TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ahmed Abdullajev 192816IADB

Rakendus hindade jälgimiseks toidupoodides

Bakalaureusetöö

Juhendaja: Aleksei Talisainen
MSc

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: Ahmed Abdullajev

27.11.2022

Annotatsioon

Probleemiks on toidupoodide tugev hinnatõus ja arusaamatus, kust konkreetseid tooteid odavamalt osta saab. Käesolevas bakalaureusetöös loob autor rakenduse, mis ise kogub erinevate toidupoe toodete andmed, sisestab need andmebaasi ja siis neid andmeid kasutades näitab tooteid koos nende hindade ja infoga, see tähendab, et autor loob veebikraabitsast ja veebirakendusest koosneva rakenduse, mis suhtlevad andmebaasiga, mille autor ka loob. Loodav rakendus tuleks käsitleda prototüübina. Veebirakendus on leitav aadressil https://unique-douhua-eca637.netlify.app

Prototüübi etapi jaoks autor loob veebikraabitsa, mis suudab koguda tooteandmeid toidupoodide nagu Prisma, Serlveri ja Maxima (Barbora) veebilehtedelt. Autor loob veebirakenduse, mis võimaldab kasutajatel vaadata tooteid kategooriate kaupa, otsida tooteid ning võrrelda erinevate toidupoodide toodete hindu interneti vahendusel, samuti võimaldab vaadata tooteid erinevatel kuupäevadel, millal veebikraabits andmeid kogus. Autor loob veebirakenduses administraatori paneeli, et administraator saaks lisada autori veebirakendusele kategooriaid, mis sisaldavad tooteid toidupoodide kategooriate veebilehtedelt ja administraatori paneel võimaldab lisada need toidupoodide kategooriate veebilehti, millelt veebikraabits tooteandmeid kogub. Administraator saab ka muid toiminguid teha.

Ka selles bakalaureusetöös analüüsib autor erinevaid tehnoloogiaid ja meetodeid veebirakenduse ja veebikraabitsa loomiseks ning autor analüüsib töö valminud praktilist osa.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 53 leheküljel, 5 peatükki, 46 joonist, 2 tabelit.

Abstract

An Application for Monitoring Prices in Grocery Stores

The problem is the strong price increase in grocery stores and the misunderstanding of where specific products can be bought cheaper. In this bachelor's thesis, the author creates an application that collects the data of various grocery store products, enters them into a database, and then using this data shows the products with their prices and information, this means that the author creates an application consisting of a web scraper and a web application that communicate with the database that the author also creates. The application being created should be treated as a prototype. The online application can be found at https://unique-douhua-eca637.netlify.app

For the prototype stage, the author creates a web scraper that can collect product data from the websites of grocery stores such as Prisma, Selver and Maxima (Barbora). The author creates a web application that allows users to view products by category, search for products and compare prices of products from different grocery stores via the Internet, and also allows users to view products on different dates when the web scraper collected data. The author creates an administrator panel in the web application so that the administrator can add categories to the author's web application that will contain products from the grocery stores category web pages, and the administrator panel allows adding the grocery stores category web pages from which the web scraper collects product data. The administrator can also perform other actions.

Also in this bachelor's thesis, the author analyzes various technologies and methods for creating a web application and a web scraper, and the author analyzes the completed practical part of the work.

The thesis is in Estonian and contains 53 pages of text, 5 chapters, 46 figures, 2 tables.

Lühendite ja mõistete sõnastik

Accept: application/json	Määrab väljundi tüübiks JSON
API	*Application Programming Interface*, rakenduste programmeerimisliides
Benchmarksgame	*Benchmarks Game*, tarkvaraprojekt programmeerimiskeelte kiiruste võrdlemiseks
CDN	*Content Delivery Network*, omavahel ühendatud serverite võrk, mis kiirendab rakenduste veebilehtede laadimise protsessi
CSS	*Cascading Style Sheets*, keel, mis kirjeldab, kuidas HTML-i elemente tuleks kuvada
DNS	*Domain Name System*, arvuti hajutatud süsteem domeenide kohta teabe hankimiseks
DOM	*Document Object Model*, on spetsiaalne puustruktuur, mis võimaldab juhtida HTML-i märgistust JavaScripti koodist
FTP	*File Transfer Protocol*, võrgu failiedastusprotokoll
GIT	Hajutatud versioonihaldussüsteem
HTML	*HyperText Markup Language*, veebilehtede standardne märgistuskeel
HTTP	*HyperText Transfer Protocol*, rakenduskihi protokoll
HTTPS	*HyperText Transfer Protocol Secure*, HTTP-protokolli laiendus, mis toetab krüptimist turvalisuse suurendamiseks
IEEE Spectrum	Elektri- ja elektroonikainseneride instituudi välja antud kuukiri
IP	Internet Protocol, on Interneti-protokoll
JSX	*JavaScript XML*, on manustatav XML-i sarnane süntaks, mis on mõeldud teisendamiseks kehtivaks JavaScriptiks
JSON	*JavaScript Object Notation*, tekstipõhine andmevahetusvorming, mis põhineb JavaScriptil
MVC	*Model-View-Controller*, koodi korraldamise viis, mis hõlmab erinevate ülesannete lahendamise eest vastutavate plokkide eraldamist
middleware	Pakub mehhanismi rakendusse sisenevate HTTP-päringute kontrollimiseks ja filtreerimiseks
Node.js	Avatud lähtekoodiga serverikeskkond

