TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Eric Rodionov 206833IAD

Mängusisese oksjoni jälgimisrakendus võrgumängus World of Warcraft

Bakalaureusetöö

Juhendaja: Einar Kivisalu
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: Eric Rodionov

15.05.2023

Annotatsioon

Mängusisese oksjoni jälgimisrakendus võrgumängus World of Warcraft

2018. aastal maailma populaarseima MMORPG-i World of Warcrafti arendajad. lõpetasid kaugoksjoni veebisaidi ja rakenduse. See otsus muutis mängijatel võimatuks lihtsalt ja kiiresti oksjonit jälgida ilma mängu sisse logimata. Õnneks arendajad on jätnud oksjoni API kasutajatele avatuks.

Selle töö eesmärk on luua avatud lähtekoodiga veebirakendus, mis skannib mängusiseseid oksjoneid World of Warcraft mängus ja kuvab ühe või mitme valitud serveri hinnadünaamikat valitud piirkonnas.

Lõputöö koosneb kahest osast: teoreetiline ja praktiline. Teoreetilise osa fookuses on API ja oksjoni struktuuri analüüs ning andmete kogumise, esitamise ja oksjoniandmete salvestamise tehnoloogiate valik. Lõputöö praktiline osa on veebirakenduse loomine ja selle GitHubis avatud lähtekoodiga väljalaskmine.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 29 leheküljel, 5 peatükki, 12 joonist, 3 tabelit.

Abstract

In-Game Auction Monitoring Application for the World of Warcraft Online Game

In 2018, developers of World of Warcraft, the most popular MMORPG in the world, discontinued the remote auction website and app. This decision made it impossible for players to easily and quickly track the auction without having to log into the game. Fortunately, the developers have left the auction API open for users.

The aim of this work is creation of a web application with open source code that scans in-game auction in World of Warcraft game and displays the price dynamics for one or more selected realms in the selected region.

The thesis consists of two parts: theoretical and practical. Analysis of the structure of API and auction, and the choice of technologies for data collection, presentation and storage of the auction data are the focus of the theoretical part. The practical part of the thesis is creating a web application and releasing it as open source on GitHub.

The thesis is in Estonian and contains 29 pages of text, 5 chapters, 12 figures, 3 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendustarkvara liides
Commodity	Kaubad, millega saab kaubelda piirkonna erinevate *Realm'i* vahel
Connected realm	*ConnectedRealms* on kahe või enama realm'i komplekt, mis on püsivalt ja sujuvalt "lingitud".
Docker image	*Image* on fail, mida kasutatakse Dockeri konteineris koodi käivitamiseks
Docker-compose	*Compose* on vahend mitme konteineriga Dockeri rakenduste käitamiseks
HTTP	*HyperText Transfer Protocol*, andmeedastusprotokoll
JSON	*JavaScript Object Notation*, tekstipõhine andmevahetusvorming, mis põhineb JavaScriptil.
MMORPG	*Massively Multiplayer Online Role-Playing Game, Interneti-rollimäng suure numbrite mängijatega*
MVP	*Minimal Viable Product, toote versioon minimaalse funktsioonidega, mis vastab kõigile kriteeriumidele.*
Non Commodity	Kaubad, millega saab kaubelda ainult ühes *Realm'is*
Permission-free	*Permission free* on litsentsitüüp, mis võimaldab teistel soovi korral teise looja teoseid uuesti kasutada.
Realm	*Realm* on näide World of Warcrafti (WoW) mängumaailmast. Mõisteid "valdkond" ja "server" kasutatakse sageli vaheldumisi.
Title Case	*Title Case* on suurtähtede kirjutamise stiil, kui kõik sõnad on suurtähtedega, välja arvatud väikesed sõnad (tavaliselt artiklid, lühikesed eessõnad ja mõned sidesõnad).
WoW	World of Warcraft, arvutimäng

Sisukord

1 Sissejuhatus

World of Warcraft on MMORPG, mille 2004. aastal andis välja Blizzard Entertainment, praegu üks populaarsemaid MMORPG-sid maailmas. Mängus täidavad mängijad ülesandeid, vallutavad koopasid, võitlevad üksteisega ja koguvad ressursse, et hankida parim varustus ja täita väljakutseid pakkuvaid saavutusi.

Mängul on tasuline finantsmudel, mänguaega saab žetoonidega pikendada. Üks žetoon maksis 20 eurot ja on võrdne ühe kuu mänguaega. Veel on võimalus osta žetoone suure hulga mänguvaluuta - kulla eest. Mängijad püüavad sageli teenida piisavalt kulda, et osta žetoone ja mitte kulutada päris raha. Ja kõige olulisem vahend kulla hankimisel on oksjon. Oksjonimaja on mängu üks peamisi aspekte, kus mängijad saavad ressursse, varustust ja tarvikuid kulla vastu vahetada. Kuna aeg on piiratud, on mängijatel ülioluline teada, milliseid ressursse maksimaalse kasumi saamiseks kaevandada. Seetõttu on praegused oksjonihinnad muutunud oluliseks teadmiseks. Kuna hinnad oksjonimajas ei ole stabiilsed, vaid sõltuvad pakkumisest ja nõudlusest, hakkasid mängijad sageli hindu jälgima ka mängu väljaspool.

2018. aastal Blizzard lõpetas kaugoksjoni veebisaidi ja rakenduse. See otsus muutis selle võimatuks mängijatel hõlpsalt ja kiiresti oksjonit jälgida, ilma et oleks vaja mängu sisse logida. Õnneks arendajad on jätnud oksjoni API kasutajatele avatuks ja selle tulemusena hakkasid entusiastid oksjonimaja jaoks oma veebisaite tegema.

2022. aastal, 18 aastat pärast mängu väljaandmist arendajad otsustasid ümber kujundada oksjoni struktuuri. See tõi kaasa olukorra, kus peaaegu kõik mängu fännide tehtud oksjonisaidid lakkasid töötamast. Ainus oksjonisait, mis praegu korralikult töötab, on suletud lähtekoodiga, nii et kui see sait mingil põhjusel lakkab töötamast, ei saa mängijad hindu võrguühenduseta jälgida [1]. Seetõttu otsustati lõputööna teha veebirakendus, mis skannib Blizzard API kaudu kaupade hindu ja nende dünaamikat, seejärel postitada rakenduse avatud lähtekoodiga, et kasutajad saaksid ise oksjonimonitori hostida või kasutada rakenduse koodi oma loomiseks oma modifikatsioonid ja analoogid.

2 Analüüs

Enne praktikaosa alustamist on vaja teha analüüs. Järgmine osa sisaldab visiooni, projekti ulatust, võtmeprobleemide analüüsi, võimalikke lahendusi ning funktsionaalsete ja mittefunktsionaalsete eesmärkide seadmist.

2.1 Visioon ja ulatus

Toote visioon on World of Warcrafti oksjoni skanner, mis võimaldab mängijatel kiiresti saada infot hinnadünaamika kohta ilma mängu sisenemata.

Esimene eesmärk on aidata mängijatel säästa aega ja raha, lihtsustades turuhindade analüüsi protsessi, et otsida tulusaid ressursse müügiks. Et mõista, kui stabiilne on toorme hind, peaksime esitama ka hinnadünaamika.

Rakendus peab Blizzardi API abil perioodiliselt taotlema oksjonimaja andmeid. Seejärel tuleks oksjonimaja andmed töödelda ja salvestada kohalikku andmebaasi, et koguda hinnaajalugu. Kõik nimetatud protsessid, välja arvatud konfiguratsioon ja rakenduse kaivitamine, peaksid töötama automaatselt, ilma kasutaja sekkumiseta. Tuleks teha kasutajaliides, et kasutaja saaks oksjonimaja andmeid mugavalt näha ilma käsitsi andmebaasipäringuteta. Täiendava mugavuse huvides peaks kasutajal olema võimalus otsida üksusi nime järgi.

Teine eesmärk on pakkuda toodet avatud lähtekoodiga, et kasutajad saaksid ise veebirakendust hostida. See vähendab sõltuvust vähestest praegu toimivatest veebioksjonitest. Sel juhul, kui need suletud lähtekoodiga projektid mingil põhjusel suletakse, on mängijatel endiselt juurdepääs oksjonimajale väljaspool mängu. See probleem muutub veelgi käegakatsutavamaks, kuna üks suurimaid praegu töötavaid veebioksjonite skannereid *The undermine journal* lõpetab töötamise 1. mail 2023 [2]. Fännide API-de katkestamise vältimiseks peavad andmed, mille andmebaasi salvestame, pärinema algallikast – Blizzard API-st.

Avatud lähtekoodiga rakenduse avaldamine tähendab ka seda, et me ei tohi rikkuda arenduse käigus kasutatud tehnoloogiate autoriõiguse litsentse. Seetõttu tuleb projekti pidevalt kontrollida, kas seda on võimalik loavaba litsentsi alusel välja anda.

Kuna projekt on kavandatud nii, et kasutajad saaksid rakendust ise hostida, peaks see olema säästlik mälu ja kettaruumi osas. Võrdluseks, *Undermine journal* käivitati eraldi hostmasinas, mis nõudis 16 GB muutmälu ja 256 GB kettaruumi [3]. Erinevalt Undermine Journalist oleks tore, kui meie rakendus töötaks koos World of Warcrafti mänguga. Seetõttu tasub otsida meetodeid, kuidas andmeid tõhusamalt salvestada, võimalusel välja filtreerida pakutavad "lisaandmed". Näiteks kaupade hinnadünaamika jälgimiseks piisab oksjonipartiide salvestamisest minimaalse hinnaga. Nii ei ole kettaruumi tohutu kokkuhoiu huvides vaja ülejäänud oksjonipartiisid salvestada. See on hea otsus, mis annab kasutajale võimaluse valida, milliseid valdkondi jälgida, nii et teavet teiste valdkondade kohta ei salvestata. Üks tähtsamaid omadusi on, et rakendus peab olema piisavalt lihtne, et seda saaks seadistada ja töös hoida tekstipõhise juhendi järgi, ilma et oleks vaja arendajpoolset abi.

2.2 Metoodika

Visioon ja töö ulatus on määratletud, nüüd peame selle visiooni saavutamiseks tegema plaani, mida tuleb järgida kogu rakenduse arendamise ajal.

Alustuseks peaks arendaja kaaluma põhikriteeriume, mille järgi saab öelda, et töö on minimaalselt elujõuline toode. Funktsionaalsed ja mittefunktsionaalsed nõuded tuleks kirjeldada, et mõista, millistele rakenduse omadustele ja parameetritele tuleks rohkem ressursse ja tähelepanu pöörata.

Järgmisena peaksime analüüsima, kuidas API töötab. Näiteks kui keeruline on WoW API kaudu kõiki vajalikke andmeid kätte saada ning kas päringutele ja nende arvule on mingeid piiranguid.

Kui rakenduse kriteeriumid ja disainifunktsioonid on teada, tuleks teha instrumentide valik. Selles etapis ei ole vaja kirjeldada väiksemaid asju, nagu raamatukogud, kuna need võivad arenduse käigus drastiliselt muutuda, kuid selles etapis tuleks teha valik põhitehnoloogiate kohta ja selgitada iga instrument valimise põhjust.

