TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Kristjan Mill 194080IADB

Filmide ja teiste meediatüüpide jälgimisplatvormi rakendusliidese arendamine

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2023

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: Kristjan Mill

14.05.2023

Annotatsioon

Antud bakalaureusetöö eesmärgiks on luua rakendusliides filmide ja teiste meediatüüpide jälgimisplatvormi jaoks. Loodava rakenduse eesmärgiks on pakkuda ühtset süsteemi, kus kasutaja saab jälgida nii filme, sarju kui ka raamatuid ning pakub ka sotsiaaliseerimis võimalust läbi foorumi.

Töö esimeses osas võrreldakse sarnaseid olemasolevaid rakendusi ning valitakse tehnoloogiad, millega rakendusliides luuakse.

Töö teises osas püstitatakse analüüsi kohaselt nii funktsionaalsed kui ka mittefunktsionaalsed nõuded ja antakse ülevaade loodud rakenduse arhitektuurist ning valminud rakendusliidesest.

Töö tulemuseks on rakendusliides, mis vastab kõikidele diplomitöös püstitatud nõetele.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 47 leheküljel, 7 peatükki, 16 joonist, 4 tabelit.

Abstract

Development of an API for Tracking Platform for Movies and Other Media Types

The aim of this bachelor's thesis is to create an API for a media tracking platform that can be used to track films and other types of media. The purpose of the application is to provide a unified system where users can track both films, TV series, and books, as well as offering socialization opportunities through a forum.

The first part of the thesis compares existing similar applications and selects the technologies to create the API. The second part establishes, based on the analysis, both functional and non-functional requirements and provides an overview of the architecture of the created application and the resulting API.

The result of this thesis is an API that meets all the requirements set out in the thesis.

The thesis is in the Estonian language and contains 47 pages of text, 7 chapters, 16 figures, 4 tables.

Lühendite ja mõistete sõnastik

API	Rakendusliides ehk *Application Programming Interface*
Anime	Jaapani päritolu animatsioon
Beetaversioon	Tarkvara varajane versioon, mida kasutatakse testimiseks ja tagasiside hankimiseks
CLI	Käsurida
CSS	*Cascading Style Sheets* – veebilehe kujundamiseks mõeldud keel
DB	Andmebaas ehk *Database*
HTTP	*Hypertext Transfer Protocol* – veebilehtede andmeedastusprotokoll
HS256	Krüptograafiline algoritm, mida tihti kasutatakse JWT genereerimisel
HTML	*Hypertext Markup Language* – veebilehe struktuuri määratlemise keel
JWT	*JSON Web Token – levinud turvalise teabevahetust tagav standard*
JSON	*JavaScript Object Notation* – tekstipõhine andmevorming
JVM	*Java Virtual Machine* – tarkvara mõeldud Java koodi käivitamiseks
ORM	Objektrelatsiooniline kaardistamine
REST	*Representational State Transfer* – veebiteenuste disainimise ja loomise standard
Repository	Andmete haldamise komponent
Service layer	Tarkvara arhitektuuri kiht, mis vastutab äriloogika eest
Middleware	REST API komponent, mis töötleb päringuid ja vastuseid enne, kui need jõuavad rakenduse loogikasse
YAML	Inimloetav andmevorm

Sisukord

1 Sissejuhatus 10
2 Ülevaade probleemist 11
2.1 Eksisteerivad lahendused 11
2.1.1 AniList 12
2.1.2 TV Time 13
2.1.3 Flox 14
2.1.4 Goodreads 15
2.2 Eksisteerivate lahenduste võrdlus 16
3 Loodava rakenduse analüüs 18
3.1 Funktsionaalsed nõuded 18
3.2 Mittefunktsionaalsed nõuded 19
3.3 Tehnoloogia valik 20
3.3.1 Potentsiaalsed programmeerimiskeeled 20
3.3.2 Populaarsed raamistikud rakendusliidese arendamiseks 22
3.3.3 Andmebaas 24
3.4 Arenduskeskkonna valik 24
3.4.1 Koodihaldussüsteem 25
3.4.2 Koodi arenduskeskkond 25
3.5 Arhitektuur 26
3.5.1 Rakendusliidese arhitektuur 26
3.5.2 API kiht 27
3.5.3 Teenuskiht 28
3.5.4 Andmebaasi kiht 29
3.5.5 Andmebaasi disain 30
3.6 Analüüsi kokkuvõte 31
4 Arendus 32
4.1 Andmebaas ja selle mudel 32
4.2 Rakendusliides 33
4.2.1 REST API 34

4.2.2 Marsruudid 35
4.2.3 Vahevara 36
4.2.4 Turvalisus 37
4.2.5 Valideerimine 38
4.2.6 Dokumentatsioon 39
5 Testimine 40
6 Rakenduse edasiarendused 41
7 Kokkuvõte 42
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 47

Jooniste loetelu

Joonis 1 AniListi kodulehekülg 12
Joonis 2 AniListi vaade sisselogitud kasutajana 13
Joonis 3 TV Time kodulehekülg 14
Joonis 4 TV Time vaade sisselogitud kasutajana 14
Joonis 5 Floxi kodulehekülg 15
Joonis 6 Goodreads kodulehekülg 16
Joonis 7. Rakendusliidese kihid 26
Joonis 8. API kihi komponentide vaheline suhe 27
Joonis 9. Teenuskihi komponentide vaheline suhe 28
Joonis 10. Andmebaasi kihi komponentide suhe 29
Joonis 11. Olemi-suhte diagramm 30
Joonis 12 Lihtsa serveri kood kasutades Gin raamistikku 34
Joonis 13 Kasutaja valideerimise vahevara 37
Joonis 14 JWT-i genereerimine 38
Joonis 15 Valideerimisreeglite definitsioon kasutaja loomise struktuuri näitel 39
Joonis 16. Automaatselt genereeritud dokumentatsiooni ekraanitõmmis 39

Tabelite loetelu

Tabel 1. Eksisteerivate lahenduste funktsionaalsuse võrdlus 16
Tabel 2. Programmeerimiskeelte võrdlus 22
Tabel 3. Raamistikke võrdlus 24
Tabel 4. API päringud loodud rakendusliideses 35

1 Sissejuhatus

Meelelahutuse ajalugu ulatub tagasi aegade hämarusse ning on olnud oluline osa inimkonna elust. Varasemad meelelahutusvormid hõlmasid jahipidamist, tantsimist, laulmist ja jutuvestmist. Hiljem on meelelahutus arenenud koos tehnoloogia edenemisega ning inimeste vajadustega. Tänapäeval hõlmab meelelahutus erinevaid meediatüüpe, nagu filmid, telesaated, raamatud, videomängud, muusika ja palju muud.

Selleks, et jälgida, millises peatükis raamat või seriaal pooleli jäi, on kasutatud erinevaid meetodeid. Alguses kasutati raamatuhoidjaid raamatute jaoks või kirjutati episoodi number märkmikusse. Tehnoloogia arenguga hakkasid inimesed lugema või kuulama raamatuid nutiseadmetes. Sama kehtib ka seriaalide ja filmide kohta, ning paljud neist pakuvad automaatset progressi jälgimist. Siiski on nende probleem selles, et need lahendused töötavad ainult nende enda rakendustes ja ei paku selle informatsiooni jagamist ning sageli antud rakenduste kasutamine on tasuline.

Alternatiiviks tasulistele automaatsetele jälgimissüsteemidele nagu Netflix, Prime Video ja teised, on tasuta ja lihtsamad süsteemid, kus kasutaja ise haldab oma jälgimist. Seda saab teha otsides raamatuid või sarju antud teenuses ning seejärel lisades need oma jälgimislisti, kuid sellised süsteemid nõuavad kasutajalt rohkem aktiivsust kui automaatsed. Samuti erinevad ka erinevad süsteemid meediatüüpide toega: mõned pakuvad jälgimist ainult filmidele ja sarjadele, teised ainult raamatutele.

Käesolev lõputöö analüüsib olemasolevaid lahendusi, kus kasutaja ise haldab oma jälgimist, ning pakub välja püstitatud probleemile lahenduse rakendusliidese näol. Loodud rakendusliides püüab tegeleda kõikide süsteemide puudujääkidega.

2 Ülevaade probleemist

Tihti peale on raske meelde jätta, mis episoodi juurde jäetakse vaadates mõnda pikka sarja või kui kaugele jõuti raamatu läbilugemisega. Selle jälgimiseks peab kasutama mitmeid eraldi olevaid rakendusi, mis erinevad funktsionaalsuses ning ka selles, milliseid meediatüüpe saab nendes jälgida. Ühtset süsteemi, kus saab mugavalt jälgida nii filme, sarju kui ka raamatuid ühes kohas, ei ole.

Ideaalis, võiks olla süsteem, kus saab mugavalt ja intuitiivselt jälgida kogu oma meedia tarbimist ühes kohas, kus on ka foorum, arvustuste kirjutamise võimalus, ajajooned kasutajate aktiivsuse jagamiseks ning kasutajate jälgimise funktsionaalsus.

2.1 Eksisteerivad lahendused

Sarnaseid süsteeme on juba olemas ning selleks, et tekkiks hea arusaam väljatöötava süsteemi nõuetest on vajalik neid võrrelda.