ORM	*Object-Relational Mapping*, see on tehnika, mis võimaldab objektorienteeritud paradigmat kasutades andmebaasis olevaid andmeid pärida ja neid töödelda
PYPL	*PopularitY of Programming Language*, see luuakse analüüsides, kui sageli Google'ist keeleõpetusi otsitakse
REST	*Representational state transfer*, hajutatud süsteemide tarkvaraarhitektuuri stiil, mis loodi juhisena veebiteenuste loomiseks
RESTful API	Rakendusprogrammi liidese (API) arhitektuurne stiil, mis on loodud REST arhitektuurilise stiili juhiseid järgides
st	See tähendab
SPA	*Single-page application*, veebirakenduse juurutamine, mis laadib ainult ühe veebidokumendi
SQL	*Structured Query Language*, standardiseeritud programmeerimiskeel, mida kasutatakse relatsiooniandmebaaside haldamiseks ja neis olevate andmetega erinevate toimingute tegemiseks
TypeScript	Tugevasti tüübitud programmeerimiskeel, mis põhineb JavaScriptil
WebDriver	Protokoll, mis määrab brauseri toimingute viisid
XML	*eXtensible Markup Language*, see on lihtne tekstipõhine vorming struktureeritud teabe esitamiseks

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Probleemiks on toidupoodide tugev hinnatõus ja arusaamatus, kust konkreetseid tooteid odavamalt osta saab. Turumajanduses võivad kaupade ja teenuste hinnad alati muutuda. Mõni hind tõuseb, mõni langeb. Inflatsioon tekib siis, kui kaupade ja teenuste, mitte ainult üksikute kaupade, hinnad tõusevad laialdaselt, see tähendab, et täna saab 1-euro eest osta vähem kui eile. Teisisõnu, inflatsioon vähendab aja jooksul valuuta väärtust. [1]

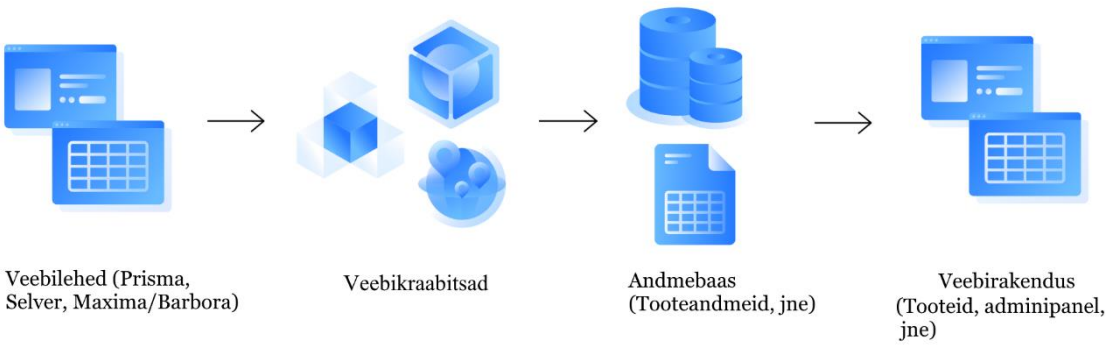
Eestis pole sellist rakendust, mis perioodiliselt ise kogub erinevate toidupoe toodete andmed, sisestab need andmebaasi ja siis neid andmeid kasutades näitab tooteid koos nende hindade ja infoga, et võrrelda erinevate toidupoodide hindu. Seetõttu otsustas autor selle idee ellu viia, kasutades selle teostamiseks erinevaid tehnoloogiaid ja meetodeid. Loodav rakendus tuleks käsitleda prototüübina.