Pärast instrumente valikut alustatakse rakenduste arendamist. Arendus peaks algama andmebaasi installimisest ja selle ühendamisest rakendusega. Praeguses etapis me ei tea, milline API vastuse objekt välja näeb, seega tasub teha paar testipäringut, et täpsemalt teada saada, milliseid andmeid pakutakse ja mida veel Blizzard API-ga töötamiseks vaja

on. Kui oleme teadlikud vastuseobjektide struktuurist ja nende pakutavatest andmetest, saame alustada andmebaasimudelite ja päringute loomist. Määratakse andmebaasi mudelid ja API objektid, koodi loetavuse ja paindlikkuse parandamiseks teeme API ja andmebaasiobjektide vahele kaardistajaid.Pärast kaardistajaid saame hakata koostama HTTP päringute eest vastutavat koodi. Peame looma klassi, mis küsib oksjonimaja andmeid ja tagastab objektid, mis on valmis vastendamiseks andmebaasiobjektidega. Kui API nõuab loa autoriseerimist, tuleks teha ka autoriseerimisloa küsimise meetod. Nüüd, kui andmebaas, API päringud ja kaardistaja töötavad, saab alustada oksjonimaja skanneri arendamist. See peaks olema taustrakendus, mis saab api-lt andmeid, teisendab need vajadusel kaardistaja abil ja salvestab andmebaasi. Viimase sammuna tuleb luua veebirakendus, mis saab andmebaasist andmed ja esitab need kasutajale sobivas vormingus.

Rakenduste arendamise viimane osa on konteinerisse paigutamine. Rakendus konteineritakse dockeris sellega töötamise hõlbustamiseks.

Pärast rakenduse arendamise lõpetamist tasub kontrollida, kui palju mälu ja operatiivmälu rakendus võtab. Kas tavakasutaja suudaks sellist rakendust käivitada ja seda tehniliste nõuete osas kasutada?

Edasi arendaja peab analüüsima kasutatavaid teeke ja kontrollima, et rakendust saaks ikka loavaba litsentsi alusel välja anda, kui leidub vahend, mis sellele kriteeriumile ei vasta, tasub leida analoog.

Kui instrumentidega probleeme pole ja toode vastab miinimumnõuetele, kirjutatakse juhend, kuidas rakendust käivitada ja konfigureerida. Lõpuks avaldatakse rakendus GitHubis avatud lähtekoodiga loavaba litsentsiga.

2.3 Süsteemi nõuded

Rakendusnõuded jagunevad vastavalt nende omadustele kahte tüüpi – funktsionaalsed ja mittefunktsionaalsed. Funktsionaalsed nõuded kirjeldavad, mida süsteem peaks tegema, mittefunktsionaalsed nõuded kirjeldavad, kuidas süsteem käitub.

2.3.1 Funktsionaalsed rakenduse nõuded:

- Toode peab küsima hinna ja kauba kirjeldusi Blizzard API-st..

- Nõutud andmeid töödeldakse ja salvestatakse kohalikku andmebaasi.
- Rakendus peab sisaldama mitte ainult jooksvad hinnad, vaid kõik varasemad skannitud hinnad.
- Oksjonimaja andmete skaneerimine ja salvestamine toimub automaatselt ja perioodiliselt.
- Rakendus peab suutma andmebaasi salvestatud üksusi kasutajale esitada.
- Kasutajal peaks olema võimalik nõuda andmeid üksuse kohta nime järgi.
- Rakenduse saab teha avalikkusele kättesaadavaks loavaba litsentsi alusel.

2.3.2 mittefunktsionaalsed rakenduse nõuded:

- Rakendus peab võtma võimalikult vähe kettaruumi ja protsessori kasutust, et mängija saaks seda oma arvutis käivitada ega tekitaks teiste rakendustega töötamisel ebamugavusi.
- Rakendus peab andma kasutajale andmed kiiresti.
- Rakendust peaks olema piisavalt lihtne installida ja konfigureerida, et kasutaja saaks seda tekstijuhise abil käivitada.

2.4 Blizzard API analüüs

Blizzard API on rakenduste programmeerimisliides, mis pakub juurdepääsu andmetele ja funktsioonidele, mis on seotud Blizzard Entertainmenti arendatud mängudega, nagu World of Warcraft, Diablo III, Hearthstone jne. Blizzardi API võimaldab arendajatel luua rakendusi, mis võivad omavahel suhelda erinevate andmetega, nagu statistika, saavutused, üksused, tegelased jne lihtsate HTTP REST päringute kaudu. Blizzardi API kasutamine on tasuta, kuid sellega kasutamiseks peate registreerima Battle.net konto (või sisse logima, kui konto on juba olemas) ja looma oma API juurdepääsu. Selle API-juurdepääsuga saab kasutaja *Client Id* ja *Client Secret*, mida on vaja API-le päringuid võimaldava loa loomiseks. Tokeni loomiseks Blizzard kasutab OAuth 2.0 autoriseerimisprotokolli. Juurdepääsumärgid kehtivad 24 tundi [4]. Blizzard API juurdepääsul on päringumäära limiit 36 000 päringut tunnis ja 100 taotlust sekundis.

Enne oksjonimaja *commodity* ja *non-commodity* selgitamist on oluline selgitada, et WoW-mängijad loovad ühenduse serveritega, nn valdkonnad ja valdkonnad on ühendatud piirkondadeks. Erinevatest piirkondadest pärit mängijad ei saa mängus üksteisega suhelda [5]

Alates 2022. aastast WoW oksjonimaja kaubad enam ei ole seotud ainult serveritega. Nüüd on kaubad jagatud kahte tüüpi: *commodities* ja *non-commodities*.

Commodity on kaubad, mida müüakse serverite vahel. See tähendab, et kõik mängijad saavad osta või müüa esemeid teistele samas piirkonnas asuvatele serveritele, välja arvatud juhul, kui tegemist on Commodity'ga. Teist tüüpi kaubad on non-commodity. *Non-commodity* kaubad on nähtavad ainult mängijatele samas valdkonnas, kus toodet sisaldav partii oli loetletud [6]. Pärast oksjoni struktuuri muutmist kujundati ümber ka API. Nüüd on võimatu ühe päringuga serverisse saada nii commodity kui ka non-commodity. Need on kaks erinevat taotlust, millel on erinevad API päringutega [7]. API päringus saame palju teavet oksjonipartii kohta, nagu hind, kogus jne. Kuid meil pole üksuse enda kohta teavet, välja arvatud selle ID. See tähendab, et kui soovime kauba kohta mingit infot kuvada, tuleb teha eraldi infopäring. Probleem on selles, et Blizzardi API ei paku võimalust hankida teavet partiide kaupa. See tähendab, et kui meil on vaja saada teavet 10 000 üksuse kohta, peame tegema eraldi 10 000 päringut. Arvestades Blizzard API päringute sageduspiiranguid, tähendab see ka seda, et ainult hinnaandmeid pole võimalik salvestada, kuna kaupade kohta päringute kaudu teabe hankimine võtab palju aega. Lihtsaim lahendus probleemile on luua lisateenus kaubakirjelduste skannimiseks ja hoida kauba kohta infot andmebaasis. See lahendus muudab andmete hankimise palju kiiremaks, sest meil pole vaja API-le päringuid teha ning piisab andmete perioodilisest uuendamisest. Võetud kettaruumi hulk üksuste teabe kaupa jääb peaaegu muutumatuks, sest see maht kasvab vaid veidi ja ainult suuremate uuendustega Ka ei pea salvestama kõiki api-st saadavaid andmeid, kuna kasutajad peavad tavaliselt kuvama ainult nime, üksuse kategooria ja alamkategooria, nii et see veelgi vähendab kettatarbimist.

Teine oluline osa on asjaolu, et oksjonid ei ole seotud mitte *realm*'iga, vaid *connected-realms*'iga [8]. *Connected realms* on mitme väiksema valdkonna koondumine, mille eesmärk on parandada mängijate kogemust oksjonimajas ja muus tugevalt elanikkonnaga

seotud mehaanikas [9]. Arengu seisukohalt tähendab see ka seda, et mittetarbekaubapartiide arv kõigis ühendatud valdkondades jaotatakse ühtlasemalt.

2.5 Tehnoloogia Pinu

Vajaliku veebirakenduse loomiseks tuleb teha **esirakendus**, mis näitab kasutajale andmeid, **tagarakendus**, mis võtab andmeid vastu, töötleb ja salvestab, ning **andmebaasi**, mis salvestab ja edastab andmeid kasutajaliidesele ja taustaprogrammile. Selles osas kirjeldatakse ja selgitatakse valitud instrumente. Samuti on meie jaoks olulised kasutatavate tehnoloogiate litsentsid, mistõttu on need mainitud iga instrumentide kirjelduse lõpus.

2.5.1 .Net Core

.Net Core on täispinu raamistik hästi optimeeritud ja mugava instumentikomplektiga triviaalsete ülesannete lahendamiseks nagu andmebaasiühendus, objektide kaardistamine, sõltuvuse süstimine jne. Mugav moodulsüsteem võimaldab laadida ainult seda, mida arendaja vajab, muutes rakenduse kiiremaks, paindlikum ja kergem. Näiteks võimaldab Razor kasutada dünaamiliste lehtede loomiseks HTML-mallides C# koodi mis teeb veebirakenduste arendamise veelgi kiiremaks.

Viimaseks, .NET Core'i levitatakse MIT Litsentsi all, mis on avatud ja tasuta litsents, mis võimaldab lähtekoodi ja kompileeritud faile kasutada, muuta ja edasi levitada ilma kasutajate või seadmete arvu piiramata.

2.5.2 TimeScaleDB

Kuna rakendus salvestab aegridu, otsustasin kasutada aegridade andmebaasi. Seda tüüpi andmebaas on optimeeritud aegridade salvestamiseks ja teenindamiseks ning kasutab ka maksimaalselt ära ajaga seotud andmefunktsioonid. Näiteks aegridade andmekogumitel on tavaliselt vähem seoseid eri tabelite andmekirjete vahel, mis võimaldab päringute kiirendamiseks kasutada spetsiaalseid indekseid [10]. Paljudest erinevatest aegridade andmebaasidest kuulsin kõige sagedamini kolmest võimalusest - TimescaleDB, InfluxDB ja eXtremeDB. eXtremeDB ei ole litsentsi tõttu abikõlblik, nad kasutavad kommertslitsentsi, seega ei saa avatud lähtekoodiga projekti teha. See ei vasta taotlemise kriteeriumidele [11]. Valides influxdb ja timescaledb vahel, suurt rolli mängib asjaolu, et esimene kasutab oma päringukeelt, mis vähendab oluliselt koodi loetavust ja

arendusmugavust. TimeScaleDb on PostgreSQL-i laiendus, mis on loodud töötama ajaga seotud andmetega. See kasutab PostgreSQL-põhist arhitektuuri, millega mul on kogemusi ja mida toetatakse Entity Framework Core'i (lühendatud nimega EFCore), objekti-andmebaasi kaardistaja .NET-i tuuma jaoks. See muudab koodi kogukonna jaoks loetavamaks ja arusaadavamaks, kuna seal ei kasutata kohandatud päringuid, vaid .Net core'i pakutavaid instrumentad.

Kuna TimescaleDB on Postgresi laiendus, tasub kaaluda ka Postgresi litsentsi, kuna tehniliselt kasutame seda endiselt.

PostgreSQL on välja antud PostgreSQL-i litsentsi all mis on sisuliselt sarnane MIT litsentsiga ja võimaldab teha *permission-free* avatud lähtekoodiga projekti [12].

TimescaleDB on avatud lähtekoodiga ja tasuta tarkvara, mis on saadaval Apache 2.0 litsentsi alusel, võimaldades seda piiranguteta kasutada mis tahes projekti jaoks [13].

2.5.3 Docker

Docker on rakenduste konteineriseerimise tehnoloogia, mis võimaldab rakendusi konteinerites käitada, muutes need kergeks, kaasaskantavaks ja turvaliseks. See muudab rakenduste ettevalmistamise palju lihtsamaks, näiteks kaob vajadus andmebaasi käsitsi installida ja selle ühendust rakendusega konfigureerida.

