation-basics/. [Kasutatud 6. oktoober 2024].

[7] „Incident Detection Method,“ Downdetector, [Võrgumaterjal]. Available: https://downdetector.com/methodology/. [Kasutatud 6. oktoober 2024].

[8] „Is It Down Right Now?,“ isitdownrightnow, [Võrgumaterjal]. Available: https://www.isitdownrightnow.com/. [Kasutatud 6. oktoober 2024].

[9] „About Disqus,“ Disqus, [Võrgumaterjal]. Available: https://disqus.com/about-us/. [Kasutatud 6. oktoober 2024].

[10] „How does Disqus work?,“ Disqus, [Võrgumaterjal]. Available: https://help.disqus.com/en/articles/1717050-how-does-disqus-work. [Kasutatud 6. oktoober 2024].

[11] „Usage Statistics and Market Share of Content Languages for Websites,“ [Võrgumaterjal]. Available: https://w3techs.com/technologies/overview/content_language. [Kasutatud 7. oktoober 2024].

[12] N. Angell, „Hypothesis: 7 Million Annotations and Counting : Hypothesis,“ Hypothesis, 11. september 2019. [Võrgumaterjal]. Available: https://web.hypothes.is/blog/hypothesis-7-million-annotations-and-counting/. [Kasutatud 19. november 2024].

[13] D. Ha, „The Numbers of Disqus,“ Disqus, 4. mai 2011. [Võrgumaterjal]. Available: https://blog.disqus.com/the-numbers-of-disqus. [Kasutatud 19. november 2024].

[14] N. J. Stroud, E. V. Duyn ja C. Peacock, „News Commenters and News Comment Readers,“ The University of Texas at Austin, 14. märts 2016. [Võrgumaterjal]. Available: https://mediaengagement.org/research/survey-of-commenters-and-comment-readers/. [Kasutatud 19. november 2024].

[15] P. Becker, G. Tebes, D. Peppino ja L. Olsina, „Applying an Improving Strategy that embeds Functional and Non-Functional Requirements Concepts,“ *Journal of Computer Science and Technology,* kd. 19, nr 2, p. 153, 1999.

[16] „Terminology and Notation,“ IEEE, [Võrgumaterjal]. Available: https://cmte.ieee.org/tcrts/education/terminology-and-notation/. [Kasutatud 7. oktoober 2024].

[17] „Browser Market Share Worldwide,“ Statcounter, [Võrgumaterjal]. Available: https://gs.statcounter.com/browser-market-share/all/worldwide/2023. [Kasutatud 7. oktoober 2024].

[18] G. M. Hall, „Adaptive Code via C#: Agile coding with design patterns and SOLID principles,“ Washington, Microsoft Press, 2014, p. 15.

[19] R. E. Johnson, „Frameworks=(components+ patterns).,“ *Communications of the ACM,* kd. 40, nr 10, p. 41, 1997.

[20] „Technology | 2024 Stack Overflow Developer Survey,“ [Võrgumaterjal]. Available: https://survey.stackoverflow.co/2024/technology#most-popular-technologies-language. [Kasutatud 9. oktoober 2024].

[21] „TypeScript: JavaScript With Syntax for Types.,“ Microsoft, [Võrgumaterjal]. Available: https://www.typescriptlang.org/. [Kasutatud 9. oktoober 2024].

[22] „Stack Overflow,“ [Võrgumaterjal]. Available: https://trends.stackoverflow.co/?tags=reactjs,vue.js,angular,svelte,angularjs,vuejs 3. [Kasutatud 9. oktoober 2024].

[23] „State of CSS 2023: CSS Frameworks,“ [Võrgumaterjal]. Available: https://2023.stateofcss.com/en-US/css-frameworks/. [Kasutatud 9. oktoober 2024].

[24] „Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.,“ tailwindcss, [Võrgumaterjal]. Available: https://tailwindcss.com/. [Kasutatud 9. oktoober 2024].

[25] „ASP.NET Core | Open-source web framework for .NET,“ Microsoft, [Võrgumaterjal]. Available: https://dotnet.microsoft.com/en-us/apps/aspnet. [Kasutatud 10. oktoober 2024].