See rakendus koosneb 2 osast, esimene osa on veebikraabitsa loomine, mis kogub tooteandmeid erinevatest toidupoodidest ja lisab need andmebaasi. Hetkel on autor valinud 3 toidupoodide veebilehti, kust kraabib tooteandmeid, need on Maxima (Barbora), Prisma ja Selveri toidupoodide veebilehed, st autor loob iga toidupoe jaoks veebikraabitsa, kasutades meetodeid ja tehnoloogiaid, mis on nendelt toidupoodide kategooriate veebilehtedelt andmete kogumiseks kõige sobivamad. Teine osa on juba veebirakenduse loomine, mille abil näidetakse infot erinevate toidupoodide toodete kohta ja millel on ka administraatoripaneel.

Selle lõputöö peamine eesmärk on luua veebikraabits, mis kraabib toidupoodide kategooriate veebilehti ja salvestab toodete andmed andmebaasi. Ning nende andmete põhjal loob autor veebirakenduse, mis näitab neid tooteid, et võrrelda erinevate toidupoodide hindu, samuti saab kasutaja näha veebikraabitsa varasemate andmete kogumise iteratsioonide tooteandmeid, st toodete erinevaid versioone kuupäevadel, mil veebikraabits kogus tooteandmeid erinevatelt veebilehtedelt. Autori veebirakenduses on kategooriad, mille alla tulevad vastavad tooted erinevatest toidupoodidest, samuti on veebirakenduses vorm toodete otsimiseks nime järgi, mis annab kasutajale tooteid välja. Autor lisab veebirakendusele ka administraatori paneeli, kus veebirakenduse

administraator saab andmebaasi lisada toidupoodide kategooriate veebilehti, st lingid kategooriate veebilehtedele, kust veebikraabits peaks tooteandmeid koguma, samuti saab uuendada ja eemaldada kategooriate veebilehti. Autor lisab autentimise ja autoriseerimise administraatori paneeli turvaliseks kasutamiseks ja sinna sisselogimiseks.

Kategooriaid ja alamkategooriaid saab lisada administraatori paneelis, kus autori veebirakendus sisaldab vastavaid tooteid, mida veebikraabits on kogunud toidupoodide kategooriate veebilehtedelt. Veebilehe lisamisel peab administraator määrama, millisesse kategooriasse toidupoe veebileht kuulub, samuti saab administraator kategooriaid kustutada ja uuendada. Veebirakenduse administraator saab kustutada ka veebikraabitsa versioone, mis eemaldab veebikraabitsa poolt kindlal kuupäeval kogutud tooteandmed. Autor majutab ka veebirakendust, et sellele pääseks juurde interneti kaudu. Rakenduse skeem on toodud joonisel 1.



Joonis 1. Rakenduse skeem [2]

2 Tehnoloogia analüüs

Selles peatükis analüüsib autor erinevaid tehnoloogiaid oma rakenduse loomiseks, valib neist sobivaima ja selgitab, miks just selle tehnoloogia valis. Ka selles peatükis valib autor välja toidupoed, kust veebikraabits kogub tooteandmeid.

2.1 Veebikraabitsa programmeerimiskeelte analüüs

Selles osas analüüsib autor erinevaid veebikraabitsa tehnoloogiaid ning lõpuks räägib, millise programmeerimiskeele oma veebikraabitsa jaoks valis ja milliseid tehnoloogiaid autor selleks kasutas iga toidupoe veebilehe puhul. Seetõttu otsustas autor analüüsida veel 3 programmeerimiskeelt, mis on veebikraapimisel populaarsed, need keeled on Python, JavaScript ja Ruby. Programmeerimiskeele valikul keskendub autor lihtsusele, süntaksi selgusele, koodi lühidusele, populaarsusele, kogukonnale, toele veebifoorumite ja õpetuste kaudu, ja autor keskendub keelele, mis on rohkem keskendunud veebikraabitsate loomisele ning millel on selleks palju tööriistu ja teeke.

2.1.1 Python

Python on kõrgetasemeline, üldotstarbeline ja väga populaarne programmeerimiskeel. Pythoni programmeerimiskeelt (uusim Python 3) kasutatakse veebiarenduses, veebikraapimises, masinõppe rakendustes ja kõigis teistes tarkvaratööstuse tehnoloogiates. Pythoni programmeerimiskeel sobib väga hästi algajatele, aga ka kogenud programmeerijatele, kes kasutavad teisi programmeerimiskeeli nagu C++ ja Java. [3]

Veebikraapimine või kogumine nõuab tõhusaks kasutamiseks head tööriista. See hõlmab andmete roomamist, sisu toomist, otsimist, sõelumist ja andmete ümbervormindamist, et muuta kogutud andmed analüüsiks ja esitamiseks valmis. Töö jaoks veebi kraapimiseks on oluline kasutada õiget tarkvara ja keeli. [4]

Pythonit peetakse veebi kraapimiseks kõige sagedamini kasutatavaks programmeerimiskeeleks. Muide, see on ka IEEE Spectrumi järgi 2022. aasta parim

programmeerimiskeel. Selle objektorienteeritud keelega on kaasas suur hulk teeke, sealhulgas masinõppe mooduleid. [4]