Võrdlusesse on võetud järgmised rakendused:

- AniList
- TV Time
- Flox
- GoodReads

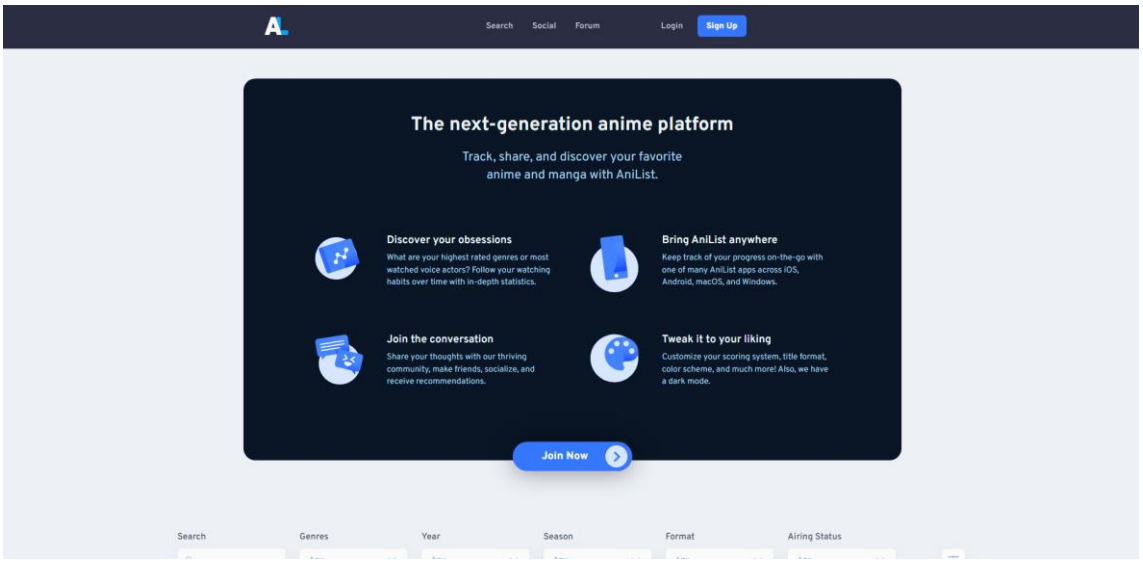
Valik oli tehtud lähtudes autori eelnevast kogemusest nende süsteemidega ning ka sõprade poolt soovitustest.

2.1.1 AniList

AniList on veebikeskkond suunatud *anime*[1] ja *manga*[2] jälgimisele [1]. Esmane versioon sellest teenusest tuli välja beetaversioonina[3] 2013. aasta septembris [2]. Antud lahendus osaliselt lahendab püstitatud probleemi. Veebilehel on võimalik jälgida ning hinnata nii *anime*t kui ka *manga*t, kuid selle jälgimisfunktsionaalsus piirdub vaid nende jälgimisega.

AniListi suurim positiivne külg on selle kommuun. Foorum, arvustuste kirjutamise ning kasutajate jälgimisfunktsionaalsus lisavad kõik kommuunitundele juurde. Lisaks on AniListis ka ajajooned kasutajate aktiivsuse jagamiseks.

Joonistel 1 ja 2 on ekraanitõmmised AniListi koduleheküljest nii välja- kui ka sisselogitud kasutajaga.

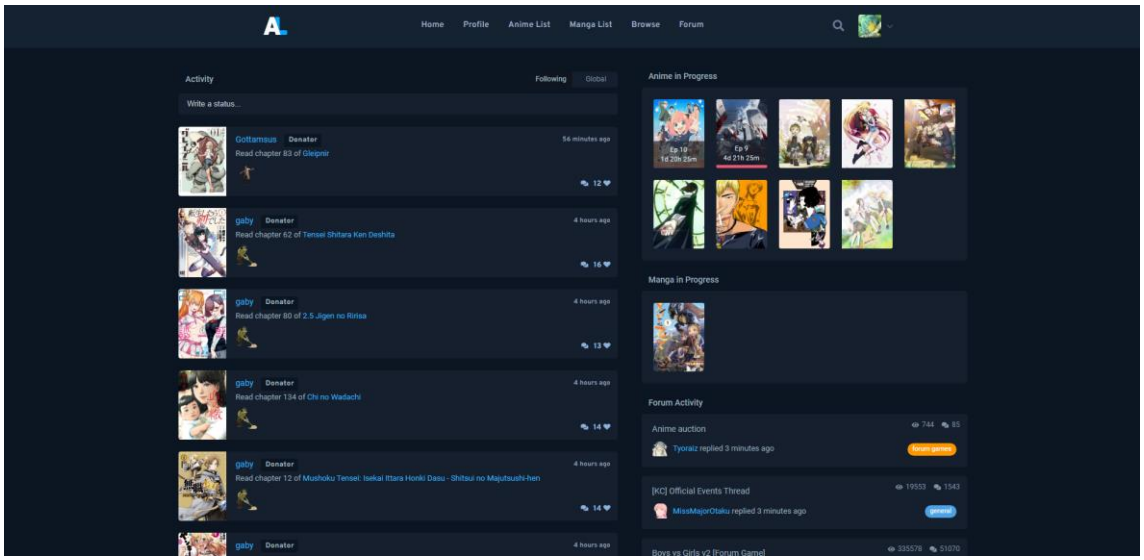


Joonis 1 AniListi koduleheküljg

[1] Anime – Jaapanist pärinevad animatsioonifilmid või -sarjad

[2] Manga – Jaapanist pärinevad koomiksid

[3] Beetaversioon – Tarkvara prooviversioon, mille käigus üritatakse leida probleeme, mis jäid arenduse või testimise käigus leidmata



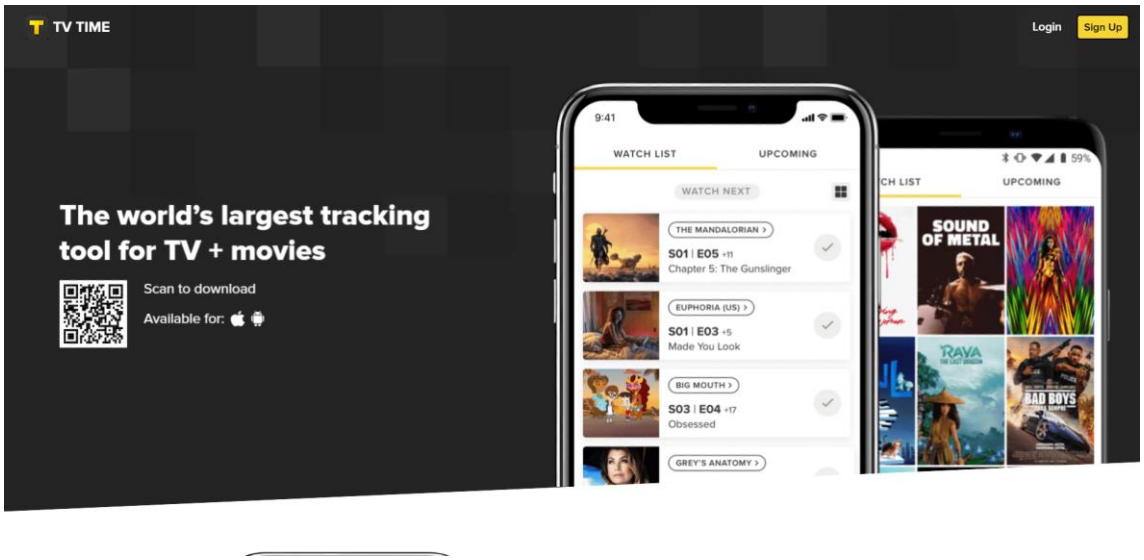
Joonis 2 AniListi vaade sisselogitud kasutajana

2.1.2 TV Time

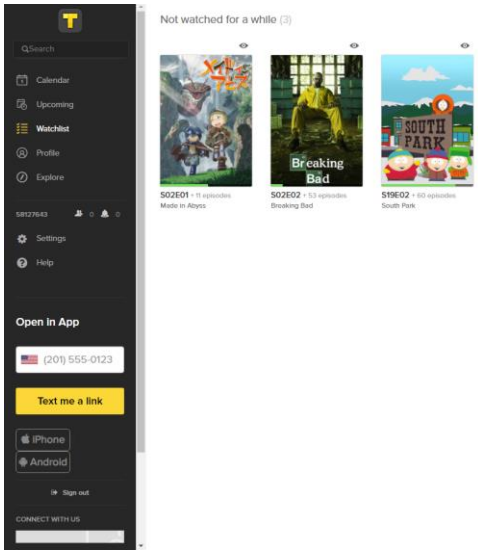
TV Time on veebikeskkond suunatud nii filmide kui ka sarjade jälgimiseks. Teenust saab kasutada nii veebilehekülje kui ka allalaaditava rakenduse näol *iOS* ning *Android* operatsioonisüsteemidel [3]. Antud rakenduses on võimalik jälgida nii filme kui ka sarju, mille hulka kuulub ka anime, kuid raamatute jälgimine ei ole võimalik.

TV Time negatiivseks küljeks on selle kommuunitunde puudus. Iga teose ning episoodi leheküljel asub kommentaarisektsioon, kus kasutajad saavad kommenteerida antud teost. Arvan, et sellist kommentaarisektsiooni ei saa võrrelda eraldi oleva arvustuste kirjutamise sektsiooniga. TV Timeil ei ole ka foorumit ning ajajooned asuvad ainult kasutajate profiilil, kus saab näha ainult nende postitatud kommentaare.

Joonisel 3 ja 4 on ekraanitõmmised TV Time kodulehest nii välja- kui ka sisselogitud kasutajaga.



Joonis 3 TV Time kodulehekülg



Joonis 4 TV Time vaade sisselogitud kasutajana

2.1.3 Flox

Flox on ise hostitud filmide, seriaalide ja animede jälgimisele suunatud veebirakendus. Kuna tegemist on ise hostitava rakendusega, ei ole see võrgus serveeritud ning selle serveerimine on kasutaja enda teha [4].