[26] „TechEmpower Framework Benchmarks,“ TechEmpower, [Võrgumaterjal]. Available: https://www.techempower.com/benchmarks/#hw=ph&test=plaintext&section=data-r22. [Kasutatud 10. oktoober 2024].

[27] „Next.js by Vercel - The React Framework,“ Vercel, [Võrgumaterjal]. Available: https://nextjs.org/. [Kasutatud 10. oktoober 2024].

[28] „Express - Node.js web application framework,“ [Võrgumaterjal]. Available: https://expressjs.com. [Kasutatud 10. oktoober 2024].

[29] „HTTP | MDN,“ Mozilla, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP. [Kasutatud 10. oktoober 2024].

[30] B. Reselman, „An architect's guide to APIs: SOAP, REST, GraphQL, and gRPC,“ Red Hat, [Võrgumaterjal]. Available: https://www.redhat.com/architect/apis-soap-rest-graphql-grpc. [Kasutatud 10. oktoober 2024].

[31] „The WebSocket API (WebSockets) - Web APIs | MDN,“ Mozilla, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. [Kasutatud 10. oktoober 2024].

[32] „Authentication methods: choosing the right type,“ National Cyber Security Centre, 21. september 2022. [Võrgumaterjal]. Available: https://www.ncsc.gov.uk/guidance/authentication-methods-choosing-the-right-type. [Kasutatud 10. oktoober 2024].

[33] M. Shafran, Seansi- ja JWT-põhiste autentimismeetodite analüüs: võrdlev uuring turvalise rakendamise näidetega, Tallinn: Tallinna Tehnikaülikool, 2023.

[34] „Using HTTP cookies - HTTP | MDN,“ Mozilla, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies. [Kasutatud 12. oktoober 2024].

[35] „Types of Databases - GeeksforGeeks,“ GeeksforGeeks, [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/types-of-databases/. [Kasutatud 12. oktoober 2024].

[36] „What Is A Relational Database (RDBMS)? | Google Cloud,“ Google, [Võrgumaterjal]. Available: https://cloud.google.com/learn/what-is-a-relational-database. [Kasutatud 12. oktoober 2024].

[37] „PostgreSQL: About,“ PostgreSQL, [Võrgumaterjal]. Available: https://www.postgresql.org/about/. [Kasutatud 12. oktoober 2024].

[38] K. Davis, B. Peabody ja P. Leach, „RFC 9562: Universally Unique IDentifiers (UUIDs),“ mai 2024. [Võrgumaterjal]. Available: https://www.rfc-editor.org/rfc/rfc9562.html. [Kasutatud 14. oktoober 2024].

[39] „PostgreSQL: Documentation: 17: 8.12. UUID Type,“ PostgreSQL, [Võrgumaterjal]. Available: https://www.postgresql.org/docs/current/datatype-uuid.html. [Kasutatud 14. oktoober 2024].

[40] „Managing Database Schemas - EF Core | Microsoft Learn,“ Microsoft, [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/ef/core/managing-schemas/. [Kasutatud 15. oktoober 2024].

[41] „Migrations Overview - EF Core | Microsoft Learn,“ Microsoft, [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli. [Kasutatud 15. oktoober 2024].

[42] „Overview of Entity Framework Core - EF Core | Microsoft Learn,“ Microsoft, [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/ef/core/. [Kasutatud 15. oktoober 2024].

[43] „Database Providers - EF Core | Microsoft Learn,“ Microsoft, [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli. [Kasutatud 15. oktoober 2024].

[44] „What is a URL? - Learn web development | MDN,“ Mozilla, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_URL. [Kasutatud 16. oktoober 2024].

[45] „What is a Domain Name? - Learn web development | MDN,“ Mozilla, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_domain_name. [Kasutatud 16. oktoober 2024].

[46] „WebSockets support in ASP.NET Core | Microsoft Learn,“ Microsoft, [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/websockets?view=aspnetcore-7.0. [Kasutatud 16. oktoober 2024].