Pythoni veebikraapimise parimaks valikuks teeb selle võime käsitleda peaaegu kõiki andmete ekstraheerimisega seotud protsesse. Peale selle, et Python on lihtne kasutada (eriti semikoolonite ja loogiliste sulgude mittekasutamine), on Python märkimisväärne muutujate otsese kasutamise poolest, kus iganes vaja. See muudab töö oluliselt lihtsamaks ja kiiremaks. Programmeerimiskeel on tuntud ka oma „väike kood, suur ülesanne“ lähenemisviisi poolest, kus koodid on üldiselt väikesed võrreldes teiste programmide omadega. [4]

Samuti on Pythoni süntaksist väga lihtne aru saada. See on nagu ingliskeelsete fraaside ja avalduste lugemine. Algajad arendajad ja isegi need, kes Pythoni abil programmeerimisest midagi ei tea, saavad tõenäoliselt aru või arendajatel on aimu, milleks need koodid on mõeldud. [4]

Samuti aitab see, et Pythonil on tohutu ülemaailmne kasutajate kogukond. Pythoni programmeerimisele on pühendatud palju aruteluplaate ja vestlusgruppe. Kasutajad saavad hõlpsasti leida abi või nõuandeid, kuidas tulla toime raskustega, mis neil veebikorjeprogrammide kirjutamisel kokku puutuvad. [4]

Eelised [5]:

- Suur kogukond ja toe kättesaadavus veebifoorumite ja õpetuste kaudu.
- Vähem kogenud arendaja jaoks on Pythonist teiste keeltega võrreldes lihtsam aru saada.
- Python pakub palju kasulikke teeke, Pythonil on palju teeke erineva keerukusega veebilehtede kraapimiseks.

Probleemid [5]:

- Python pole teadaolevalt üks kiiremaid programmeerimiskeeli. Tõepoolest, vastavalt Benchmarksgame'ile on see palju aeglasem kui Java. Programmi kiirus sõltub aga veebikraapimisel palju koodist ja veebilehtele esitatavatest päringutest,

seega võib veebikraapimise protsess olla sama pikk, olenemata kasutatavast programmeerimiskeelest.

Parimad Pythoni teegid veebikraapimiseks [6]:

- BeautifulSoup
- Scrapy
- Selenium
- Requests
- Urllib3
- Lxml
- MechanicalSoup.

2.1.2 JavaScript

JavaScript on nii kliendi kui ka serveri poolel kasutatav tekstipõhine programmeerimiskeel, mis võimaldab muuta veebilehti interaktiivseks. Kui HTML ja CSS on keeled, mis annavad veebilehtedele struktuuri ja stiili, siis JavaScript annab veebilehtedele interaktiivseid elemente, mis kasutajat kaasavad. [7]

Node.js keskkonna abil kasutatakse seda palju rohkem ka veebirakenduste arendamiseks. Node.js pakub selliseid teeke nagu Puppeteer ja Nightmare, mida tavaliselt kasutatakse veebi kraapimiseks. [5]

JavaScripti kasutatakse peamiselt veebipõhiste rakenduste ja veebibrauserite jaoks. Kuid JavaScripti kasutatakse ka tarkvaras, serverites ja manustatud riistvara juhtelementides väljaspool veebi. Siin on mõned põhilised asjad, mille jaoks JavaScripti kasutatakse [7]:

- Veebilehtedele interaktiivse käitumise lisamine: JavaScript võimaldab kasutajatel veebilehtedega suhelda. Asjadele, mida saate veebilehel JavaScriptiga teha, pole peaaegu mingeid piiranguid

- Veebi- ja mobiilirakenduste loomine: arendajad saavad veebi- ja mobiilirakenduste arendamiseks ja ehitamiseks kasutada erinevaid JavaScripti raamistikke. JavaScripti raamistikud on JavaScripti kooditeekide kogud, mis pakuvad arendajatele rutiinsete programmeerimisfunktsioonide ja -ülesannete jaoks kasutatavat eelkirjutatud koodi, sõna otseses mõttes raamistikku veebilehtede või veebirakenduste loomiseks

- Veebiserverite loomine ja serverirakenduste arendamine: lisaks veebilehtedele ja rakendustele saavad arendajad kasutada JavaScripti ka lihtsate veebiserverite loomiseks ja Node.js-i abil taustainfrastruktuuri arendamiseks

Eelised [5]:

- Suur kogukond ja toe kättesaadavus veebifoorumite ja õpetuste kaudu

- Node.js saab väga tõhusalt käsitleda samaaegseid veebilehe päringuid

Probleemid [5]:

- Ei ole lihtne mõista, eriti vähem kogenud arendajatele

- Pole nii töökindel ja tõhus kui Python protsessorit nõudvate ülesannete jaoks, näiteks suure hulga veebiandmete sõelumine pärast kogumist.