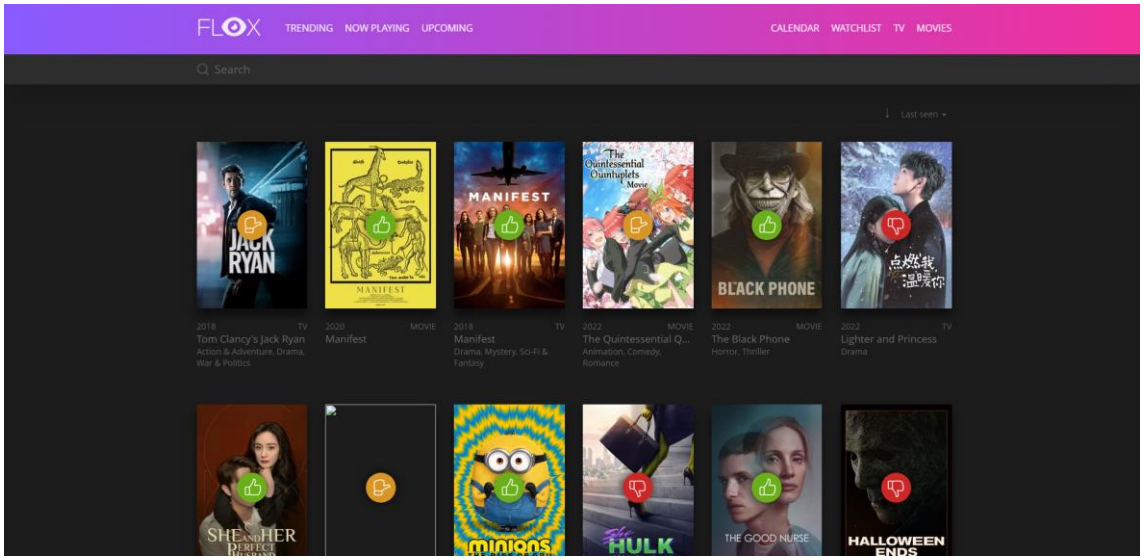
Floxi funktsionaalsusesse kuulub, kuid ei piirdu nendega:

- Teoste jälgimine episooditäpsusega ning nender hindamine
- Soovitused

- Kalender uute teoste esilinastuskuupäevadega [4].

Floxi puhul ei saagi üldse rääkida kommuunitundest, kuna tegemist on siiski ise hostitava rakendusega. Seega, sellel puudub arvustuste kirjutamine, kommenteerimine, ajajooned ja kõik muu funktsionaalsus, mis on kuidagi seotud kasutajate sotsialiseerimisega.

Joonisel 5 on ekraanitõmmis Flox koduleheküljest.



Joonis 5 Floxi koduleheküle

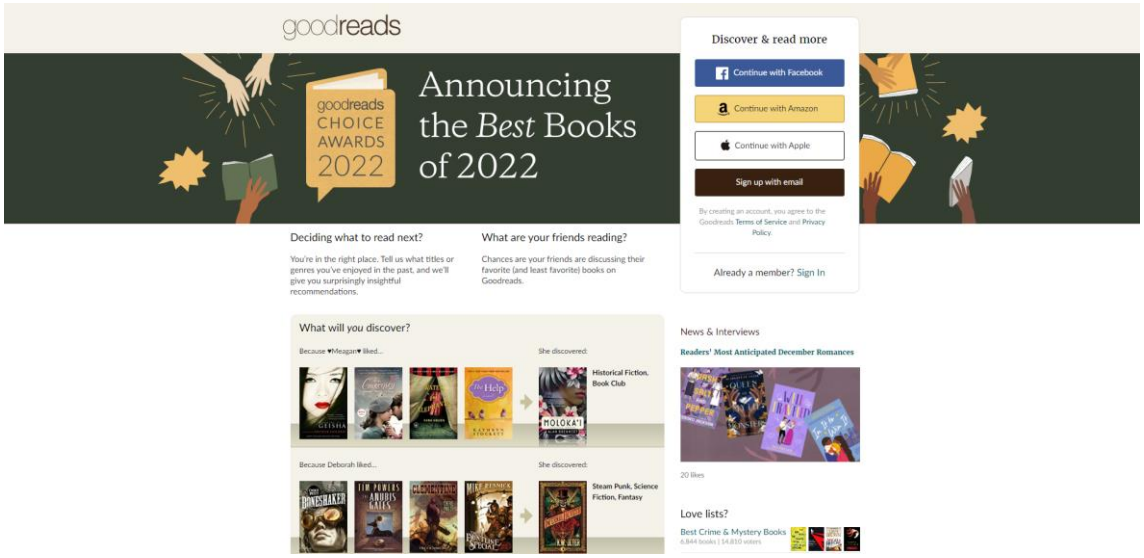
2.1.4 Goodreads

Goodreads on raamatute jälgimisele suunatud veebileht. Tegemist on suurima veebilehega suunatud just raamatute jälgimisele ning soovitustele. Antud teenus käivitati 2007. aastal [5].

Goodreadsis saab jälgida raamatuid, vaadata milliseid raamatuid teised kasutajad loevad, saada soovitusi raamatute kohta ning nii kirjutada kui ka lugeda arvustusi teoste kohta [5]. Lisaks sellele, saab ka raamatuid hinnata.

GoodReadsis on hea kommuuniga seotud funktsionaalsus. Sellel on grupid, mis töötavad nagu väiksemad kasutajate enda poolt haldavad foorumid. Grupides saavad kasutajad jagada videosid, pilte ning arutada teiste kasutajatega raamatute või teistel teemadel.

Joonisel 6 on ekraanitõmmis Goodreads koduleheküljest.



Joonis 6 Goodreads koduleheküIg

2.2 Eksisteerivate lahenduste võrdlus

Eksisteerivate lahenduste võrdluseks koostas autor tabeli koos kõikide vaadeldud rakendustega ning loodava rakenduse soovitud funktsionaalsustega (Tabel 1). Rohelisega on märgitud rahuldav, kollasega puudulik ning punasega mitterahuldav funktsionaalsus.

Tabel 1. Eksisteerivate lahenduste funktsionaalsuse võrdlus

	AniList	TV Time	Flox	Goodreads
Filmide jälgimine	Ainult anime			
Sarjade jälgimine	Ainult anime			
Raamatute jälgimine	Ainult manga			
Foorumi olemasolu				
Kasutajate jälgimine				
Ajajooned kasutajate aktiivsuse jagamiseks		Kasutajate profiilil vaid nende isiklik aktiivsus		

Arvustuste kirjutamine				
Profiili privaatsuse sätted				

Tabelist on näha, et AniList on kõige lähedasem loodava rakenduse funktsionaalsusele: puudulikuks jäävad vaid filmide, sarjade ning raamatute jälgimise funktsionaalsus. Goodreads on samuti lähedal, kuid selle jälgimisfunktsionaalsus piirdub raamatutega.

3 Loodava rakenduse analüüs

Nõuded on kirjapandud kasutajaloode kujul, kus põhirollideks on tavakasutaja, moderaator ning administraator.

Nõuded, mis jäävad väljaspoole antud lõputöö skoobi käsitletatakse lõputöö lõpus.

3.1 Funktsionaalsed nõuded

Tavakasutaja funktsionaalsed nõuded:

- Tavakasutajana soovin ennast registreerida.
- Tavakasutajana soovin logida sisse.
- Tavakasutajana soovin enda profiili muuta.
- Tavakasutajana soovin lisada teoseid enda teoste listi.
- Tavakasutajana soovin hinnata teoseid.
- Tavakasutajana soovin lühidalt kommenteerida teoseid.
- Tavakasutajana soovin avaldada oma arvamust teose kohta arvustuse kujul.
- Tavakasutajana soovin jälgida teisi kasutajaid.
- Tavakasutajana soovin näha kogu oma vaadatud ja/või loetud teoste ajalugu.
- Tavakasutajana soovin osaleda foorumi diskussioonides.
- Tavakasutajana soovin avada uusi foorumi diskussioone.

Moderaatori funktsionaalsed nõuded:

- Moderaatorina soovin kustutada tavakasutajate postitusi reeglirikkumiste korral.
- Moderaatorina soovin deaktiveerida tavakasutajate kontosid korduvate reeglite rikkujate korral.
- Moderaatorina soovin lisada uusi teoseid.

- Moderaatorina soovin redigeerida teoseid.

Administraatori funktsionaalsed nõuded:

- Administraatorina soovin hallata kasutajate rolle.

3.2 Mittefunktsionaalsed nõuded

Tavakasutaja mittefunktsionaalsed nõuded:

- Tavakasutajana soovin, et minu andmed oleksid turvalised.

Tarkvaraarendaja mittefunktsionaalsed nõuded:

- Tarkvaraarendajana soovin suhelda rakendusliidesega HTTP päringutega.
- Tarkvaraarendajana soovin saada andmeid JSON-i kujul.

Administraatori mittefunktsionaalsed nõuded:

- Administraatorina soovin, et koodibaas oleks hallatav läbi versioonihaldussüsteemi tsentraalses repositooriumis.

3.3 Tehnoloogia valik

Rakendusliidese arendamiseks on võimalik kasutada väga paljusid tehnoloogiaid: erinevad programmeerimiskeeled, raamistikud, andmebaasid jne. Antud lõputöö raames vaadeldakse tehnoloogiaid, millega autor on eelnevalt kokku puutunud.

3.3.1 Potentsiaalsed programmeerimiskeeled

Programmeerimiskeeli on maailmas tohutult palju. Autor on kokku puutunud järgmiste keeltega: Python, Java, Kotlin, C#, Go, JavaScript ning TypeScript.

3.3.1.1 Python

Python on interpreteeritud, objektorienteeritud kõrgetasemeline programmeerimiskeel. Pythoni lihtne ja kergesti õpitav süntaks rõhutab loetavust ja vähendab seetõttu programmi hoolduskulusid. Python toetab mooduleid ja pakette, mis soodustab programmide modulaarsust ja koodi taaskasutamist. Selle ulatuslik standardteek on kõikidel suurematel platvormidel tasuta saadaval lähtekoodina või kompileeritud programmina [6].

3.3.1.2 Java

Java on programmeerimiskeel, mis arenes tagasihoidlikust algusest, et toita suur osa tänapäeva digitaalmaailmast, pakkudes usaldusväärset platvormi, millele on üles ehitatud paljud teenused ja rakendused [7].

Tegemist on eelkõige interpriteeritud objektorienteeritud programmeerimiskeelega.

3.3.1.3 Kotlin

Kotlin on objektorienteeritud programmeerimiskeel, mis on koostalitlusvõimeline Java virtuaalmasina, Java teekide ja Androidiga [8].

Kotlinit peetakse Java asendajaks. Kuigi see ei ühildu süntaksiga, on see koostalitlusvõimeline Java koodi ja raamistikkudega. On olemas ka raamistikke, mis on eelkõige mõeldud Kotlini jaoks. Algselt kasutati Kotlinit Androidi rakenduste arendamiseks, kuid tänapäeval kasutatakse seda ka näiteks serverrakenduste ja veebirakenduste arendamiseks [8].