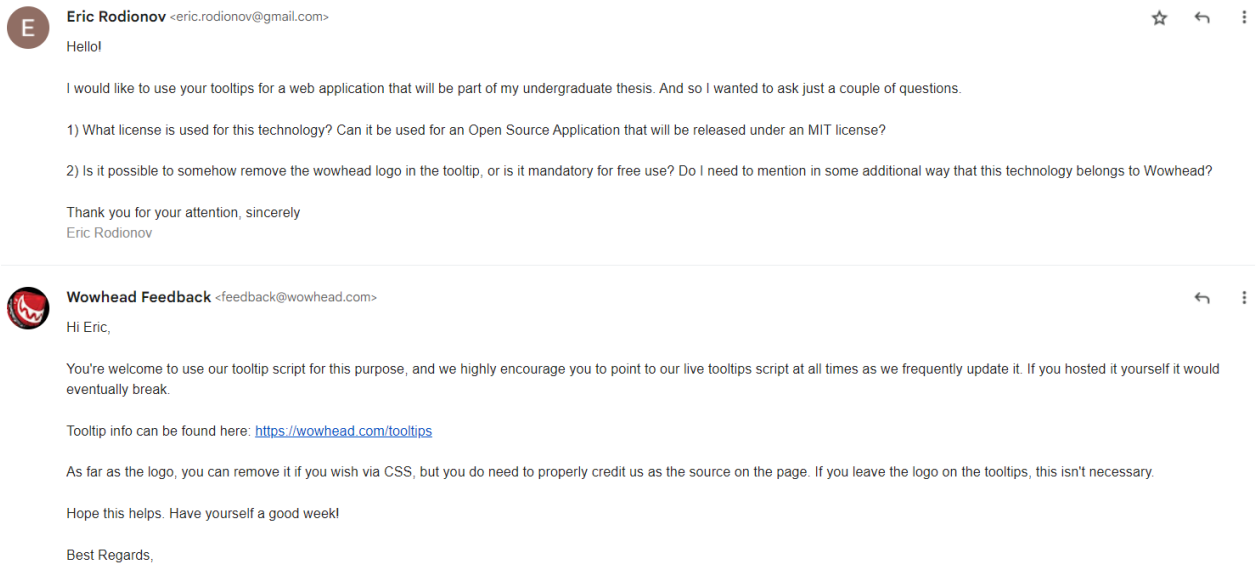
Docker on isiklikuks kasutamiseks ja avatud lähtekoodiga projektide jaoks tasuta [14]. See mõistab et luua avatud lähtekoodiga rakenduse, kasutades dockeri konteinereid on võimalik. Kui inimene otsustab rakendust ärilistel eesmärkidel kasutada, ei pea ta tegema muud, kui muutma failis appsettings.json ühendusstringi ja käivitama rakenduse otse, mitte dockeris loodud pildi kaudu.

2.5.4 Andmete visualiseerimise vahendit

Wowhead Tooltips on JavaScripti kirjetatud skript, mida kasutatakse veebilehtedel WoW-stiilis üksuste vihjete kuvamiseks. Tehnoloogiat kasutatakse, kuna see muudab kasutaja suhtlemise andmetega palju lihtsamaks, kuna inimene saab eseme kirjelduse ja selle välimuse täpselt samal kujul kui mängus

Litsentsi kohta teavet toote veebisaidilt ei leitud, mistõttu otsustati kirjutada otse arendajatele ja küsida, kas nende toodet saab kasutada loavaba litsentsi alusel. Joonisel 1

näete kirja, milles arendajad on lubanud oma toodet loavaba litsentsi alusel kasutada kahel juhul: kasutaja kasutab tehnoloogiat sellisel kujul või kui kasutaja otsustas logo eemaldada, lehe allikas tuleks siiski märkida tehnoloogia autorsus.



Eric Rodionov <eric.rodionov@gmail.com>

Hello!

I would like to use your tooltips for a web application that will be part of my undergraduate thesis. And so I wanted to ask just a couple of questions.

1) What license is used for this technology? Can it be used for an Open Source Application that will be released under an MIT license?

2) Is it possible to somehow remove the wowhead logo in the tooltip, or is it mandatory for free use? Do I need to mention in some additional way that this technology belongs to Wowhead?

Thank you for your attention, sincerely
Eric Rodionov

Wowhead Feedback <feedback@wowhead.com>

Hi Eric,

You're welcome to use our tooltip script for this purpose, and we highly encourage you to point to our live tooltips script at all times as we frequently update it. If you hosted it yourself it would eventually break.

Tooltip info can be found here: https://wowhead.com/tooltips

As far as the logo, you can remove it if you wish via CSS, but you do need to properly credit us as the source on the page. If you leave the logo on the tooltips, this isn't necessary.

Hope this helps. Have yourself a good week!

Best Regards,

Joonis 1. Tingimused, mille alusel arendajad lubavad oma tehnoloogiat kasutada.

Google Charts on Google'i pakutav tasuta veebilehtede diagrammi- ja graafikutööriist. See on väga mugav ja optimeeritud tehnoloogia, seda kasutatakse hinnadünaamika graafiliseks kuvamiseks kasutaja mugavuse kasuks.

Projekti levitatakse Apache License 2.0 all [15].

3 Programmi arendamine

See peatükk annab ülevaate rakenduse disainist ja arendamisest, võimalikest ja/või ilmnenud probleemidest arenduse käigus.

Kasutaja mugavuse huvides on kõik vajalikud andmed ja seaded, mida kasutaja käsitsi täidaks, ühes failis - *appsettings.json*. Seetõttu saadakse sellest failist kõik kasutajalt rakenduse toimimiseks vajalikud andmed..

3.1 Andmebaasi seadistamine

Andmebaas on üks olulisemaid tarkvarakomponente. See on tööriist andmete salvestamiseks, korraldamiseks, haldamiseks ja töötlemiseks.

Kuna andmebaas asub dockeris, peame looma dockeri koostamise faili. See fail konfigureerib rakenduse teenused. Seejärel on ühe käsuga võimalik luua ja käivitada kõik teenused antud konfiguratsioonist. Kuigi *TimescaleDB* on vaid postgresi laiendus, installijuhendist pärit dockeri *image* on juba sisaldab postgres [16]. *Docker-compose*'is määrame pildi aadressi ja kohalikud muutujad, nagu andmebaas, nimi, parool ja kasutajanimi. Kasutame andmebaasiga suhtlemiseks Entity Framework Core'i. Entity Framework Core'iga saate luua ja töötada objekte, mis esindavad andmebaasis olevaid andmeid, samas kui Entity Framework Core hoolitseb selle eest, kuidas neid andmeid andmebaasist salvestatakse ja sealt välja tuuakse [17]. Koodi lõplik versioon, kus on valmis andmebaasi konteineristamise näidiskood, on näha joonisel 6 punktis 3.6.2.

Nüüd peame looma klassi, mis pärib *DbContext*. See klass on meie andmebaasi kontekst. Kontekst toimib nii hoidla, st andmebaasile juurdepääsu fassaadina, kui ka *Unit of Work*'ina, mis tähendab, et see koordineerib mitme repositooriumi tööd, luues ühe andmebaasi kontekstiklassi, mida nad kõik jagavad [18]. Lisaks andmete esitamisele kirjeldati kontekstiklassis ka mudelite loomist ja seadistamist. Mudel on andmebaasi olemi klassi esitus.

Kui kontekst on loodud, on hea tava lisada see .Net Core Dependency Injection'is. Nüüd saab andmebaasiga suhtlemise konteksti hankida .Net Core Dependency Injection abil. Praegu ei saa me andmebaasi jaoks mudeleid teha, kuna me ei tea, millist teavet Blizzard

Web API pakkuda suudab. Seetõttu tuleb enne relatsiooniskeemi loomist täpsustada, milliseid andmeid me API-lt saame.

3.2 Blizzard API andmepäring

Järgmine samm on luua klass HTTP-päringute jaoks Blizzard API-le. Taotluste jaoks kasutan *RestSharp*i teeki. RestSharp on avatud lähtekoodiga [19] teek, mis toimib ümbrisena natiivse .Net Core HttpClient klassi jaoks ja pakub mõningaid lisafunktsioone. Valisin selle, kuna see lihtsustab vastuste serialiseerimise ja deserialiseerimise protsessi ning muudab koodi loetavamaks.

Esimene samm on teha juurdepääsuluba saamiseks autoriseerimistaotlused, vastasel juhul pole ülejäänud API-ga võimalik töötada. Järgmine meetod on *connected realms*i teabe skannimine. Kasutaja mugavuse huvides otsustati teha nii, et kui *appsettings.jsoni* väli "RealmId" on tühi, siis sait kuvab kõigi *connected realm*i nimede ja ID-de loendi.

Järgmisena käsitletakse commodity ja non-commodity hindade saamise meetodeid. Commodity ja non-commodity omadused on peaaegu sarnased. Ainus erinevus seisneb selles, et commoditiy'il on atribuutide lisaomadused "unit_price" ja non-commodity kaks lisaomadust: "buyout" ja "bid". Kauba väärtus on märgitud vaskmüntides. Mängu teisendus on 1 kuld = 100 hõbedat = 10 000 vaskmünti. Võib oletada, et arendajad otsustasid API-s määrata hinnad kulla asemel vasele, et vältida ujukoma vigade leevendamise probleemi.

Tabel 1, commodity ja non-commodity võrdlus

CommodityAuction Fields	NonCommodityAuction Fields
ItemId, long	ItemId, long
Quantity, int	Quantity, int
TimeLeft, string	TimeLeft, string
UnitPrice, long	Buyout, long
	Bid, long

Viimane API-päringuid vajav meetod on üksuste kohta teabe hankimise meetod. On atribuute, mida kasutab mänguloogika või mis pole oksjonihinna skanneri jaoks vajalikud.

Kõigi üksuste skannimiseks peate hankima kõigi unikaalsete üksuste ID-de loendi. ID-de nimekirja saame odavaima kauba- ja mittekaubapartiide loendist. Oksjonil on esemeid, mille kohta teabe hankimisel ilmneb tõrge. Seda tasub rakenduse arendamisel arvestada, kuna tooted võivad salvestada valeandmeid või takistada meetodi töötamist. Otsustati lisada veel kord teema kohta info täiendav kontroll, juhuks kui tegemist on ühekordse veaga või kui päringud olid liiga kiired, kuna Blizzard hoiatas, et tunnikvoodi ületamisel võib väljastada vea [20]. Kui teise päringu ajal viga ei parandatud, jäetakse toode vahele. Samuti Blizzard ei anna dokumentatsioonis mingit vastuse andmeväljade teavet, seega on mõne välja eesmärk teadmata. Seetõttu võtame optimeerimise eesmärgil ainult seda teavet, mis on kasulik andmete mugavaks kuvamiseks või võib olla tulevikus kasulik andmete sortimiseks. Kauba teabe kirje sisaldab järgmisi andmeid:

Tabel 2, itemInfo omadused

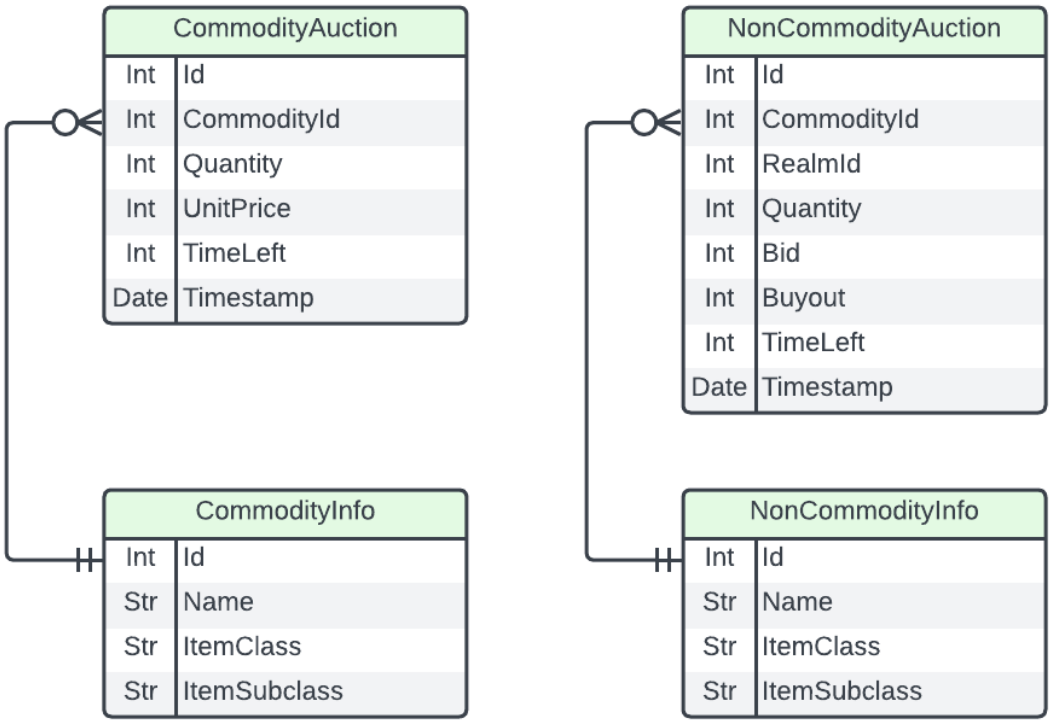
Item info Fields
Item id, long
Item name, string
Item class (mängusiseselt tuntud ka kui item type või item category), string
Item subclass (mängusiseselt tuntud ka kui item subtype või item subcategory), string

3.3 Mudelite ja Recordite loomine, mapper

3.3.1 Mudelid ja andmebaas

Nüüd, kui teame, milliseid andmeid API pakkuda suudab, on aeg teha andmebaasi jaoks mudeleid. Andmebaasi rakendamise mugavuse huvides tuleks koostada Entity Relationship Diagram, või Olemi suhete diagramm. Koodiosa ei pea täpselt kordama

diagrammil näidatud interaktsioone, vaid peab järgima vähemalt loogikat, mille me sellesse paneme.



Joonis 2, andmebaasi olemite relatsiooniskeem.

Esmapilgul võib tunduda, et tegime põhimõttelise vea, kuna eirame normaliseerimisreegleid, tehes kaks eraldi tabelit, millel on samad omadused. Seda tehakse selleks, et saaksime commodity ja non-commodity atribuute iseseisvalt muuta. Tulevikus soovime võib-olla kasutada lisavälju non-commodity jaoks, nagu nõutav tase, varustuse pesa, kauba tase jne. Sel juhul võimaldab tabeli kaheks osaks jagamine koodil olla paindlikum. Lõpuks, kuna andmebaasis on nii vähe tabeleid, ei kahjusta see jõudlust palju.

Mudelite andmebaasi pääsemiseks peab looma migratsiooni. Migratsiooni loomiseks peame määratlema mudelid varem loodud kontekstis. Kontekstis peate looma meetodi OnModelCreating() ja kirjeldama mudelite peamisi võtmeid. Näidiskoodist joonisel 3 näete, et CommodityAuction ja NonCommodityAuction kasutavad liitvõtmeid. See on vajalik, kuna ühe ID-ga üksuse puhul tehakse palju koopiaid erinevatel aegadel ja NonCommodity puhul kasutatakse ühendatud domeeni ID-d, kuna on võimalus, et salvestame ID mitme realmi jaoks.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);

    modelBuilder.Entity<CommodityAuction>()
        .HasKey(x => new{ CommodityId = x.ItemId,x.TimeStamp});

    modelBuilder.Entity<CommodityInfo>()
        .HasKey(x => x.Id);

    modelBuilder.Entity<NonCommodityAuction>()
        .HasKey(x => new{ NonCommodityId = x.ItemId,x.TimeStamp,x.RealmId});

    modelBuilder.Entity<NonCommodityInfo>()
        .HasKey(x => x.Id);
}
```

Joonis 3, mudelite määratlemiseks näidiskood

Lisaks API-lt saadud andmetele lisasime ka ajatemplid ja täpsustasime väljad NonCommodity. Täiendavad andmed luuakse enne andmebaasi salvestamist. Samuti salvestame ajatemplisse ainult kuupäeva ja tunni, jättes minutid, sekundid ja millisekundid nulliks. See on vajalik selleks, et edaspidi oleks viimasel tunnil skaneeringute vastuvõtt mugavam.

3.3.2 Hypertables

Nüüd, kui andmebaas on moodustatud, peame tegema *hypertable***.** Kuna EFcore toetab Postgresi osa, kuid TimeScaleDB-d ei toetata, toimub nende tabelite loomine käsitsi. Praegu versioonis migreerimisele on lisatud koodilõigu, mis loob kaupade ja mittekaupade jaoks hüpertable'id. Muudatused tehakse andmebaasi värskendamise ajal migreerimise kaudu, nii et seda osa saab kasutajale hõlpsasti ette valmistada ja ta peab enne rakenduse kasutamist migreerimisi värskendama ainult ühe käsu kirjutamisega.

```
migrationBuilder.Sql(
"SELECT create_hypertable( '\"CommodityAuctions\"', 'TimeStamp');\n" +
"CREATE INDEX ix_commodity_timestamp ON \"CommodityAuctions\" (\"ItemId\",
\"TimeStamp\" DESC)"
);

migrationBuilder.Sql(
"SELECT create_hypertable( '\"NonCommodityAuctions\"', 'TimeStamp');\n" +
"CREATE INDEX ix_noncommodity_timestamp ON \"NonCommodityAuctions\"
(\"ItemId\", \"TimeStamp\", \"RealmId\" DESC)"
);
```

Joonis 4, hüpertabelite loomiseks kood

TimeScaleDB funktsionaalsuse kasutamiseks on vaja hüpertabelid.

Kood loob uusi hüpertabeleid, kuid neile pääseb juurde samade meetoditega nagu enne tabelite hüpertabeliteks teisendamist. See ka mõistab, et Entity Framework Core'i puhul ei ole tabel kuidagi muutunud ja see käitub samamoodi nagu tavaline tabel. Kuid hoolimata asjaolust, et interaktsioon tabelitega jääb samaks, on andmebaasi sisemise toimimise seisukohalt tabel muutunud.

Hüpertabel võimaldab kiiremat tööd andmetega, mis sisaldavad aegridu andmete ajajaotuse tõttu. Seetõttu võib isegi selline lihtne samm nagu tavalise tabeli teisendamine hüpertable'iks oluliselt kiirendada andmebaasi. [21]

3.3.3 Mudelid ja recordid.

Andmehalduse lihtsustamiseks teisendame JSON-stringina saadud API vastused objektideks. RestSharp teisendab objektid automaatselt ja me kasutame kirjeid objekti "skeemina". Kuna osa JSON-i andmeid on pesastatud ja RestSharpi deserialiseerimine on automaatne, võidakse teha täiendavaid kirjeid, mis esindavad JSON-i pesastuse järgmist taset. Varem näidatud tabelid on "tasased" versioonid, kus pesastatud andmed teisendatakse ühe taseme võtme-väärtuste paarideks. Seetõttu on mudelid ja RestSharpi kirjed koodi kasutamise hõlbustamiseks ja ka koodi paindlikkuse huvides eraldi objektid. See tähendab, et kood ei deserialiseeri päringu andmeid otse mudeliklassi objektiks, vaid deserialiseeritakse esmalt vahekirjeks, mis seejärel kaardistatakse mudeliobjektiga. Samuti suurendab see lähenemisviis koodi üldist paindlikkust. Kui tulevikus on vaja kirjetelt rohkem andmeid hankida, on vaja ainult kirjeid ja vastendust muuta, siis mudeli koodiosa ise ei muutu.

3.3.4 Automapper

Selleks, et kõik ei saaks kaardistajat käsitsi kirjutada, on olemas tööriist nimega *AutoMapper*. See raamatukogu on avatud lähtekoodiga ja loata litsentsitud [22], mille eesmärk on kaardistada andmeid ühelt objektilt teisele.

Kohandatud AutoMapperi profiili loomiseks peame looma uue klassi, mis pärib *Profile* ja rakendage meetodit *AuctionMappingProfile()*. Selle meetodi puhul määrab kasutaja meetodi abil lähte- ja sihtobjektid, mis kaardistatakse. *Automapper* proovib sarnaseid väljanimesid automaatselt sobitada, kuid kui väljade nimed ei ühti, tuleb seosed siiski käsitsi määrata.

Lõpuks loodud profiil *AutoMapper* tuleb lisada .Net Core teenustele kasutades builder.Services.AddAutoMapper() et seda saaks välja kutsuda sõltuvuse süstimise kaudu, kasutades *IMapper*.

3.4 Skannerid

Skannerid tähendavad programmi osa, mis küsib perioodiliselt andmeid oksjoni kohta, töötleb neid ja salvestab andmebaasi. Nüüd, kui andmebaas, API päringud ja kaardistaja töötavad, saab alustada oksjonimaja skannerite arendamist. Vajame teha kaks skannerit – ühte hindade jaoks, mida uuendatakse kord tunnis ja teist kaupade jaoks, mida uuendatakse kord nädalas. Nädalane periood on võetud tänu sellele, et igal kolmapäeval teevad WoW arendajad hooldustöid, mille käigus mängu uuendatakse. See tähendab, et üksuse teave võib samuti muutuda või lisanduda võib uusi üksusi. Et mitte luua kahte lisarakendust, on võimalus kasutada klassi **BackgroundService**. See on klass, mille pakub .Net Core taustaülesannete jaoks, mis sobib pikkade ja perioodiliselt töötavate protsesside jaoks [23].

Täpsemalt on BackgroundService IHostedService'i alamklass. IHostedService'il on kaks meetodit: StartAsync taustaprotsessi käivitamiseks, StopAsync protsessi peatamiseks, kui host teostab graatsilist sulgemist. BackgroundService'is on arendaja mugavuse huvides ja mitmete asünkroonsete meetoditega seotud vigade vältimiseks need kaks meetodit juba rakendatud ning kasutusmugavuse huvides on lisatud uus meetod – ExecuteAsync. Uus BackgroundService'i meetod kutsutakse välja taustateenuse käitamiseks ja erinevalt StartAsyncist ei blokeeri rakendust käivitamist, kuna hostitud teenuseid käitatakse järjestikku ja täiendavaid teenuseid ei käivitata enne, kui StartAsync on lõpetatud.

3.4.1 Konfiguratsioonid

Enne skanneri koodi juurutamist peab esitada ka seadete andmed. Näiteks rakendus peab teadma, milliseid connected-realm'i ja piirkond skannida, Blizzard API *Client Id, Client Secret* jne. Seadetest andmete saamiseks skanneriklassis tuleb deklareerida konstrueerijas *IConfiguration*. See kutsub konfiguratsiooniobjekt, kasutades sõltuvussüsti, millest andmeid hankite appsettings.json'ilt [24]. Joonistel 5 on võimalus vaadata, kuidas konfiguratsioonide lõplik versioon välja näeb.

Esimene parameeter on *ConnectionStrings*, mis määrab kõik andmebaasiga ühenduse loomiseks vajalikud andmed, mida vajame skanneri tulemuste salvestamiseks. *Logging* parameeter on vajalik selleks, et näidata, kui palju andmeid kasutaja soovib rakenduse töötamise ajal näha. Iga logimise allika jaoks saab määrata eraldi üksikasjalikkuse taseme. See on kasulik, kui kasutaja näiteks soovib saada ainult veateateid või kui ta soovib näha rakenduse toimivuse kohta üksikasjalikumat aruannet. *AllowedHosts* parameeter on mõeldud pigem juhuks, kui kasutaja soovib rakendust hostida mitte ainult kohapeal, vaid ka teistele inimestele. Seda parameetrit kasutatakse usaldusväärsete hostide loendi piiramiseks, millelt päringuid saab vastu võtta. See parandab rakenduse turvalisust. Hosti filtreerimise vahevara vaikimisi ei tööta. Vahevara sisselülitamiseks tuleb kirjutada veebiadressid. *AllowedHosts* saate määrata kehtivad hostinimed või IP-aadressid, millelt päringud võivad tulla. [25]

Lisaks API paroolidele ja valdkonna connected realmi Id-ile peame lisama ka *Locale* ja *Region* parameetrid. *Locale* on kasutatud lokaliseerimiseks. Kui kasutaja seda parameetrit ei sisesta, Blizzard API kuvab iga keele nime, mis muudab päringu suuremaks ja API reageerimisaja aeglasemaks. Suurem päring tähendab ka täiendavat andmetöötlust, seega on parem lähenemine võimalusel alati *Locale* parametrit sisestada [26]. *Region* parameeter vastutab piirkonna eest. Piirkonna kasutatakse selleks, et määrata, millisest *namespace*'ist andmed üles laadida.