2.1.3 Ruby

Ruby keele autor tahtis luua paindliku objektorienteeritud keele, mida programmeerijatele meeldiks kasutada. Rubyt kasutatakse peamiselt veebirakenduste loomiseks ja see on kasulik muude programmeerimisprojektide jaoks. Seda kasutatakse laialdaselt serverite ehitamiseks ja andmetöötluseks, veebi kraapimiseks ja roomamiseks. [8]

Võrreldes Pythoni ja JavaScriptiga kasutavad programmeerijad Rubyt vähem, kuid sellel on spetsiifilised funktsioonid, mis on kohandatud veebikraapimise kasutamiseks. Ruby Nokogiri teek pakub võimsaid meetodeid HTML-i ja XML-i sõelumiseks, mis on kaks levinumat veebikraapimise väljundi vormingut. [5]

Eelised [5]:

- Ruby süntaks ei ole nii tõlgendatav kui Pythoni oma, kuid sama funktsiooni saab programmeerida vähemate koodiridadega Rubys

- Ruby Bundler muudab GitHubi pakettide haldamise ja juurutamise lihtsamaks, mis säästab aega eriti projektide puhul, mis vajavad olemasolevat paketti

- Nokogiri teek saab katkise HTML-koodiga hakkama lihtsamini kui teised keeled

Probleemid [5]:

- Arvestades, et Ruby ei ole nii eelistatud kui Python ja Javascript, on Ruby jaoks saadaval vähem ressursse ja kogukonna tööriistu

- Pole populaarne programmeerimiskeel

2.1.4 Valitud tehnoloogia veebikraapimiseks

Pärast nende programmeerimiskeelte üksikasjalikku analüüsi otsustas autor nende programmeerimiskeelte populaarsust täpsemalt võrrelda. (Tabel 1).

Tabel 1. Python, JavaScript ja Ruby populaarsuse võrdlus

	Python	JavaScript	Ruby
Koht	1 [9]	3 [9]	15 [9]

PYPL-i programmeerimiskeele populaarsuse indeks luuakse analüüsides, kui sageli Google'is keeleõpetusi otsitakse. Mida rohkem keeleõpetust otsitakse, seda populaarsemaks see keel eeldatakse. See on ka näitaja, sest mida populaarsem on keel, seda suurem on kogukond ja seda lihtsam on foorumitest küsimusele vastust leida. Toorandmed pärinevad Google Trend-sist. [9]

Peamiseks kaalutluseks peaks olema programmeerimiskeele tundmine, kuna veebkraapimist saab toetada peaaegu igas keeles. Teine oluline kaalutlus on ka veebiressursside kättesaadavus vea lahendamiseks või probleemile alternatiivsete kodeerimislahenduste otsimiseks. [5] Autori ülaltpoolt läbi viidud analüüsi abil sai autor aru, et Ruby programmeerimiskeel sobib kõige vähem toidupoodide veebikraapimiseks, kuna Ruby jaoks on Internetis vähe ressursse ja kogukonna tööriistu, samuti on autor pole

kunagi selles keeles koodi kirjutatud, samuti pole see populaarne programmeerimiskeel. Selle tulemusena jääb üle valida JavaScripti ja Pythoni vahel.

Eelpool autori poolt läbiviidud analüüsi põhjal saab väita, et JavaScript sobib paremini veebilehtede loomiseks, täpsemalt veebilehtede esiosa loomiseks. Python seevastu on väga populaarne keel ja seda kasutatakse sageli veebikraapimiseks, mistõttu on väga lihtne leida ressursse ja erinevaid teeke veebilehtede veebikraabitsate kirjutamiseks. Analüüsi abil võib öelda, et python on lihtne keel, arusaadava süntaksiga, populaarne, suure kogukonnaga, suurepärast tuge läbi veebifoorumite ja õpetuste, sobib hästi veebikraapimiseks ja omab suurt hulka teegid erineva keerukusega veebilehtede kraapimiseks. Tänu autori ülaltoodud analüüsile otsustas autor luua Pythoni programmeerimiskeelt kasutades veebikraabitsad erinevate toidupoodide veebilehtede jaoks.