3.3.1.4 C#

C# on objektorienteeritud keel, mis võimaldab arendajatel luua mitut tüüpi turvalisi ja töökindlaid rakendusi, mis töötavad .NET-is. C# juured pärinevad C-keelte perekonnast ja on C, C++, Java ja JavaScripti programmeerijatele kohe tuttavad [9].

3.3.1.5 Go

Go, tuntud ka kui Golang, on avatud lähtekoodiga, kompileeritav ja staatilise tüübikontrolliga programmeerimiskeel, mille on väljatöötanud Google enda koodibaasi lihtsustamise eesmärgi saavutamiseks. Go kiiresti kasvas populaarsuses ja sai oma lihtsuse, loetavuse ning tõhususe tõttu paljude arendajate lemmikuks [10].

3.3.1.6 JavaScript ja TypeScript

JavaScript ja TypeScript on väga sarnased keeled.

JavaScript on scriptimis- ja programmerimiskeel, mis võimaldab implementeerida keerukaid funktsionaalsusi veebilehtedel. See funktsionaalsus võib olla näiteks õigeaegne sisuvärskendus, interaktiivsed kaardid, animeeritud 2D- või 3D-graafika jne. See on üks kolmest standardsete veebitehnoloogiate kihtidest, millest teised on CSS ja HTML [11].

TypeScript on programmeerimiskeel, mis põhineb JavaScriptil, pakkudes täiendavat süntaksi, et toetada tihedamat integreerimist koodiredaktoritega, mis omakorda aitab leida vigu varakult ülesse. TypeScripti kood teiseneb JavaScriptiks, mis töötab kõikjal, kus JavaScript töötab [12].

Nii JavaScripti kui ka TypeScripti saab kasutada ka serveripoolsetes rakendustes kasutades näiteks Node.js käitussüsteemi [13].

3.3.1.7 Programmeerimiskeelte võrdlus

Järgnevalt on toodud autori poolt koostatud tabel (Tabel 2), et võrrelda potentsiaalseid programmeerimiskeeli kogemuse ja keerukuse poolest. Kogemuse all mõeldakse kui palju on autor antud keelega kokku puutunud.

Tabel 2. Programmeerimiskeelte võrdlus

Keel	**Kogemus**	**Keerukus**
Python	Vähene	Madal [14]
Java	Suur	Keskmine [14]
Kotlin	Keskmine	Keskmine [15]
C#	Keskmine	Keskmine [15]
Go	Keskmine	Madal [14]
JavaScript	Vähene	Madal [15]
TypeScript	Keskmine	Keskmine [15]

3.3.2 Populaarsed raamistikud rakendusliidese arendamiseks

Autor arvab, et programmeerimiskeele valikul on oluline ka vaadelda erinevaid raamistikke.

Django on raamistik, mis on mõeldud veebirakenduste loomiseks, kasutades programmeerimiskeelt Python. Selle eesmärk on aidata arendajatel arendada rakendusi kontseptsioonist valmisproduktini võimalikult lühikese aja jooksul [16].

Spring on raamistik, mis on suunatud eelkõige Java-põhistele rakendustele, mis pakub programmeerimis- ja konfiguratsioonimudelit äri-loogika implementeerimiseks [17]. Springi saab kasutada ka teiste programmeerimiskeeltega, mis on JVM-platvormil ehitatud, nagu näiteks Kotlin ja Groovy [18].

Ktor on raamistik, mis on mõeldud eelkõige asünkroonsete rakenduste jaoks, mis on ehitatud Kotlini keele abil, mis võimaldab ehitada veebirakendusi, HTTP teenuseid ja

mobiilirakendusi. Selle eesmärk on pakkuda täielikku mitmeplatvormilist raamistikku ühendatud rakenduste jaoks [19].

Micronaut on kaasaegne, JVM-põhine raamistik, mis on mõeldud eelkõige mikroteenuste arendamiseks. Selle raamistiku eripäraks on lühike käivitusaeg ja väike mälu tarbimine hoolimata projekti suurusest ning väga lihtne testitavus [20].

ASP.NET on kaasaegne raamistik, mis on loodud veebirakenduste arendamiseks programmeerimiskeele C# abil. See raamistik pakub võimalusi ehitada rakendusliideseid ja mikroteenuseid [21].

Gin on raamistik, mille eesmärk on veebirakenduste arendamine programmeerimiskeele Go abil, mis on suunatud rakenduste jõudlusele ning arendaja produktiivsusele. Raamistiku funktsionaalsus on ka lihtsasti arendaja poolt laiendav, kuna pakub väga lihtsat viisi vahevara lisamiseks [22].

Next.js on raamistik, mis on loodud veebirakenduste arendamiseks programmeerimiskeelte JavaScript ja TypeScript abil. Kuna tegemist on JavaScripti ja TypeScriptile suunatud raamistikuga, pakub see ka mugavat kliendirakenduste arendust antud programmeerimiskeelte abil [23].

3.3.2.1 Raamistikude võrdlus

Järgnevalt on toodud autori koostatud tabel (Tabel 3), milles võrreldakse eelnevalt mainitud raamistike kogemuslikku ulatust. Raamistiku valikul tuleb arvestada raamistiku aktuaalsuse, perspektiivi ning autori enda kogemust.

Tabel 3. Raamistikke võrdlus

Raamistik	**Kogemus**
Django	Puudub
Spring	Suur
Ktor	Keskmine
Micronaut	Vähene
ASP.NET	Keskmine
Gin	Vähene
Next.js	Puudub

3.3.3 Andmebaas

Autor piiritleb andmebaasi valikuvõimalusi ainult nendega, millega tal on isiklik kogemus. Konkreetsemalt on fookuses Oracle andmebaas ja PostgreSQL.

Oracle andmebaas on äärmiselt võimas relatsiooniline andmebaasihaldussüsteem, mille funktsionaalsus on erakordselt lai. Oracle Corporation, selle loojad, pakuvad kasutajatele kulude optimeeritud ja kõrge jõudlusega versioone [24]. Siiski võib märkida, et Oracle andmebaasi üheks märgatavaks miinuseks on selle hind, sest tasuta versiooni sellest tarkvarast ei pakuta [25].

Teisalt, PostgreSQL on samuti tugev relatsiooniline andmebaasihaldussüsteem, mille tarkvara on loodud avatud lähtekoodiga. Seetõttu on selle kasutamine täiesti tasuta [26].

3.4 Arenduskeskkonna valik

Arenduskeskkonna valikul peab silmas pidada koodihaldust ning koodi arenduskeskkonda.

3.4.1 Koodihaldussüsteem

GitHub, GitLab ja BitBucket on kõik populaarsed versioonihaldustarkvarad, mis võimaldavad arendajatel hallata ja jagada oma koodibaase. Kuigi need tarkvarad pakuvad sarnaseid funktsioone, on nende vahel mõned erinevused.

GitHub on kõige populaarsem versioonihaldustarkvara ja seda kasutatakse kogu maailmas laialdaselt. GitHub pakub avatud lähtekoodiga tarkvara ja on tasuta kasutamiseks nii avalike kui ka privaatsete projektide jaoks. Samuti on GitHubil lai kogukond, mis tähendab, et dokumentatsiooni ja abi leidmine ei osutu probleemiks. [27].

GitLab pakub ka avatud lähtekoodiga tarkvara, kuid erinevalt GitHubist on GitLab võimalik installida oma serveritesse. See tähendab, et GitLab võib olla privaatsete projektide jaoks parem valik, sest see annab arendajatele rohkem kontrolli oma koodibaasi üle [28].

BitBucket on Atlassiani pakutav versioonihaldustarkvara, mis on mõeldud eelkõige ettevõtetele. BitBucket pakub tasuta privaatsete projektide jaoks kuni viie kasutaja jaoks ning seda saab integreerida ka teiste Atlassiani toodetega, nagu JIRA ja Confluence [29].

3.4.2 Koodi arenduskeskkond

Koodi arenduskeskkonna puhul ei ole valik ka üsna keeruline. Valikus on kas Visual Studio Code või mõni Jetbrains’i integreeritud arenduskeskkond ehk IDE.

Visual Studio Code ja Jetbrains IDE-de erinevused seisnevad peamiselt nende eesmärkides ja funktsioonides. Visual Studio Code on kergema kaaluga ja selle eesmärk on pakkuda arendajatele suurepärast tekstiredaktorit, mis on kohandatav ja integreeritav paljude erinevate programmeerimiskeeledega [30]. Jetbrains IDE-d seevastu on spetsialiseerunud ühele või mitmele kindlale programmeerimiskeelele ning pakuvad rohkem funktsioone ja tööriistu konkreetsete töövoogude ja probleemide lahendamiseks [31].

3.5 Arhitektuur

Arhitektuurilise lahenduse valikuks tuleb arvesse võtta rakendusliidese eeldatava suurust ning keerukust. Kuna projekt on üldiselt väike, autor otsustas mitte vaadelda mikroteenuse lahendusi ning hoopis pöörduda monoliitse arhitektuuriga lahenduse poole.