```
{
  "ConnectionStrings": {
    "NpgsqlConnection":
"Host=localhost;Port=5432;Database=yourdatabasename;Username=yourusername;Pas
sword=yourpassword;IncludeErrorDetail=true"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "OAuth2Credentials": {
    "ClientId": "ef3e3d3c1bf843528e66c9db9b640ebb",
    "ClientSecret": "Gn4BDUrgAB0s6FzOwOfYLKai5AipYrxG"
  },
  "RequestParameters": {
    "Region": "eu",
    "RealmId": "3391,1396",
    "Locale": "en_US"
  }
}
```

Joonis 5, appsettings.json'i kood

3.4.2 Rakendamise nüansid

Mõlema skanneri kokkuvõtet saab lihtsustada järgmiste sammudeni: hankige API-st kaubainfo või hindade andmed, kaardistage ja salvestage need andmebaasi, hankige API-st mittekaupade andmed, kaardistage ja salvestage need andmebaasi. Ainus probleem teenuste juurutamise ajal oli see, et ma ei saanud konstruktori kaudu andmebaasi konteksti kutsuda, kuna see tekitas vea. Seda seetõttu, et kontekst, mida proovime kutsuda, on *Scoped* teenus [27] ja BackgroundService on *Singleton* [28]. Üldiselt peetakse Singleton'is Scoped teenuse jõuga kasutamist halvaks mõtteks, kuna see võib põhjustada ootamatuid vigu ja mälulekkeid [29]. Siiski on stsenaariume, näiteks andmebaasi kasutamine kontekstis, kui peate siiski looma Scope'i Singletonis. Seda tüüpi toimingute jaoks on .Net Core'il **IServiceScopeFactory**, mis loob skoobi ja Scoped teenused. Rakendustsükli rakendamiseks võite kasutada tsükli lõpus olevat while-tsüklit ja Task.Delay(), et täitmistsüklite vahel oleks viivitus.

Taustteenuste rakendamiseks rakenduses tuleb need failis **program.cs** määrata, lisades rea builder.Services.AddHostedService<TService>(), kus *TService* – klass, mis pärib klassist *BackgroundService*. Nüüd aktiveerib .Net Core need rakenduse käivitumisel ise.

3.5 Kasutajaliides

Arendusprotsessi lõppedes liigume edasi esiplaanile. Peame looma kolm lehte - Kodu, kus saate valida seadetes määratud connectedrealmi, ühendatud realmi lehe, kus kuvatakse kõik tooted ja võimalus otsida märksõna järgi. Ja viimane, tooteleht, kus kuvatakse hinnadünaamika.

Veebirakendus kasutab **Model-View-Controller** muster. Seda disainimustrit rakendatakse meie puhul järgmiselt:

- **Model** - Meie puhul on see klass, mis annab andmed *View* osale ja muudab selle olekut *Controller* mõjul.
- **View** - vastutab mudeliandmete kasutajale kuvamise eest ning edastab ka kasutaja interaktsioonid *Controller*'ile.
- **Controller** - Klass, mis võtab kasutajalt *View* kaudu vastu andmeid, töötleb neid ettenähtud loogika abil ja tagastab töödeldud andmed *Model* objekti kaudu.

3.5.1 Avaleht

Avalehel peame hankima konfiguratsioonid ja API taotluste eest vastutava klassi. Nõuame seadetes määratud connected realms'ide loendit ja saame API kaudu ühendatud valdkonnaga seotud valdkondade nimed. Realm'ide nimi on hüperlink, mis suunab valitud connected realmi oksjonilehele. Kui connected realms pole seadetes määratud, peaks rakendus taotlema piirkonnast kõiki ühendatud piirkondi ja ka nende kirjeldust ning märkima selle kõrval nende ID. See tööriist on loodud kasutaja mugavuse huvides, kuna puudub teave connected realms'ide loendi kohta veebis saadaolevate valduste nimedega.

3.5.2 Connected Realm'i oksjoni leht

Lehekülje kiireks ja mugavaks kuvamiseks tuleks teha abiklass koos lehekülgede, filtreerimise ja andmete sorteerimisega. Enne Commodities ja NonCommodities andmete ühendamist filtreerime need aja järgi, et saada ainult uusimad hinnad, seejärel filtreerime NonCommoditiesi oksjonipartiid ühendatud valdkonna ID järgi. Nüüd peame kasutama ainult context.Join() ning looge selles CommodityAuctioni ja CommodityInfo liit. Seejärel ühendame NonCommodityAuctioni ja NonCommodityInfo. Nüüd ühendage need kaks täielikku andmete loendisse, kasutades .Concat(). Nüüd on andmed ette

valmistatud ja valmis sorteerimiseks või filtreerimiseks, kasutades .OrderBy() või .Where(), ja .Skip(y) ja .Take(x) abil saad x objekti alates y positsioonist. Lehe kontrolleris tasub tagastada mitte kombineeritud kaupade ja mittekaupade kogu, vaid spetsiaalselt loodud objekt, olgu selle nimeks *AuctionSlotModel*. See AuctionSlotModel ei salvesta mitte ainult oksjonipartiide kogumit, vaid ka muud teavet, mis võib olla kasulik, nagu lehekülje number, lehtede koguarv ja näiteks sõnajada, mida sõna otsimiseks kasutame. Nii pakume paindlikku lähenemist andmete edastamisele ilma täiendavaid lahendusi välja mõtlemata, näiteks *ViewData* väärtusi kasutades.

Samuti lisame sellele lehele otsingufunktsiooni. Kasutaja mugavuse huvides teisendame otsimisel otsingusõna ja kõik stringid, mida kontrollime, väiketähtedeks. See muudab otsimise lihtsamaks, kuna WoW-nimesüsteem järgib *Title Case*'i reeglit. Lõpuks lisame kasutaja mugavuse huvides *Wowhead Tooltips*. Pärast skripti installimist ilmub tööriistale üksuse ID määramisel selle nime lähedale ikoon asjadega. Kui kasutaja hõljutab kursorit üksuse nime kohal, kuvatakse tööriistavihje ja nimi muudab oma värvi olenevalt üksuse haruldusest mängus [30]. Need üksuste nimed toimivad ka hüperlingina, mis suunab ümber asjade lehele ja annab lehe kontrollerile asjade ID ja ühendatud domeeni.

3.5.3 Item Page

Kauba lehe jaoks saame kasutada sama tabeli ühendamise loogikat, mille lõime oksjonipartii lehe jaotises saaleabi klassi kaudu, kuid nüüd sorteerime andmed mitte ühendatud välja ID ja viimase skannimise kuupäeva, vaid ühendatud välja järgi. ja kauba ID. Kauba andmed kuvatakse kahel kujul. Esimene vorm on diagramm, kus üks telg on aeg ja teine on palju kulda. Teine vorm on tabel üksikasjalikuma teabe vaatamiseks. Eseme maksumus tuleb ümber arvestada kuldmüntideks, sest maksumuse kuvamine vases on WoW mängija jaoks ebamugav. Tabeleid saab teha HTML-i abil, kuid graafika nõuab lisatööriistu.

Loome teise klassi, mis toimib mudelina, kus anname vajalike partiide nimekirja, välja ID ja elemendi. Hinnadünaamika diagrammi renderdamiseks kasutatakse Google'i diagramme. Google Chartsil on nii hästi kirjutatud dokumentatsioon kui ka valmis mallid erinevat tüüpi diagrammide jaoks. [31]

3.6 Docker konteineriseerimine

Järgmine samm on rakenduse konteineriseerimine. Konteinerisse paigutamist on vaja, nii et kasutaja ei pea muretsema keskkonna pärast ja teeb seadistamise ja käivitamise lihtsamaks. Rakenduse konteineriseerimiseks peame tegema kahte sammu: looma *DockerFile*'i ja lisama *docker-compose*, mida varem kasutasime ainult andmebaasi jaoks.

Docker-compose'i esmakordseks käivitamiseks peate käivitama käsu ‘docker-compose up’ kataloogist, kus asub fail *docker-compose.yml*.

3.6.1 DockerFile

Dockerfile on vajalik konteineri kujutise loomiseks, mida kasutatakse docker-compose’is. Teisisõnu võib DockerFile'i nimetada dokkipildi loomise skeemiks. Selles failis määrame, millist .Net Core'i baaspildi versiooni kasutatakse, millist porti kasutatakse rakendusega suhtlemiseks, milliseid faile kopeerida sõltuvuste tuvastamiseks ja pildi loomiseks [32].

3.6.2 Docker-compose lõpetamine

Joonisel 6 on võimalus vaadata docker-compose’i valmis koodi. Kuna *appsettings.json* viitas andmebaasile ilma konteineris viibimata, määrame *environment*-is uue ühendusstringi, mis ühendub otse andmebaasi konteineriga. Erinevalt TimescaleDB-st, mis kasutab parameetrit *image* ja laadib juba tehakse valmispildist, rakenduses parameeter *image* ainult annab pildile nime, et Docker ei määraks juhuslikku. Pildi jaoks kasutatakse välja *build*, kus määrame pildi koostamiseks kasutatava DockerFile'i suhtelise asukoha ja nime.