2.2 Toidupoodide valimine tooteandmete kogumiseks

Pärast seda, kui autor valis toidupoodide veebilehtede veebikraapimiseks Pythoni programmeerimiskeele, hakkas autor valima, millistest toidupoodidest autor kraabib. Autor valis välja need toidupoed, millel on veebileht infoga kõigi nende poodidest müüdavate toodete kohta ning autor valis need poed, kust on mugav andmeid koguda, sellised veebilehed olid:

- www.prismamarket.ee, kus on info Prisma kauplustes olevate toodete kohta
- www.selver.ee, kus on info Selveri kauplustes olevate toodete kohta
- www.barbora.ee, kus on infot toodete kohta Maxima kauplustes ja Barbora veebipoes endas

2.3 Pythoni teekide valimine veebilehtedelt andmete kogumiseks

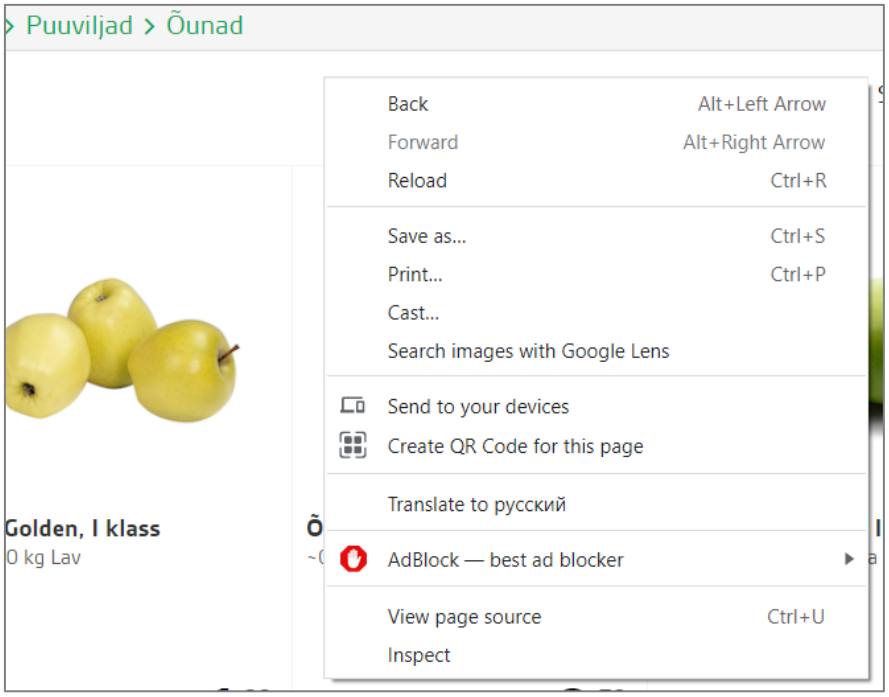
Kuna autor on juba valinud, millises programmeerimiskeeles kirjutab toidupoodide jaoks veebikraabitsaid ja valinud toidupoodide veebilehed, millelt tooteandmeid koguda, hakkas autor analüüsima iga veebilehte, millelt andmeid kogub, ja tegi selle põhjal valiku sobivad meetodid ja Pythoni teeke veebikraapimiseks. Selles jaotises räägib autor, milliseid meetodeid ja Pythoni teeke autor toidupoodide veebikraapimiseks kasutas,

kuidas autor iga toidupoodi kraabis, millised probleemid autoril tekkisid ja kuidas need lahendati.

2.3.1 Prisma

Prisma veebileht oli esimene veebileht, kust autor hakkas oma veebikraabitsat looma. Autor analüüsis esmalt, kust ja kuidas andmed selle veebilehte ette jõuavad. Et teada saada, kust veebileht andmeid saab, pidi autor avama brauseris arendaja tööriistad ja avama seal vaatepaneeli nimega „Võrk“ (Network) ning sealt sai autor analüüsida, kuidas see leht serverist andmeid saab. Võrguvaade võimaldab näha ja analüüsida võrgupäringuid, mis moodustavad iga üksiku lehe laadimise ühe kasutaja seansi jooksul. [10]