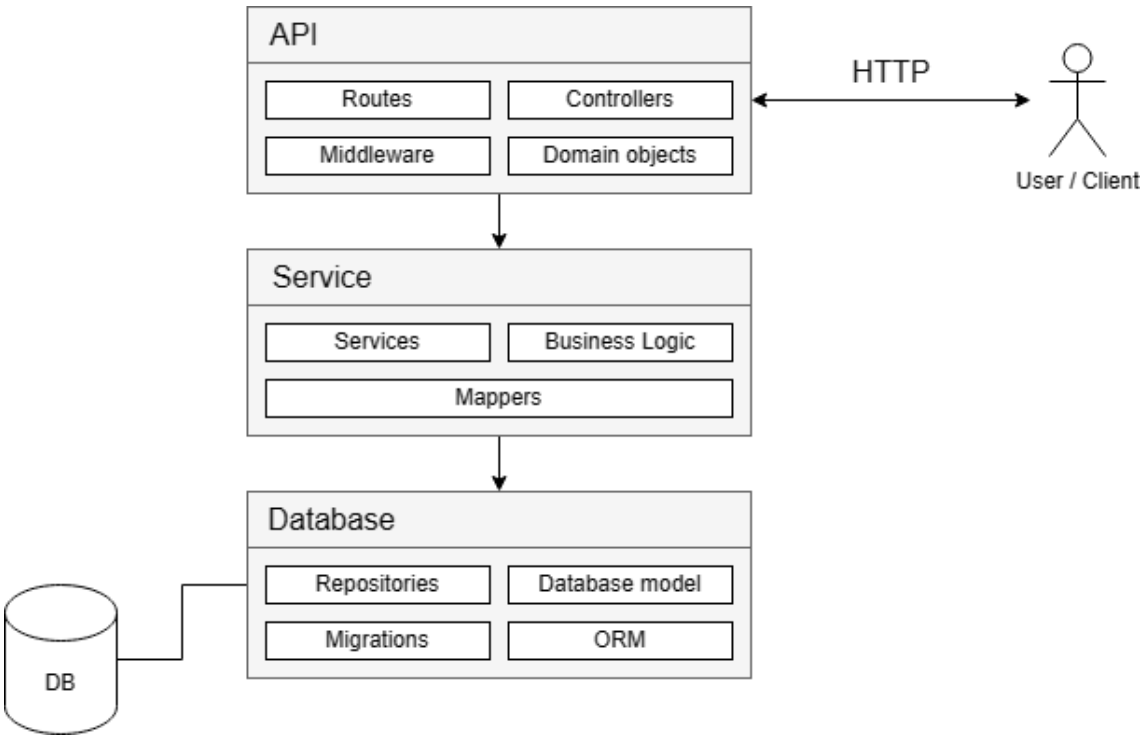
3.5.1 Rakendusliidese arhitektuur

Antud töö raames jagatakse rakendusliides kolmeks kihiks:

1. API kiht
2. Teenuskiht
3. Andmebaasi kiht

Need kihid töötavad koos, et pakkuda liidest kasutajatele süsteemiga suhtlemiseks.

Joonis 7 näitab eelnevalt nimetatud komponentide omavahelist suhet:



Joonis 7. Rakendusliidese kihid

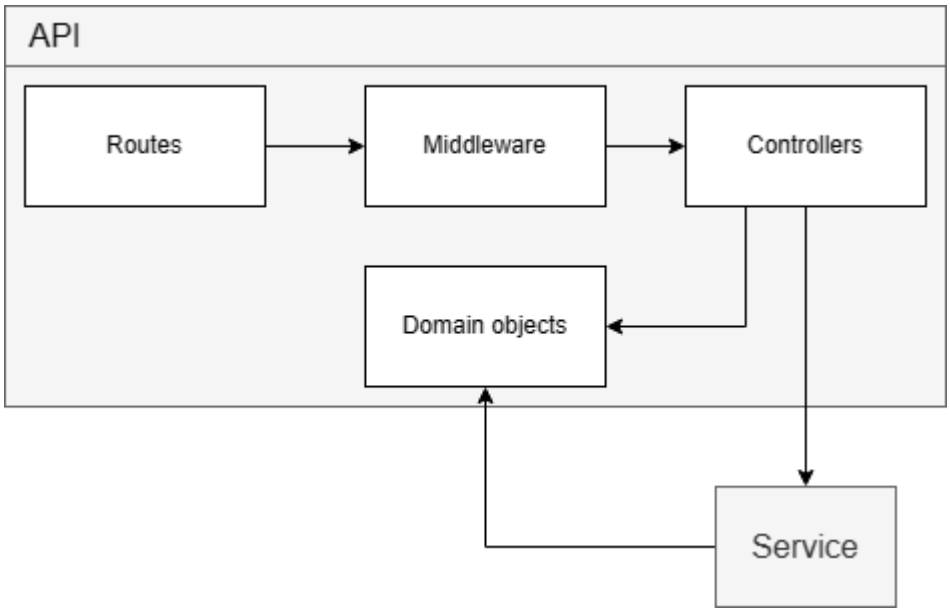
3.5.2 API kiht

Rakenduse API kiht vastutab andmete edastamise eest klientide ja serveri vahel, järgides REST API põhimõtteid, mille kohaselt kasutatakse HTTP meetodeid nagu GET, POST, PUT ja DELETE andmevahetuse tagamiseks.

Antud kiht vastutab ka route'ide ehk marsruutide konfigureerimise eest. Iga marsruut kasutab ka middleware ehk vahevarasid, mille abil täidetakse erinevaid ülesandeid, nagu autentimine ja logimine, tagamaks vastava marsruudi ärireeglite rahuldamise. Kui vahevara käivitamisel tekib probleeme, näiteks autentimine ebaõnnestub, katkestatakse protsess varakult ning kliendile antakse sellest teada.

Pärast vahevara edukat käivitamist edastatakse päring vastavale kontrollerile, kes vastutab päringu andmete kogumise ja teenuskihiga suhtlemise eest.

Joonis 8 näitab API kihi komponentide vahelist suhet.



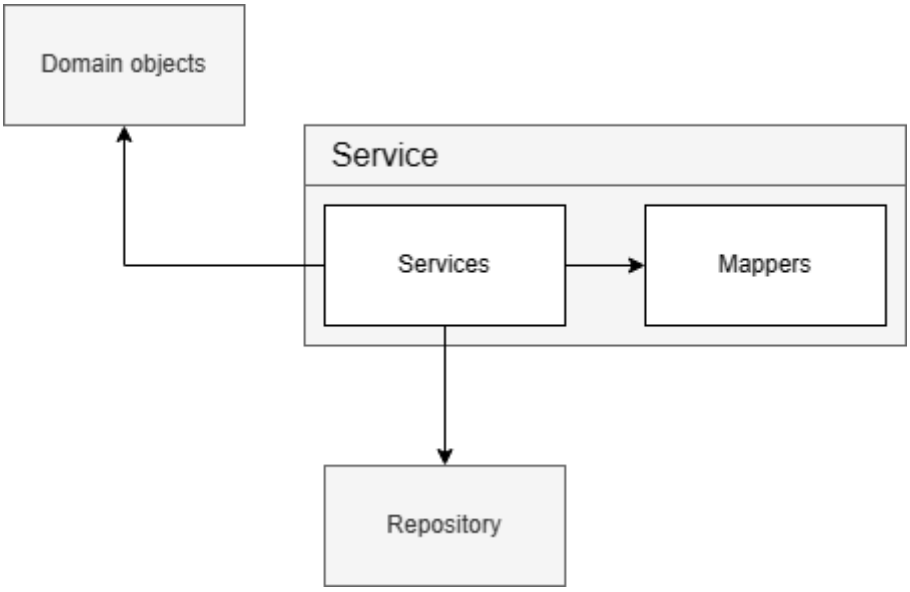
Joonis 8. API kihi komponentide vaheline suhe

3.5.3 Teenuskiht

Teenuskiht ehk *service layer* on vahenduskiht rakenduse domeenimudeli ja andmebaasi kihi vahel. Teenuskiht pakub domeenimudelile spetsiifilist funktsionaalsust, mis on vajalik rakenduse äriloogika rakendamiseks [32].

Teenuskiht tegeleb ka andmebaasi mudeli objektide teisendamisega domeenobjektideks.

Joonis 9 näitab teenuskihi komponentide vahelist suhet.



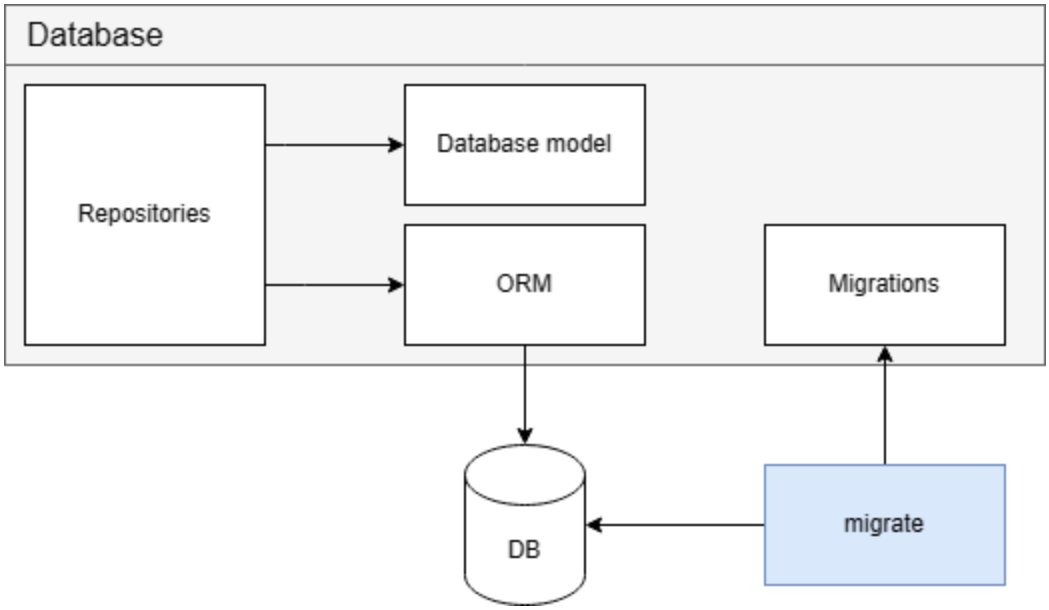
Joonis 9. Teenuskihi komponentide vaheline suhe

3.5.4 Andmebaasi kiht

Andmebaasi kiht vastutab andmebaasiühenduse ja andmete salvestamise eest, mille kaudu saab rakendus andmeid salvestada, luigeda, uuendada ja kustutada. Antud kiht kasutab *repository* pattern'it ehk hoidla tarkvaraarenduse mustrit, mis võimaldab rakendusel andmeid salvestada ja lugeda ilma andmebaasi spetsiifilisi funktsioone kasutamata, mistõttu võib andmebaasi muutmine olla lihtsam [33].