```
version: '3.9'
services:
  auctionhousetracker-webapp:
    container_name: AuctionHouseTracker-webapp
    image: auctionhousetrackerwow
    build:
      context: ./WebAuctionHouseTracker/
      dockerfile: DockerFile
    environment:
      -
"ConnectionStrings__NpgsqlConnection=Host=TimeScaleDB;Port=5432;Database=your
databasename;Username=yourusername;Password=yourpassword;IncludeErrorDetail=t
rue"
    ports:
      - "80:80"

  timescaledb:
    container_name: TimeScaleDB
    image: timescale/timescaledb:latest-pg13
    environment:
      POSTGRES_PASSWORD: yourpassword
      POSTGRES_USER: yourusername
      POSTGRES_DB: yourdatabasename
    volumes:
      - ./data:/var/lib/postgresql/data
    ports:
      - "5432:5432"
```

Joonis 6, docker-compose'i kood

See lõpetab konteineriseerimise koodiosa kirjutamist, nüüd piisab käsu 'docker-compose up' käivitamisest, et kõik tööle saada. Varem loodud timescaledb-koodi osa kuidagi ei muuda.

3.7 Litsentsi valik

Tarkvara litsentsimine on mis tahes tarkvaratoote arendamise ja kasutamise oluline aspekt. Litsentsid määratlevad tarkvara kasutamisega seotud õigused ja kohustused ning selle levitamise ja kasutamise piirangud. Sõltuvalt litsentsi tüübist võivad arendajatel ja kasutajatel olla tarkvara kasutamiseks erinevad õigused, samuti selle levitamise ja muutmisega seotud piirangud ja kohustused. Kuna arenduse käigus kirjeldati kõiki kasutatud tehnoloogiaid ja ka nende litsentse, siis praeguses etapis võib väita, et avatud lähtekoodiga litsentse saab probleemideta kasutada.

Kuna projekt antakse välja loavaba litsentsi all, tasub valida üks kahest hetkel populaarseimast IT-tehnoloogiast: MIT või Apache 2.0. MIT poolt hääletati mitmel järgmisel põhjusel:

- Apache litsentsi on tavainimese jaoks palju raskem lugeda, kuna see sisaldab keerulist juriidilist keelt ja on ka palju pikem kui lühike MIT-litsents.

- Apache 2.0 litsents nõuab, et kõik selle litsentsi alusel tehtud koodi suuremad muudatused tuleb dokumenteerida. See on väga suur puudus, kuna see on lisakoormus neile, kes kavatsevad rakendust muuta, seega on see valik märkimisväärne puudus.

Arvestades ülaltoodud tegureid, otsustati mängukogukonna poolt lähtekoodi muutmise lihtsuse ja mugavuse **kasuks kasutada MIT-i litsentsi**.

MIT-litsents on tasuta tarkvaralitsents, mis võimaldab teil tarkvara vabalt kasutada, muuta, levitada ja müüa või kasutada seda ärilistel eesmärkidel, eeldusel, et säilitate originaallitsentsi koopia ja loobute garantiidest. MIT-litsents ei sisalda piiranguid tarkvara kasutamisele, levitamisele ega muutmisele ning võimaldab koodi kasutada nii tasuta kui ka omandiõigusega kaitstud projektides.

MIT-litsentsi põhiidee seisneb selles, et arendajatel on vabadus koodi kasutada ja muuta, ilma et neid piiraks nende endi intellektuaalomandi õigused. Samuti saavad nad koodi kombineerida teiste projektidega ja isegi kasutada seda kommertsprojektides, säilitades samal ajal originaallitsentsi koopia. See muudab MIT-i litsentsi meie tarkvara jaoks üheks parimaks litsentsiks.

Selleks tuleb repositooriumisse lisada tekstifail nimega LICENSE ning sinna sisestada dokument vastavalt LISA 2 näitele.

4 Töö tulemused

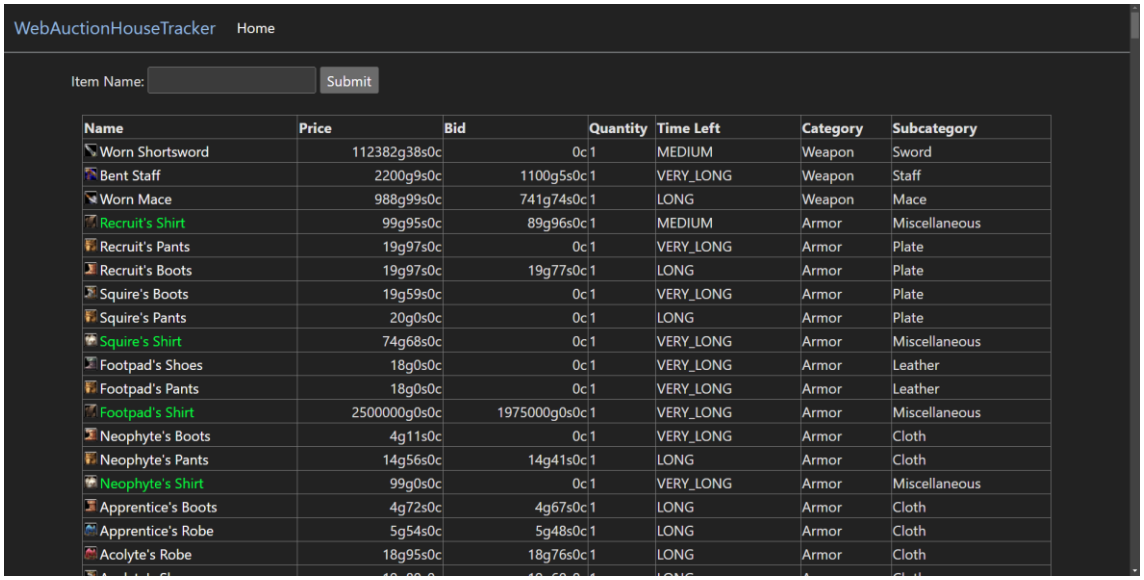
Pärast arenduse valmimist tasub tulemused kokku võtta ja kontrollida, kas kõik funktsionaalsed nõuded on täidetud, kui hästi on need mittefunktsionaalsete nõuetega toime tulnud.

4.1 Loodud funktsionaalsus

Selle tulemusena oleme loonud rakenduse, mis küsib Blizzard API-lt andmeid, töötleb neid ja salvestab andmebaasi. Andmeid küsitakse kahe taustprotsessi abil:

- Esimene taustprotsess küsib kaupade hinnaloendit, kuid API ei anna neis olevate üksuste kohta teavet.
- Teine taustprotsess küsib üksuste kohta teavet, mida on vaja esimesest taustteenusest saadud üksuste kirjeldamiseks.

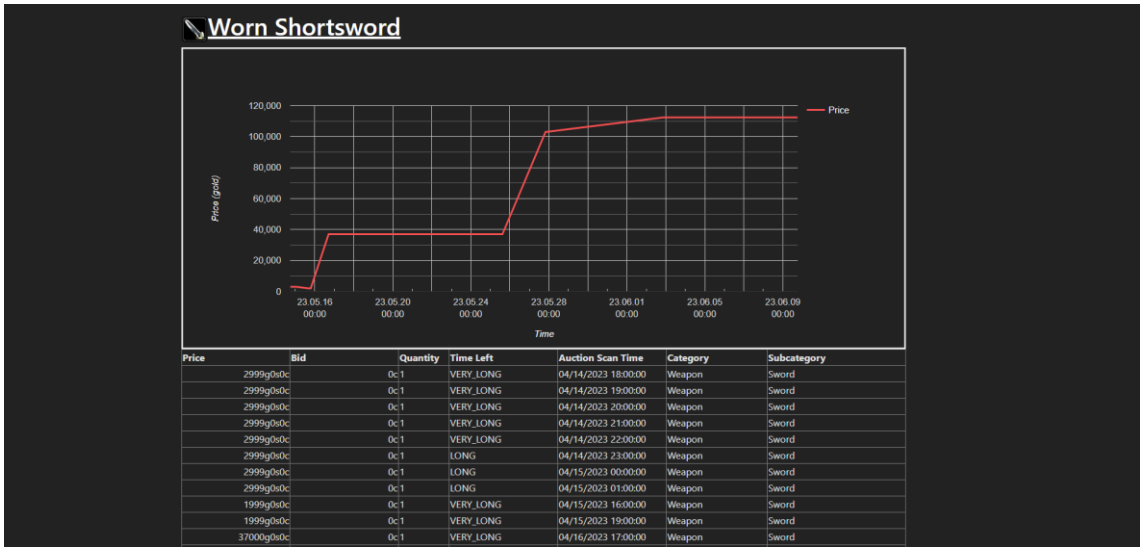
Saadud andmeid saab vaadata realmi lehel, vt Joonis 8. Lehel on pealkirjaotsing ja igal lehel kuvatakse 500 üksust. Lehel on nimeotsing ja igal lehel kuvatakse 500 toodet koos ostuhinna ja pakkumisega, kui see on olemas. Samuti on kaupade puhul märgitud kategooria, alamkategooria ja partii kasutusiga. Kasutaja mugavuse huvides stiliseeritakse üksuse nimi WoW liidese jaoks, kasutades Wowhead Tooltips .



Name	Price	Bid	Quantity	Time Left	Category	Subcategory
Worn Shortsword	112382g38s0c	0c	1	MEDIUM	Weapon	Sword
Bent Staff	2200g9s0c	1100g5s0c	1	VERY_LONG	Weapon	Staff
Worn Mace	988g99s0c	741g74s0c	1	LONG	Weapon	Mace
Recruit's Shirt	99g95s0c	89g96s0c	1	MEDIUM	Armor	Miscellaneous
Recruit's Pants	19g97s0c	0c	1	VERY_LONG	Armor	Plate
Recruit's Boots	19g97s0c	19g77s0c	1	LONG	Armor	Plate
Squire's Boots	19g59s0c	0c	1	VERY_LONG	Armor	Plate
Squire's Pants	20g0s0c	0c	1	LONG	Armor	Plate
Squire's Shirt	74g68s0c	0c	1	VERY_LONG	Armor	Miscellaneous
Footpad's Shoes	18g0s0c	0c	1	VERY_LONG	Armor	Leather
Footpad's Pants	18g0s0c	0c	1	VERY_LONG	Armor	Leather
Footpad's Shirt	2500000g0s0c	1975000g0s0c	1	VERY_LONG	Armor	Miscellaneous
Neophyte's Boots	4g11s0c	0c	1	VERY_LONG	Armor	Cloth
Neophyte's Pants	14g56s0c	14g41s0c	1	LONG	Armor	Cloth
Neophyte's Shirt	99g0s0c	0c	1	VERY_LONG	Armor	Miscellaneous
Apprentice's Boots	4g72s0c	4g67s0c	1	LONG	Armor	Cloth
Apprentice's Robe	5g54s0c	5g48s0c	1	LONG	Armor	Cloth
Acolyte's Robe	18g95s0c	18g76s0c	1	LONG	Armor	Cloth

Joonis 7, realmi leht

Valitud toote lehel kuvatakse hindade ajalugu: Google Chartsi abil koostatud graafiku kujul ja tabelina. Graafik kuvab andmed õigesti, isegi kui mingil põhjusel andmeid teatud ajaperioodi jooksul pole.



Price	Bid	Quantity	Time Left	Auction Scan Time	Category	Subcategory
2999g0s0c	0c	1	VERY_LONG	04/14/2023 18:00:00	Weapon	Sword
2999g0s0c	0c	1	VERY_LONG	04/14/2023 19:00:00	Weapon	Sword
2999g0s0c	0c	1	VERY_LONG	04/14/2023 20:00:00	Weapon	Sword
2999g0s0c	0c	1	VERY_LONG	04/14/2023 21:00:00	Weapon	Sword
2999g0s0c	0c	1	VERY_LONG	04/14/2023 22:00:00	Weapon	Sword
2999g0s0c	0c	1	LONG	04/14/2023 23:00:00	Weapon	Sword
2999g0s0c	0c	1	LONG	04/15/2023 00:00:00	Weapon	Sword
2999g0s0c	0c	1	LONG	04/15/2023 01:00:00	Weapon	Sword
1999g0s0c	0c	1	VERY_LONG	04/15/2023 16:00:00	Weapon	Sword
1999g0s0c	0c	1	VERY_LONG	04/15/2023 19:00:00	Weapon	Sword
37000g0s0c	0c	1	VERY_LONG	04/16/2023 17:00:00	Weapon	Sword

Joonis 8, toode leht

Valitud realmid, valdkonnad, lokaat ja paroolid on konfigureeritud ühes kohas, failis nimega *appsettings.json*. Rakendus ja andmebaas on mugavalt konteinerisse paigutatud ning pärast installimist ühe klõpsuga käivitatavad. Kui *appsettings.json*'is ei ole jälgimiseks määratud *connected realm*'id, rakendus küsib connected realm loendit ja esitab need avalehel, vt joonis 9. Vasakul real on realmi nimed ja paremal real on *connected realm*'i id.

Realm name	Realm id
Garona, Sargeras, Ner'zhul	509
Vol'jin, Chants éternels	510
Arak-arahm, Rashgarroth, Kael'thas, Throk'Feroth	512
Dethecus, Theradras, Onyxia, Mug'thol, Terrordar	531