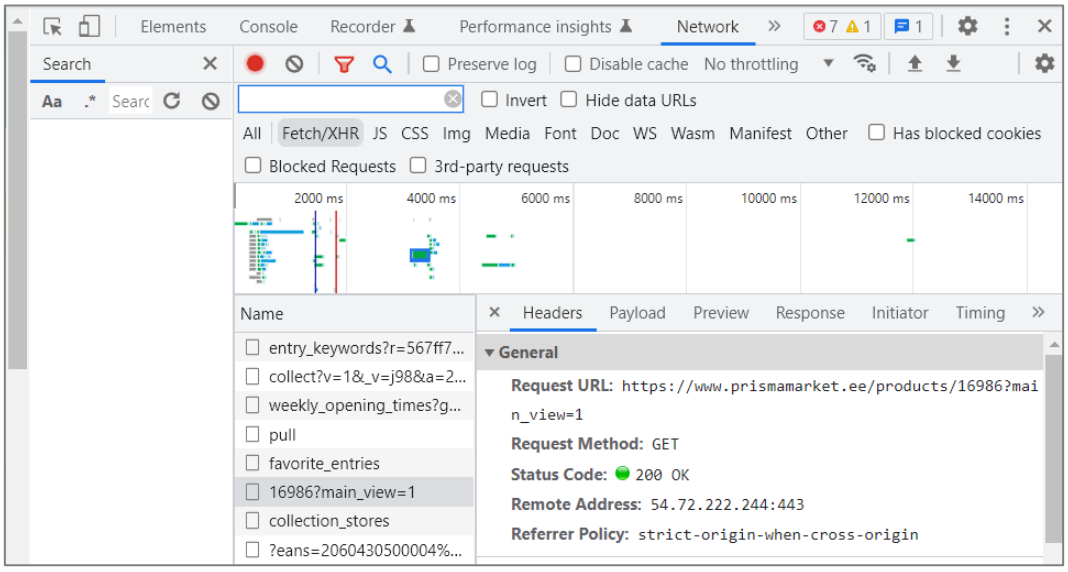
Autor läks lihtsalt Prisma veebilehele, kategooriasse „Õunad“, tegi Google Chrome'i brauseris paremklõpsu, mis andis autorile nimekirja erinevatest tegevustest. Autori soovitud valik oli valik „Üle vaatama“ (Inspect) (joonis 2).



Joonis 2. Prisma õunakategooria veebileht [11]

Mis annab selle lehe kohta palju teavet, näiteks lehe elemendid, lehe stiilid ja palju muud. Autori soovitud vaatepaneel oli „Võrk“ (Network). Võguvaades (joonis 3) autor leidis selle lehe poolt serverisse tehtud päringu lingist

https://www.prismamarket.ee/products/16986, kust sai see leht andmed kõigi kategoorias „Õunad“ kuuluvate toodete kohta.



Joonis 3. Võrguvaade Prisma õunakategooria veebilehel [11]

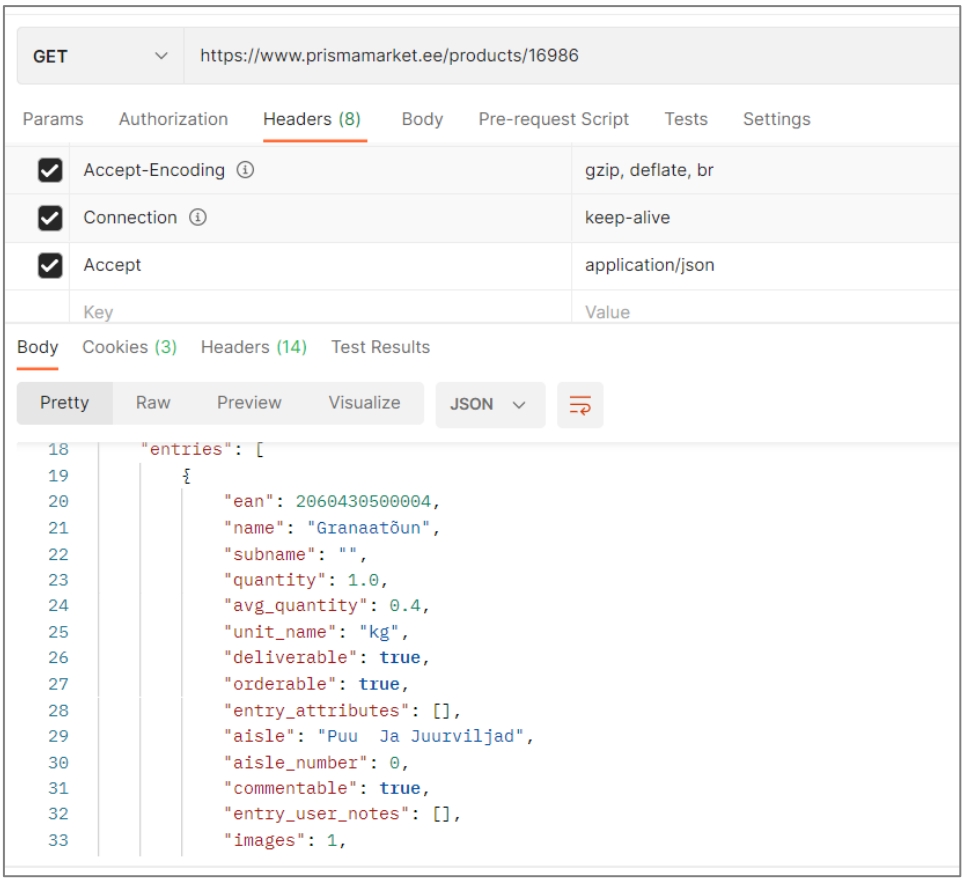
Et näha, kuidas see link (aadress) kliendile andmeid tagastab ja milliseid andmeid tagastab, kasutab autor rakendust Postman.