[47] „Manifest file format | Chrome for Developers,“ Google, [Võrgumaterjal]. Available: https://developer.chrome.com/docs/extensions/reference/manifest#minimal-manifest. [Kasutatud 17. oktoober 2024].

[48] O. Dunk ja A. Steam, „Design a superior user experience with the new Side Panel API | Blog | Chrome for Developers,“ Google, [Võrgumaterjal]. Available: https://developer.chrome.com/blog/extension-side-panel-launch. [Kasutatud 21. november 2024].

[49] „chrome.tabs | API | Chrome for Developers,“ Google, [Võrgumaterjal]. Available: https://developer.chrome.com/docs/extensions/reference/api/tabs. [Kasutatud 17. oktoober 2024].

[50] „React Router: Declarative Routing for React.js,“ [Võrgumaterjal]. Available: https://v5.reactrouter.com/web/api/MemoryRouter. [Kasutatud 17. oktoober 2024].

[51] M. Yi, „A Complete Guide to Line Charts | Atlassian,“ Atlassian, [Võrgumaterjal]. Available: https://www.atlassian.com/data/charts/line-chart-complete-guide. [Kasutatud 20. oktoober 2024].

[52] „Difference between Fetch and Axios for Making HTTP Requests - GeeksforGeeks,“ GeeksForGeeks, [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/difference-between-fetch-and-axios-js-for-making-http-requests/. [Kasutatud 26. oktoober 2024].

[53] „Request Config | Axios Docs,“ [Võrgumaterjal]. Available: https://axios-http.com/docs/req_config. [Kasutatud 26. oktoober 2024].

[54] „Authentication and authorization in ASP.NET Core SignalR | Microsoft Learn,“ Microsoft, [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/aspnet/core/signalr/authn-and-authz?view=aspnetcore-8.0. [Kasutatud 26. oktoober 2024].

[55] „Foreign and principal keys in relationships - EF Core | Microsoft Learn,“ Microsoft, [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/ef/core/modeling/relationships/foreign-and-principal-keys. [Kasutatud 27. oktoober 2024].

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

Mina, Roma Imran Tariq

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Brauseri pistikprogramm veebilehega seotud informatsiooni jagamiseks“, mille juhendaja on Jaanus Pöial
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

01.12.2024

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 – Lähtekood

- Esirakendus: https://github.com/romatariq/chat-app-extension
- Tagarakendus: https://github.com/romatariq/chat-app-api

Lisa 3 – Kasutajate tagasiside

Kas olete varem mõnda sarnast lahendust kasutanud?	Kuidas hindaksite kasutajaliidest?	Kuidas hindaksite rakenduse üldist kasutajakogemust ?	Mis teile lahenduse puhul kõige enam meeldis?	Kas on miskit, mis võiks teie arvates lahenduses teisiti olla või miskit mida juurde lisada?	Kas kasutaksite rakendust ka tulevikus?	Milliseid funktsionaalsusi kasutaksite?	Kas soovitaksite rakendust ka teistele?
Ei	4	4	Reguleerimata jututuba, mis meenutab varajasemat interneti kasutamist (vähem reguleerimist)	Liides on suhteliselt tagasihoidlik (peidus ja väike ekraan, sageli ununeb.	Jah	Kommenteerimine; Reaalajas suhtlemine	Jah
Ei	5	4	Lihtne, arusaadav ja kerge kasutada	Teiste kasutajate blokeerimis/reporti mis lahendus. Liveis like/dislike funktsioon	Jah	Kommenteerimine; Reaalajas suhtlemine;Tõrgetest teavitamine	Jah
Ei	4	4	Lihtsus, darkmode, kiirus.	Selgem kasutajaliides, kirjeldus iga tabi kohta, modernsem lahendus.	Ei	Kommenteerimine; Tõrgetest teavitamine	Ei
Ei	4	4	Annab võimaluse suure grupi inimeste puhul, kes mingit Ühist tegevust teevad, suhelda ilma suurema "clutterita" ja erinevatel teemadel erinevatel lehtedel.	Lihtsam parooli tingimustik, võibolla võimalus private chat alustada teiste kasutajatega?	Jah	Reaalajas suhtlemine;Tõrgetest teavitamine	Jah