Lothar, Baelgun, Azshara, Krag'jin	570
Arthas, Blutkessel, Wrathbringer, Durotan, Kel'Thuzad, Vek'lor, Tirion	578
Blackmoore, Tichondrius, Lordaeron	580
Blackrock	581
Thrall, Kargath, Ambossar	604
Nefarian, Gilneas, Destromath, Ulduar, Mannoroth, Gorgonnash, Nera'thor	612
Kor'gall, Executus, Shattered Hand, Bloodfeather, Terokkar, Saurfang, Darkspear, Burning Steppes	633
Khadgar, Bloodhoof	1080
Kul Tiras, Alonsus, Anachronos	1082
Dentarg, Tarren Mill	1084
Moonglade, Steamwheedle Cartel, The Sha'tar	1085
Emeriss, Twilight's Hammer, Bloodscalp, Crushridge, Agamaggan, Hakkar	1091
Drak'thul, Burning Blade	1092
Scarshield Legion, Sporeggar, Earthen Ring, Defias Brotherhood, Ravenholdt, Darkmoon Faire, The Venture Co	1096
Ysera, Malorne	1097
Malygos, Malfurion	1098
Rexxar, Alleria	1099
Anetheron, Kil'jaeden, Rajaxx, Festung der Stürme, Gul'dan, Nathrezim	1104
Nazjatar, Zuluhed, Dalvengyr, Aman'Thul, Frostmourne, Anub'arak	1105
Die Arguswacht, Die ewige Wacht, Die Todeskrallen, Das Syndikat, Der abyssische Rat, Kult der Verdammten, Das Konsortium, Die Silberne Hand	1121

Joonis 9, connected-realmi loend

4.2 Kettaruumi ja töömälu kasutamine

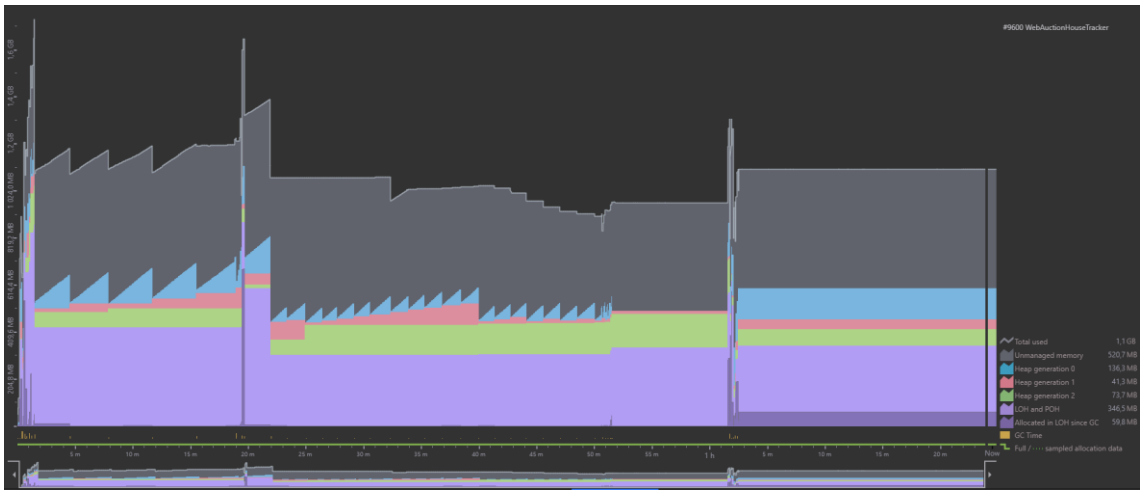
Varem mittefunktsionaalsetes nõuetes oli täpsustatud, et rakendus peaks tarbima võimalikult vähe kettaruumi ja mälu. Mälu ja ketta optimeerimine võib jätkuda ka pärast rakenduse vabastamist, aga igal juhul tuleb uurida, kui ressursimahukas rakendus on ja kui hästi rakendus esimesel valmisversioonil ressursse säästab. Skannerite kõige nõudlikumad parameetrid on tavaliselt tarbitud operatiivmälu ja kettaruum, seega kontrollimine tuleb keskendada nendele kahele parameetrile.

4.2.1 Operatiivse mälu mõõtmine

Mälukasutuse mõõtmiseks kasutatakse dotMemory. dotMemory on tööriist mitmesuguste .NET-i ja .NET Core'i tarbitud mälu hindamiseks [33]. DotMemory variatsiooni kasutatakse JetBrains Rideri arenduskeskkonna pistikprogrammina. Kui profiiliseanss on algatatud, rullib ajaskaala graafik reaalajas lahti, näitlikustades, kuidas rakendus profiilide koostamise ajal mälu kulutas.

Mälukulu mõõtmiseks tehakse kaks mõõtmist - Esimeses testis mõõdetakse üks tsükkel hindade ja esemete info skaneerimisest, teises jäetakse rakendus pikemaks ajaks, seega on kolm hindade skaneerimise tsüklid.

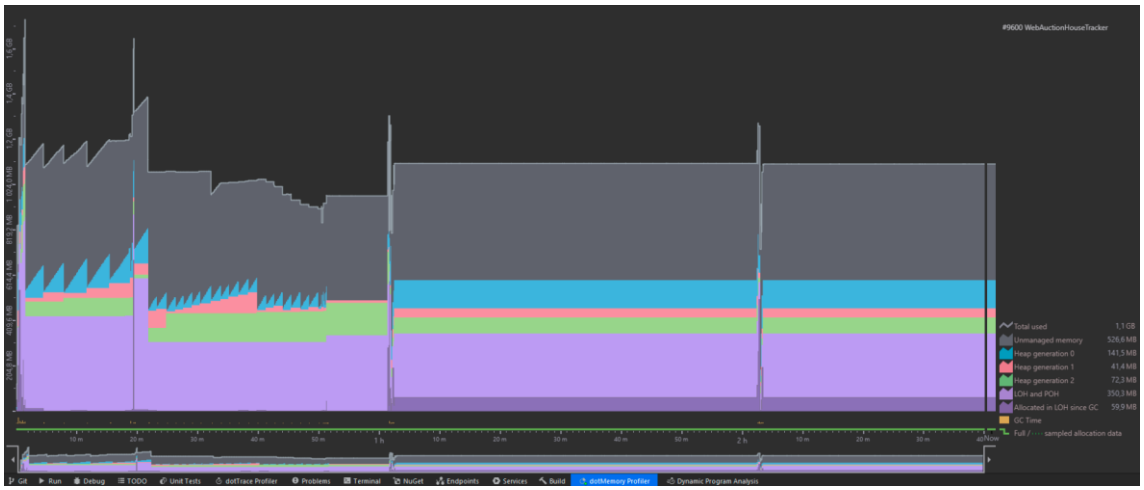
Joonisel 10 on illustreeritud esimese testi mälutarbimise graafikut. Graafikult on näha, et kohe alguses, kui rakendus samal ajal käivitatakse, skannitakse hindu ja kaupu, saavutab mälujaotuse haripunkt kuni ligikaudse märgini 1,7 GB. Pärast seda väheneb tarbitava mälu maht ja jääb keskmiselt 1.1 GB piiresse.



Joonis 10, Esimene mälukontroll

Arvestades ülaltoodud graafikut, võime öelda, et rakendus ei tarbi liiga palju mälu. Siiski on ruumi optimeerimiseks, samuti tippude pehmendamiseks. Kuid pikaajalise mälukulu vaatamiseks tasub teha ka mitme skaneerimise hinnatest.

Joonisel 11 on illustreeritud esimese testi mälutarbimise graafikut. Graafikult on näha, et pärast teist skannimist jäi tarbitud mälumaht 1,1 GB piiresse, seega olulisi muutusi mälukasutuses ei toimunud.



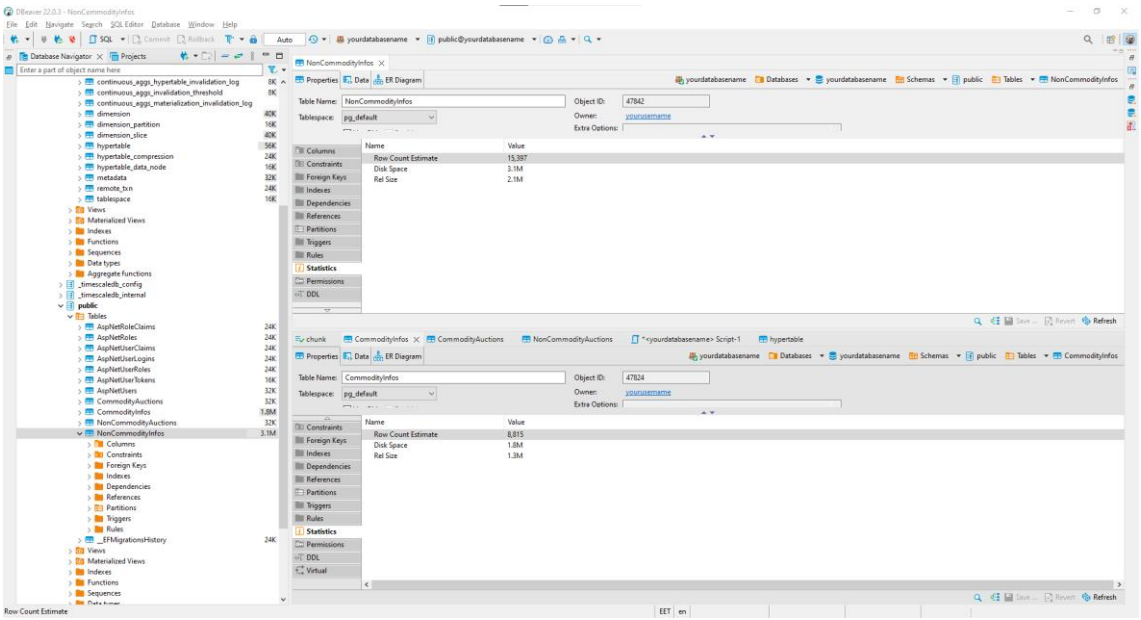
Joonis 11, teine mälukontroll

Sellest tulenevalt on võimalus öelda, et rakendus kulutab üsna vähe mälu ning mälutipud tekivad harva ja on äärmiselt lühiajalised. Siiski on veel ruumi optimeerimiseks ja edaspidi võib projekt proovida mälutarbimise tippe siluda.

4.2.2 Kettaruumi mõõtmine

Kettaruumi tarbimise mõõtmine on äärmiselt oluline, kuna see võimaldab ennustada, kui palju mälu kulub andmete salvestamiseks. Kauba info ei kasva koos skaneeringute arvuga, seega piisab talle ühest skaneeringust. andmebaasi mõõtmiseks ja sellega suhtlemiseks kasutatakse DBeaverit. Dbeaver on andmebaasi haldustööriist, mis pakub palju lisafunktsioone. Üks neist funktsioonidest on võimalus vaadata, kui palju andmeid tabel kettal võtab.

Joonisel 12 on näha, et teabeüksused võtavad enda alla 2,1 megabaiti ja 1,3 megabaiti. Kui võtta arvesse kogutarbimine, sealhulgas indeksid ja nii edasi, siis koguteave kaupade kohta võtab 4,9 megabaiti See on vähe kettaruumi ja aja jooksul vaevalt see kasvab.



Joonis 12, teabeüksuste kettaruumi kasutus

Kuna hindadega seotud tabelid on hüpertabelid, ei näe programm tegelikku kasutatud kettaruumi. Õnneks TimescaleDB on pakkunud tööriista ja dokumentatsiooni funktsiooniga *hypertable_size()* hüpertabelite tarbitud kettaruumi mõõtmiseks. Funktsioon esitab andmebaas tabeli täissuuruses koos kõigi seotud andmetega [34]. Hüpertabeli suurus kuvatakse baitides. Mõõtmised tehakse kolm korda: 10 tundi, 20 tundi ja 30 tundi skaneeritud andmetega. Rakenduse töötamise ajal jälgiti kahte valdkonda.

Skaneerimise tulemuste põhjal (vt. Tabel 3), 10-20 tunni vahel andmemaht kasvas 7,3 megabaiti ja 20-30 tunni vahel 5,4 megabaiti. See tähendab, et andmeid saab salvestada ka tavakasutaja arvutisse ja need ei võta liiga palju ruumi, sest eeldatav kettaruumi tarbimine on umbes 6-7 megabaiti 10 tunni jooksul. Tulevikus on võimalik kettaruumi kasutust veelgi suurem optimeerida, näiteks rakendades andmete tihendamist.

Tabel 3, kettaruumi kasutamine sõltuvalt skannimis tundide arvust

Skaneeritud tundide arv	Commodity kaupa tabeli suurus, baitid	Non commodity kaupa tabeli suurus, baitid
10	45 367 296	139 493 376
20	46 546 944	145 579 469
30	47 161 344	150 372 352

5 Kokkuvõtte

Töö eesmärgiks oli avatud lähtekoodiga rakendus "World of Warcrafti" mängusisese oksjoni jälgimiseks loomine. Nende eesmärkide põhjuseks oli oksjonit jälgivate saitide arvu märkimisväärne vähenemine. Oksjonite "langemine" tõi kaasa 2022. aasta oksjoni struktuuri ümbertöötamise

Töö valmimisel kasutatakse selliseid tehnoloogiaid ja teeke nagu: .Net Core 6, Postgres, TimescaleDB, Docker, Wowhead Tooltips, Google Charts, RestSharp, AutoMapper. Loodud rakendust levitatakse avatud lähtekoodiga MIT litsentsi all GitHubi platvormil.

Töö tulemusena töötati välja veebirkendus . Rakendus vastab taotlus kõigile taotlemise alguses seatud nõuetele. Rakendus küsib automaatselt andmeid Blizzard API-lt, töötleb neid ja salvestab need andmebaasi. Saadud andmeid saab nii saidil vaadata kui ka nime järgi filtreerida. Hindade ja kauba skannimine toimub ilma kasutaja abita automaatselt perioodiliste intervallidega. Kaubalehel kuvatakse hindade dünaamika graafiku ja tabeli kujul kogu skannimise aja kohta. Skaneeritav piirkonna ja connected realmi id sisestatakse enne rakenduse käivitamist seadete kaudu. Koos rakendusega on lisatud ingliskeelsed juhised rakenduse seadistamiseks ja esmakordseks käivitamiseks.

Töö edasiarendusena on võimalik edasi optimeerida rakenduse kiirust, mälu- ja kettaruumi kasutamist. Tuleb täiustada saidi kujundust ning lisada lisafunktsionaalsust, näiteks kaubakategooriate kaupa sorteerimine, filtrite lisamine ja mugavam navigeerimine.

Kasutatud kirjandus

[1] Erorus, "Oribos exchange auction house online tracker," [Online]. Available: https://oribos.exchange/. [Accessed 1 May 2023].