Andmebaasiga suhtlemist hõlbsustab ka kasutusel olev ORM[1], mis on automaatselt genereeritud andmebaasi mudeli alusel.

Joonis 10 näitab andmebaasi kihi komponentide vahelist suhet.

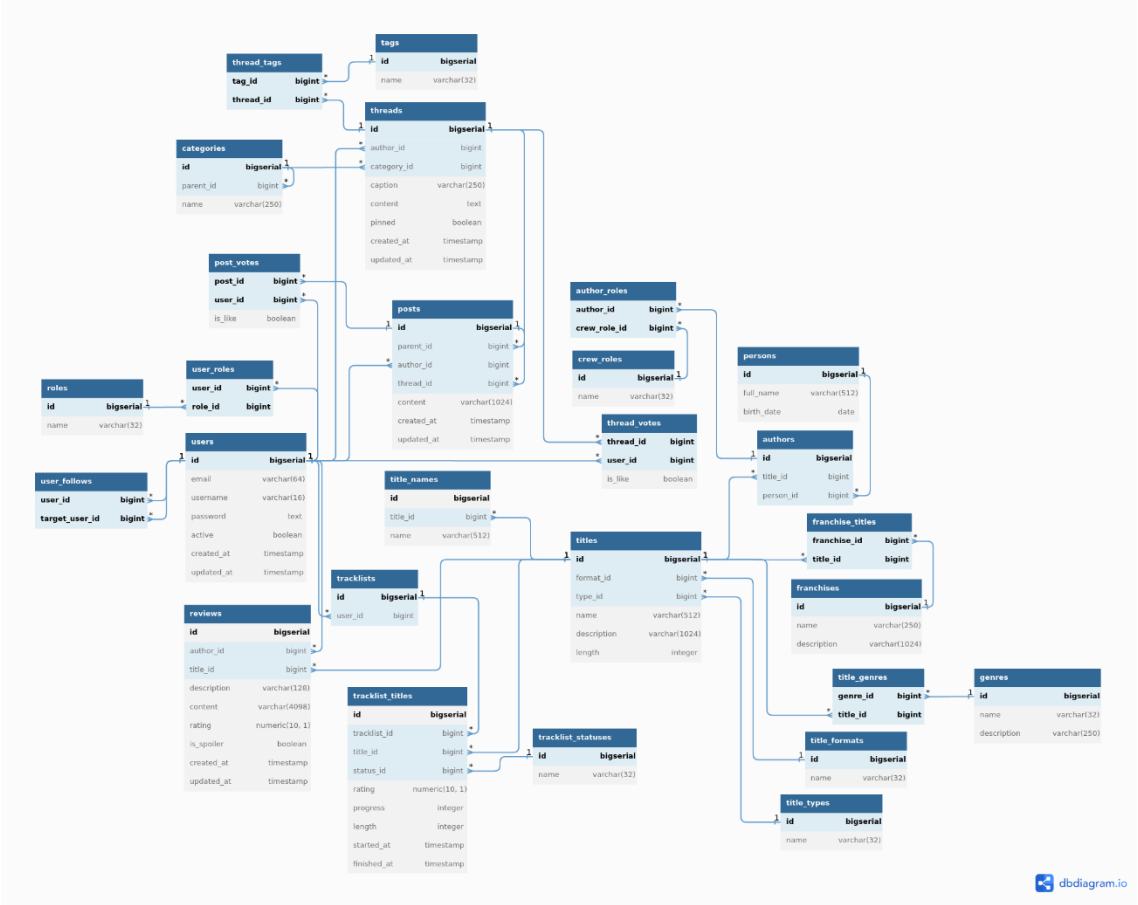


Joonis 10. Andmebaasi kihi komponentide suhe

[1] ORM – objekt-relatsiooniline kaardistamine.

3.5.5 Andmebaasi disain

Funktsionaalsete ja mittefunktsionaalsete nõuete kohaselt loos antud süsteemi olemi-suhte diagramm ja seega, andmebaasi mudel (Joonis 11).



Joonis 11. Olemi-suhte diagramm

3.6 Analüüsi kokkuvõte

Analüüsi käigus keskenduti funktsionaalsetele ja mittefunktsionaalsetele nõuetele, arutleti erinevate kasutajagruppide üle ning kavandati andmebaasi mudel vastavalt nõuetele ning valiti sobiv rakendusliidese arhitektuur.

Programmeerimiskeele valiku puhul eelistati Go-d, hoolimata eelnevast Java kogemusest ning seetõttu otsustati kasutada ka Gin raamistikku.

Andmebaasi valik oli otsustatud lihtsalt: kuna Oracle'i kasutamiseks on vaja litsentsi, valiti PostgreSQL. Lisaks pakub PostgreSQL sarnast funktsionaalsust ning autor leiab, et selle kasutamine on palju lihtsam.

Koodi arenduskeskkonnaks valiti Jetbrains’i IDE nimega Goland, mis on spetsiifiliselt arendatud Go tarkvaraarenduse eesmärgiks ning koodihaldussüsteemiks valiti GitHub, kuna autoril on eelnev kogemus GitHubi kasutamisega.

4 Arendus

Arenduse peatükis käsitletakse kogu tehtud arendustöö. Arendusprotsess on jaotatud kolmeks:

1. Andmebaas ja selle mudel
2. Rakendusliides
3. Testimine.

4.1 Andmebaas ja selle mudel

Analüüsi tulemusena valiti PostgreSQL andmebaas süsteemi alusena. Andmebaasi käivitamiseks otsustas töö autor kasutada Dockeri tehnoloogiat, mis võimaldab hõlpsasti luua ja käivitada selle rakenduse. Selleks, et hõlbustada Dockeri konteineri seadistamist, kasutati Docker Compose tööriista, mis võimaldab kasutada YAML konfiguratsioonifaili Dockeri konteineri seadistamiseks.

Andmebaasi mudeli loomiseks kasutas autor dbdiagram tööriista, mis on tasuta ja lihtne tööriist andmebaasi mudeli ja diagrammide loomiseks, kasutades DBML[1] keelt.

Migratsioonide haldamiseks valis töö autor tööriista nimega „migrate“, mis on kirjutatud Go programmeerimiskeeles ja mõeldud spetsiaalselt Go programmeerimiskeele rakenduste jaoks. Seda saab kasutada nii CLI[2] kui ka otse Go teegina. Käesoleva töö raames kasutatakse seda tööriista CLI kujul [34].

Andmebaasiga suhtlemiseks on kasutusele võetud ka ORM, mis on automaatselt genereeritud andmebaasi mudeli järgi tööriista SQLBoiler abil. Lisaks ORMile, genereerimise tulemuseks on ka ORMi ühilduvustestid andmebaasiga [35].

[1] DBML ehk Database Markup Language – lihtne andmebaasi märgistuskeel, mis on loodud andmebaasistruktuuride määratlemiseks [41].

[2] CLI – käsurealiides

4.2 Rakendusliides

Antud peatükis käsitletakse eelnevalt välja toodud arhitektuuri disaini elluviimist kasutades programmeerimis keelt Go, Gin raamistikku ning muid teeke, mis hõlbustasid antud protsessi.

Rakendusliidese failid on jaotud kaustade vahel järgneval viisil:

- api – REST API-ga seotud koodibaas.
    - controller – REST API kontrollerid.
    - route – REST API marsruudid.
    - domain – domeen objektid.
    - middleware – vahevarad.
- service – äriloogikaga seotud koodibaas.
- db – andmebaasiga seotud koodibaas.
    - repository – hoidlad andmebaasiga suhtlemiseks.
    - dbmodel – andmebaasi mudel ja ORM, mis on automaatselt genereeritud kasutades sqlboiler teegi kasutusel.
    - migrations – andmebaasi migratsioonifailid.
- docs – Swagger dokumentatsioonifailid
- errors – vigadega seotud funktsionaalsus
- docker – Dockeriga seotud failid

4.2.1 REST API

REST API ehk *Representational State Transfer Application Programming Interface* on tarkvara arendusmeetod, mis põhineb HTTP protokollil. REST API-l on mitu omadust, mis muudavad selle kasutamise tõhusaks ja paindlikuks. REST API saab vastata päringutele erinevates esitlustes nagu JSON ehk *JavaScript Object Notation*, XML ehk *Extensible Markup Language* ja muud vormingud vastavalt kliendi vajadustele. Loodud rakendusliides vastab päringutele vaid JSON-i kujul [36].

Lisaks sellele kasutab REST API HTTP meetodeid nagu GET, POST, PUT, DELETE, PATCH jne, et võimaldada andmete edastamist serveri ja kliendi vahel [36].

Go standardteek on piisavalt võimeline REST API väljatöötamiseks, kuid raamistiku kasutamine võib muuta protsessi palju lihtsamaks. REST API arendamine Gin abil on väga kerge.

Joonisel 12 on koodijupp lihtsast serverist kasutades Gin raamistikku.

```
package main

import "github.com/gin-gonic/gin"

func main() {
        r := gin.Default()
        r.GET("/ping", func(c *gin.Context) {
                c.JSON(200, gin.H{
                        "message": "pong",
                })
        })
        r.Run() // listen and serve on 0.0.0.0:8080
}
```

Joonis 12 Lihtsa serveri kood kasutades Gin raamistikku

4.2.2 Marsruudid

REST API marsruudid on grupeeritud vastaval nõuetele. Tabelis 4 on kujutatud kõik loodud rakenduse marsruudid.

Tabel 4. API päringud loodud rakendusliideses

Kirjeldus	HTTP päringu tüüp	Marsruut
Autentimine		
Registreerimine	POST	*/auth/register*
Sisselogimine	POST	*/auth/login*
Kasutaja		
Kasutajate pärimine	GET	*/users*
Kasutaja pärimine	GET	*/users/:id*
Kasutajate muutmine	PUT	*/users/:id*
Kasutajate aktiveerimine	PUT	*/users/:id/activate*
Kasutajate deaktiveerimine	PUT	*/users/:id/deactivate*
Kasutajate otsing	GET	*/users/search/:username*
Foorum		
Kategooriate päring	GET	*/categories*
Alamkategooriate päring	GET	*/categories/:id*
Kategooria loomine	POST	*/categories/*
Kategooria muutmine	PUT	*/categories/:id*
Kategooria kustutamine	DELETE	*/categories/:id*
Teemade pärimine kategooria järgi	GET	*/threads/category/:id*
Teema pärimine	GET	*/threads/:id*
Teema postide pärimine	GET	*/threads/:id/posts*
Teema loomine	POST	*/threads/:id*
Teema muutmine	PUT	*/threads/:id*
Teema kustutamine	DELETE	*/threads/:id*
Teema meeldivaks märkimine	POST	*/threads/:id/upvote*
Teema ebameeldivaks märkimine	POST	*/threads/:id/downvote*
Postide pärimine	GET	*/posts*
Posti loomine	POST	*/posts*
Posti pärimine	GET	*/posts/:id*
Posti muutmine	PUT	*/posts/:id*
Posti kustutamine	DELETE	*/posts/:id*
Posti meeldivaks märkimine	POST	*/posts/:id/upvote*
Posti ebameeldivaks märkimine	POST	*/posts/:id/downvote*
Arvustused		
Arvustuste päring teose järgi	GET	*/reviews/title/:id*
Arvustuse päring	GET	*/reviews/:id*
Arvustuse loomine	POST	*/reviews*

Arvustuse muutmine	PUT	*/reviews/:id*
Arvustuse kustutamine	DELETE	*/reviews/:id*
Teosed		
Teoste pärimine	GET	*/titles*
Teose pärimine	GET	*/titles/:id*
Teoste pärimine žanri järgi	GET	*/titles/genres/:id*
Teose otsing	GET	*/titles/search/:search*
Teose loomine	POST	*/titles*
Teose muutmine	PUT	*/titles/:id*
Teose kustutamine	DELETE	*/titles/:id*
Teose žanri lisamine	POST	*/titles/:id/genres/:id*
Teose žanrite seadmine	PUT	*/titles/:id/genres*
Teose žanri eemaldamine	DELETE	*/titles/:id/genres/:id*
Frantsiiside päring	GET	*/franchises*
Frantsiisi päring	GET	*/franchises/:id*
Frantsiisi lisamine	POST	*/franchises*
Frantsiisi muutmine	PUT	*/franchises*
Frantsiisi kustutamine	DELETE	*/franchises/:id*
Teose alternatiivsete nimede seadmine	PUT	*/titles/:id/names/set*
Teose alternatiivse nime lisamine	POST	*/titles/:id/names*
Teose alternatiivse nime muutmine	PUT	*/titles/:id/names/:id*
Teose alternatiivse nime kustutamine	DELETE	*/titles/:id/names/:id*
Teoste jälgimine		
Kasutaja jälgimispäevik	GET	*/tracklist/:userid*
Teose päevikusse lisamine	POST	*/tracklist/:userid/title/:id*
Teose muutmine päevikus	PUT	*/tracklist/:userid/title/:id*
Teose kustutamine päevikust	DELETE	*/tracklist/:userid/title/:id*

Igal marsruudi grupil on oma kontroller, mida kasutatakse teenuskihiga suhtlemiseks.

4.2.3 Vahevara

Vahevara on tarkvarakomponent, mis asub erinevate tarkvara protsesside vahel ning pakub võimalusi tarkvara funktsionaalsuse laiendamiseks ja parandamiseks. Vahevarasid kasutatakse laialdaselt tarkvararakenduste arendamisel ja haldamisel. Näiteks võib veebirakenduses kasutada vahevara, et logida päringuid ja vastuseid ning autentimine [37].

Joonisel 13 on koodijupp vahevarast, mis vastutab kasutaja poolt saadetud andmete valideerimise eest.

```
func ValidateStruct[T any](v *validator.Validate) gin.HandlerFunc {
	return func(c *gin.Context) {
		var t T
		if err := c.ShouldBindJSON(&t); err != nil {
			c.Error(err)
			c.AbortWithStatusJSON(
				http.StatusBadRequest,
				response.NewErrorBody(
					http.StatusBadRequest,
					[]error{errs.BadRequest},
				),
			)
			return
		}

		if err := v.Struct(t); err != nil {
			validationErrors := err.(validator.ValidationErrors)
			c.Error(validationErrors)
			c.AbortWithStatusJSON(
				http.StatusBadRequest,
				response.NewErrorBody(
					http.StatusBadRequest,
					[]error{validationErrors},
				),
			)
			return
		}

		c.Set("struct", t)
	}
}
```

Joonis 13 Kasutaja valideerimise vahevara

Antud projekti raames kasutatakse vahevarasid pärast igat sissetulevat HTTP päringut vigade logimiseks, autentimiseks, rollide tuvastamiseks jne.

4.2.4 Turvalisus

Turvalisuse tagamiseks on kasutusele võetud JWT (JSON Web Token) ehk JSON-põhine veebitoken, mis on standard, mis võimaldab kasutajatel autentida ja volitada end serverite ja rakenduste vahel ilma lisakinnitusteta [38].

JWT-i on lihtne kasutada, kuna need ei vaja serveripoolset salvestamist ega ole seotud konkreetse seansiga. See koosneb kolmest komponendist: päis, allkiri ning koormus, mis sisaldab informatsiooni kasutaja kohta [38].

Turvalisuse tagamiseks kasutatakse JWT-i genereerimiseks erinevaid krüptograafilisi algoritme ning seatakse aegumisaega. Antud diplomitöö raames kasutatakse HS256 algoritmi ja aegumisajaks seati 30 minutit.

Joonisel 14 on koodijupp, mis vastutab JWT-i genereerimise eest.

```
func GenerateToken(claims *Claims) (string, error) {
	expiresAt := time.Now().Add(time.Second * TokenTimeUntilExpiration)
		.Unix()
	claims.ExpiresAt = expiresAt
	claims.IssuedAt = time.Now().Unix()
	claims.Issuer = getIssuer()

	jwtToken := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)
	tokenString, err := jwtToken.SignedString([]byte(getJWTSecret()))
	if err != nil {
		return "", err
	}

	return tokenString, nil
}
```

Joonis 14 JWT-i genereerimine

4.2.5 Valideerimine

Domeeni objektide valideerimiseks on kasutusele võetud teek nimega „validator“, mis annab võimaluse lihtsasti defineerida valideerimisreegleid [39].

Antud projektis valideerimine toimub varevara etapil, mis valideerib kasutaja poolt saadetud andmeid.

Joonisel 15 on koodijupp struktuurist, mis hoiab kasutaja poolt saadetud registreerimis andmeid ning samuti ka valideerimisreeglite definitsiooni.

```
type UserCreation struct {
	Email    string `json:"email" validate:"required,email"`
	Username string `json:"username" validate:"required,min=3,max=16"`
	Password string `json:"password" validate:"required,min=6"`
}
```

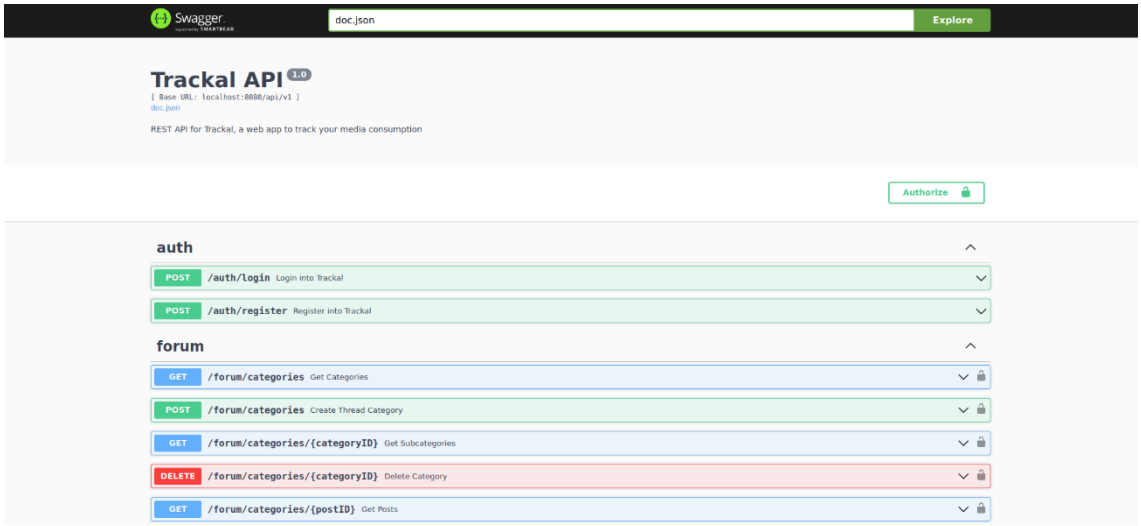
Joonis 15 Valideerimisreeglite definitsioon kasutaja loomise struktuuri näitel

4.2.6 Dokumentatsioon

Dokumentatsiooni loomiseks genereeris autor Swagger dokumentatsiooni kasutades teeke „gin-swagger“ ja „swag“.

Swagger dokumentatsiooni genereerimiseks defineeriti marsruutide konfiguratsioon iga rakenduseliidese marsruudi kommentaaridesse ning jooksutati vajalikud käsurea käsud.

Joonisel 16 on automaatselt genereeritud dokumentatsiooni ekraanitõmmis.



Joonis 16. Automaatselt genereeritud dokumentatsiooni ekraanitõmmis

5 Testimine

Antud diplomitöö raames valminud rakendusliidese testimiseks kasutati manuaalset testimist. Antud testimisviisi eesmärgiks on kinnitada, et iga päring tagastab nõuetekohast vastust. Manuaaltestimiseks oli kasutusele võetud Postman.

Postman on rakendus rakendusliideste loomise ja kasutamise jaoks, mida saab kasutada ka API testimiseks. Postmani abil on võimalik saata erinevaid päringuid, lisades päringule päringuparameetreid, päise ning andmeid [40].

Testimiseks oli loodud Postmani projekt ning defineeritud kõik rakendusliideses defineeritud marsruudid. Postman päringude defineerimist hõlbustas ka automaatselt genereeritud dokumentatsioon.