[2] Erorus, "The Undermine Journal," [Online]. Available: https://theunderminejournal.com/. [Accessed 25 March 2023].

[3] Erorus, "Newsstand," [Online]. Available: https://github.com/erorus/newsstand. [Accessed 4 May 2023].

[4] BLIZZARD ENTERTAINMENT, INC., "Using OAuth," BLIZZARD ENTERTAINMENT, INC., [Online]. Available: https://develop.battle.net/documentation/guides/using-oauth. [Accessed 5 May 2023].

[5] BLIZZARD ENTERTAINMENT, INC., "World of Warcraft Realm Options," BLIZZARD ENTERTAINMENT, INC., [Online]. Available: https://us.battle.net/support/en/article/286374#]. [Accessed 4 May 2023].

[6] BLIZZARD ENTERTAINMENT, INC., "Shadowlands 9.2.7 Update Notes—Now Live!," BLIZZARD ENTERTAINMENT, INC., 16 06 2022. [Online]. Available: https://worldofwarcraft.blizzard.com/en-us/news/23833174/shadowlands-927-update-notes-now-live. [Accessed 5 May 2023].

[7] BLIZZARD ENTERTAINMENT, INC., "Immediate change to Auction APIs for Commodities with 9.2.7," BLIZZARD ENTERTAINMENT, INC., [Online]. Available: https://us.forums.blizzard.com/en/blizzard/t/immediate-change-to-auction-apis-for-commodities-with-927/31522.

[8] BLIZZARD ENTERTAINMENT, INC., "Connected Realms," BLIZZARD ENTERTAINMENT, INC., [Online]. Available: https://us.battle.net/support/en/article/14296#:~:text=Connected%20Realms%20are%20a%20set,the%20realms%20connected%20to%20it. [Accessed 5 May 2023].

[9] BLIZZARD ENTERTAINMENT, INC., "Patch 5.4 Feature Preview: Connected Realms," BLIZZARD ENTERTAINMENT, INC., 5 8 2013. [Online]. Available: https://web.archive.org/web/20130808061356/http://us.battle.net/wow/en/blog/10551009/Patch_54_Feature_Preview_Connected_Realms-8_5_2013. [Accessed 5 May 2023].

[10] P. Wayner, "Database trends: The rise of the time-series database," VentureBeat, 15 01 2021. [Online]. Available: https://venturebeat.com/business/database-trends-the-rise-of-the-time-series-database/. [Accessed 5 May 2023].

[11] McObject LLC., "License," McObject LLC., [Online]. Available: https://www.mcobject.com/docs/Content/License.htm#:~:

text=eXtremeDB%20is%20commercial%20(not%20open%20source)%20software. [Accessed 5 May 2023].

[12] The PostgreSQL Global Development Group, "License," The PostgreSQL Global Development Group, [Online]. Available: https://www.postgresql.org/about/licence/. [Accessed 5 May 2023].

[13] Timescale, Inc., "Licenses," Timescale, Inc., [Online]. Available: https://www.timescale.com/legal/licenses. [Accessed 5 May 2023].

[14] Docker Inc., "Pricing and Subscriptions," Docker Inc., [Online]. Available: https://www.docker.com/pricing/. [Accessed 5 May 2023].

[15] Google LLC, "LICENSE," Google LLC, [Online]. Available: https://github.com/GoogleWebComponents/google-chart/blob/main/LICENSE. [Accessed 5 May 2023].

[16] Timescale, Inc., "Install TimescaleDB from a pre-built container," Timescale, Inc., [Online]. Available: https://docs.timescale.com/self-hosted/latest/install/installation-docker/. [Accessed 5 May 2023].

[17] M. Corporation, "Entity Framework Core," Microsoft Corporation, 25 05 2021. [Online]. Available: https://learn.microsoft.com/en-us/ef/core/. [Accessed 4 May 2023].

[18] Microsoft Corporation, "DbContext Class," Microsoft Corporation, [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/system.data.entity.dbcontext?view=entity-framework-6.2.0. [Accessed 5 May 2023].

[19] .NET Foundation and Contributors, "LICENSE.txt," 22 09 2022. [Online]. Available: https://github.com/restsharp/RestSharp/blob/dev/LICENSE.txt. [Accessed 5 May 2023].

[20] I. BLIZZARD ENTERTAINMENT, "Getting Started," BLIZZARD ENTERTAINMENT, INC., [Online]. Available: https://develop.battle.net/documentation/guides/getting-started. [Accessed 4 May 2023].

[21] Timescale, Inc., "About hypertables," Timescale, Inc., [Online]. Available: https://docs.timescale.com/use-timescale/latest/hypertables/about-hypertables/. [Accessed 5 May 2023].

[22] J. Bogard, "LICENSE.txt," [Online]. Available: https://github.com/AutoMapper/AutoMapper/blob/master/LICENSE.txt. [Accessed 4 May 2023].

[23] Microsoft Corporation, "Background tasks with hosted services in ASP.NET Core," Microsoft Corporation, 22 March 2023. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/host/hosted-services?view=aspnetcore-7.0&tabs=visual -studio. [Accessed 5 May 2023].

[24] Microsoft Corporation, "Configuration in ASP.NET Core," Microsoft Corporation, 15 01 2023. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/configuration/?view=aspnetcore-7.0. [Accessed 5 May 2023].

[25] Microsoft corporation, "Kestrel web server in ASP.NET Core," Microsoft corporation, [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel?view=aspnetcore-3.1#host-filtering. [Accessed 5 May 2023].

[26] BLIZZARD ENTERTAINMENT, INC., "World of Warcraft Localization," BLIZZARD ENTERTAINMENT, INC., [Online]. Available: https://develop.battle.net/documentation/world-of-warcraft/guides/localization. [Accessed 6 May 2023].

[27] Microsoft Corporation, "DbContext Lifetime, Configuration, and Initialization," Microsoft Corporation, 18 02 2023. [Online]. Available: https://learn.microsoft.com/en-us/ef/core/dbcontext-configuration/. [Accessed 5 May 2023].

[28] M. Corporation, "Use scoped services within a BackgroundService," Microsoft Corporation, 09 03 2023. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/extensions/scoped-service. [Accessed 5 May 2023].

[29] S. Walpole, "Using scoped services inside singletons," 21 09 2020. [Online]. Available: https://samwalpole.com/using-scoped-services-inside-singletons. [Accessed 5 May 2023].

[30] Wowhead, "Tooltips," ZAM Network, LLC, [Online]. Available: https://www.wowhead.com/tooltips. [Accessed 5 May 2023].

[31] G. LLC, "Line Chart," Google LLC, [Online]. Available: https://developers.google.com/chart/interactive/docs/gallery/linechart. [Accessed 6 May 2023].

[32] Microsoft corporation, "Tutorial: Containerize a .NET app," Microsoft corporation, [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/docker/build-container?tabs=windows. [Accessed 5 May 2023].

[33] JetBrains s.r.o., "The .NET Memory Profiler," JetBrains s.r.o., [Online]. Available: https://www.jetbrains.com/dotmemory/. [Accessed 5 May 2023].

[34] Timescale, Inc., "hypertable_size()," Timescale, Inc., [Online]. Available: https://docs.timescale.com/api/latest/hypertable/hypertable_size/.

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

Mina, Eric Rodionov

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Mängusisese oksjoni jälgimisrakendus võrgumängus World of Warcraft", mille juhendaja on Einar Kivisalu
   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.

12.05.2023

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.

Lisa 2 - Rakenduses kasutatava litsentsi näide

```
MIT License

Copyright (c) 2023 Eric Rodionov

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```