Antud testimisviisi eeliseks on see, et see testib rakenduse kõiki kihte korraga, kuid testimisprotsess on palju aeglasem kui automaattestide jooksutamine.

Testimise tulemusel leiti rakendusliideses olevad vead ning seejärel parandati need ära.

6 Rakenduse edasiarendused

Käesoleva töö käigus valminud rakendusliides täidab diplomitöö alguses püstitatud nõudeid, kuid ainult rakendusliidesest ei piisa. Selleks, et kasutajad saaksid antud rakendus kasutada on oluline arendada kliendirakendust, mis suhtleks antud rakendusliidesega.

Järgmiseks oluliseks arenduseks on tagada tarkvara kvaliteeti ja tõrgeteta toimimist automaatsete ühiktestide abil, mis katavad kogu koodibaasi. See võimaldab tuvastada tarkvara probleemid varakult ja tagab, et uute funktsioonide lisamine ei riku olemasolevat funktsionaalsust.

Samuti plaanib autor rakendusliidest ja tulevikus loodud kliendirakendust serveerida erinevatel platvormidel, nagu AWS, Azure või Google Cloud.

7 Kokkuvõte

Antud lõputöö eesmärgiks oli luua ühtse süsteemi, mis pakub kasutajatele mugavat ja intuitiivset meedia tarbimise jälgimist ning suurendada ühtlasi kasutajate kaasatust ja osalust. Püstitatud eesmärgid olid edukalt täidetud.

Eksisteerivate lahenduste analüüs andis hea ülevaate, milline loodud rakendusliides peaks olema ning pandi sellele varakult nõuded püsti. Analüüsis oli ka juttu tehnoloogia valikust, rakendusliideses arhitektuurist ning põhjendused tehtud valikutele.

Arenduse osas räägiti põhjalikumalt loodud rakenduse arhitektuurist ja üleüldiselt arenduseteemast.

Töö koostamisel rakendas autor ülikoolis õpitud teadmisi ning sai kogemust Go programmeerimiskeele tehnoloogiapinuga töötamises ning sellega rakendusliidese arendamises.

Kasutatud kirjandus

[1] „AniList,“ AniList, September 2013. [Võrgumaterjal]. Available: https://anilist.co/. [Kasutatud 5 märts 2022].

[2] „Wayback Machine,“ Internet Archive, 8 September 2013. [Võrgumaterjal]. Available: https://web.archive.org/web/*/anilist.co. [Kasutatud 5 märts 2022].

[3] „TV Time,“ TV Time, 2011. [Võrgumaterjal]. Available: https://www.tvtime.com/about. [Kasutatud 5 märts 2022].

[4] „Flox,“ Open Source Agenda, [Võrgumaterjal]. Available: https://www.opensourceagenda.com/projects/flox. [Kasutatud 5 märts 2022].

[5] „About Goodreads,“ Goodreads, [Võrgumaterjal]. Available: https://www.goodreads.com/about/us. [Kasutatud 5 märts 2022].

[6] „What is Python?,“ Python Software Foundation, [Võrgumaterjal]. Available: https://www.python.org/doc/essays/blurb/. [Kasutatud 13 mai 2023].

[7] „What is Java?,“ Oracle, [Võrgumaterjal]. Available: https://www.java.com/en/download/help/whatis_java.html. [Kasutatud 13 mai 2023].

[8] B. Lutkevich, „What is Kotlin?,“ [Võrgumaterjal]. Available: https://www.techtarget.com/whatis/definition/Kotlin. [Kasutatud 13 mai 2023].

[9] „A tour of C#,“ Microsoft, [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. [Kasutatud 13 mai 2023].

[10] C. Kolade, „What is Go?,“ freeCodeCamp, [Võrgumaterjal]. Available: https://www.freecodecamp.org/news/what-is-go-programming-language/. [Kasutatud 13 mai 2023].

[11] „What is JavaScript?,“ Mozilla Corporation, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. [Kasutatud 13 mai 2023].

[12] „TypeScript: Why does TypeScript exist?,“ Microsoft, [Võrgumaterjal]. Available: https://www.typescriptlang.org/why-create-typescript. [Kasutatud 13 mai 2023].

[13] „About Node.js,“ OpenJS Foundation, [Võrgumaterjal]. Available: https://nodejs.org/en/about. [Kasutatud 13 mai 2023].

[14] „Programming Languages From Easy To Hard To Learn,“ Code & Hack, 5 jaanuar 2023. [Võrgumaterjal]. Available: https://codeandhack.com/easy-to-hard-to-learn-programming-languages/. [Kasutatud 13 mai 2023].

[15] S. Veeraraghavan, „20 Best Programming Languages to Learn in 2023,“ Simplilearn, [Võrgumaterjal]. Available: https://www.simplilearn.com/best-programming-languages-start-learning-today-article. [Kasutatud 13 mai 2023].

[16] „Django overview,“ Django Software Foundation, [Võrgumaterjal]. Available: https://www.djangoproject.com/start/overview/. [Kasutatud 13 mai 2023].

[17] „Spring Framework,“ VMware Tanzu, [Võrgumaterjal]. Available: https://spring.io/projects/spring-framework/. [Kasutatud 13 mai 2023].

[18] „Language Support,“ VMware Tanzu, [Võrgumaterjal]. Available: https://docs.spring.io/spring-framework/docs/current/reference/html/languages.html. [Kasutatud 13 mai 2023].

[19] „Ktor,“ JetBrains, [Võrgumaterjal]. Available: https://ktor.io/docs/welcome.html. [Kasutatud 13 mai 2023].

[20] „Micronaut,“ Micronaut Foundation, [Võrgumaterjal]. Available: https://docs.micronaut.io/latest/guide/. [Kasutatud 13 mai 2023].

[21] „Overview of ASP.NET Core,“ Microsoft, [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0. [Kasutatud 13 mai 2023].

[22] „Gin Web Framework Documentation,“ Gin Team, [Võrgumaterjal]. Available: https://gin-gonic.com/docs/. [Kasutatud 13 mai 2023].

[23] „Next.js Documentation,“ Vercel, [Võrgumaterjal]. Available: https://nextjs.org/docs. [Kasutatud 13 mai 2023].

[24] „What is Oracle?,“ javatpoint, [Võrgumaterjal]. Available: https://www.javatpoint.com/what-is-oracle. [Kasutatud 13 mai 2023].

[25] „Oracle Technology Global Price List,“ Oracle Corporation, [Võrgumaterjal]. Available: https://www.oracle.com/assets/technology-price-list-070617.pdf. [Kasutatud 13 mai 2023].

[26] „About PostgreSQL,“ The PostgreSQL Global Development Group, [Võrgumaterjal]. Available: https://www.postgresql.org/about/. [Kasutatud 13 mai 2023].

[27] „GitHub Features,“ GitHub, Inc., [Võrgumaterjal]. Available: https://github.com/features. [Kasutatud 13 mai 2023].

[28] „GitLab, The DevSecOps Platform,“ GitLab B.V., [Võrgumaterjal]. Available: https://about.gitlab.com/. [Kasutatud 13 mai 2023].

[29] „BitBucket Overview,“ Atlassian, [Võrgumaterjal]. Available: https://bitbucket.org/product/guides/getting-started/overview#bitbucket-software-hosting-options. [Kasutatud 13 mai 2023].

[30] „Why Visual Studio Code?,“ Microsoft, [Võrgumaterjal]. Available: https://code.visualstudio.com/docs/editor/whyvscode. [Kasutatud 13 mai 2023].

[31] „IntelliJ IDEA Features,“ JetBrains s.r.o., [Võrgumaterjal]. Available: https://www.jetbrains.com/idea/features/. [Kasutatud 13 mai 2023].

[32] „Domain-Driven Design: Layers,“ HiBit, [Võrgumaterjal]. Available: https://www.hibit.dev/posts/15/domain-driven-design-layers. [Kasutatud 13 mai 2023].

[33] „Repository Pattern,“ DevIQ, [Võrgumaterjal]. Available: https://deviq.com/design-patterns/repository-pattern. [Kasutatud 13 mai 2023].

[34] „migrate package - Go packages,“ golang-migrate, [Võrgumaterjal]. Available: https://pkg.go.dev/github.com/golang-migrate/migrate/v4. [Kasutatud 13 mai 2023].

[35] „SQLBoiler: Generate a Go ORM tailored to your database schema,“ Volatile Technologies Inc., [Võrgumaterjal]. Available: https://github.com/volatiletech/sqlboiler. [Kasutatud 13 mai 2023].

[36] L. Richardson, M. Amundsen ja S. Ruby, RESTful Web APIs, Sebastopol: O'Reilly Media, Inc., 2013, p. 404.

[37] „What is Middleware,“ Microsoft, [Võrgumaterjal]. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-middleware/. [Kasutatud 13 mai 2023].

[38] „JSON Web Token Introduction,“ okta, [Võrgumaterjal]. Available: https://jwt.io/introduction. [Kasutatud 13 mai 2023].

[39] „validator package - Go packages,“ Go Playgound, [Võrgumaterjal]. Available: https://pkg.go.dev/github.com/go-playground/validator/v10. [Kasutatud 13 mai 2023].

[40] „API Tools | Postman API Platform,“ Postman, Inc., [Võrgumaterjal]. Available: https://www.postman.com/product/tools/. [Kasutatud 13 mai 2023].

[41] „DBML dokumentatsioon,“ Holistics, [Võrgumaterjal]. Available: https://github.com/holistics/dbml/tree/master/dbml-homepage/docs.

[42] „TV Time koduleht,“ [Võrgumaterjal]. Available: https://www.tvtime.com/. [Kasutatud 14 mai 2023].

[43] „Flox Live Demo,“ Flox, [Võrgumaterjal]. Available: https://flox-demo.pyxl.dev/. [Kasutatud 14 mai 2023].

[44] „Goodreads koduleht,“ [Võrgumaterjal]. Available: https://www.goodreads.com/. [Kasutatud 14 mai 2023].

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

Mina, Kristjan Mill

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Filmide ja teiste meediatüüpide jälgimisplatvormi rakendusliidese arendamine," mille juhendaja on Meelis Antoi.
   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.

13.05.2023

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.