Postman on rakendus, mida kasutatakse API testimiseks. See on HTTP-klient, mis testib HTTP-päringuid, kasutades graafilist kasutajaliidest, mille kaudu saab erinevat tüüpi vastuseid, mida tuleb hiljem kinnitada. [12]

Postman pakub palju lõpp-punktide interaktsiooni meetodeid. Järgmised on mõned enim kasutatud, sealhulgas nende funktsioonid [12]:

- GET: Teavet saama
- POST: Teavet lisada
- PUT: Teavet uuendama
- DELETE: Teavet kustutama

Autor saatis Postmani abil HTTP GET päringu päisega „Accept: application/json“ aadressile https://www.prismamarket.ee/products/16986 ja sai serverilt vastuseks JSON-tulemuse koos toodete nimekirjaga ja nende andmed, mida autor lihtsalt vajab andmebaasi salvestamiseks (joonis 4).



Joonis 4. JSON-tulemus koos Postmaniga [13]

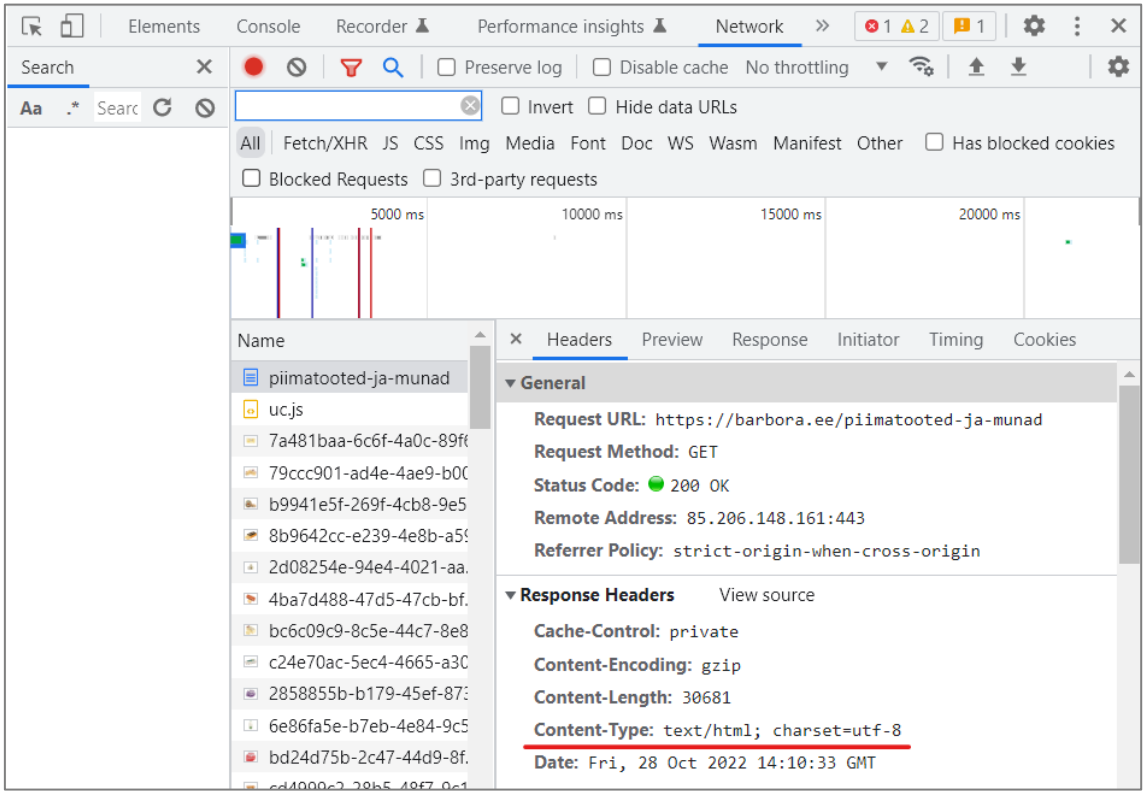
JSON on vorming, mida kasutatakse andmete esitamiseks. See võeti kasutusele 2000. aastate alguses JavaScripti osana ja seda laiendati järk-järgult, et saada kõige tavalisemaks meediumiks tekstipõhiste andmete kirjeldamiseks ja vahetamiseks. JSON-objekt sisaldab andmeid võtme/väärtuse paari kujul. Võtmed on stringid ja väärtused JSON-tüüpi. Võtmed ja väärtused on eraldatud kooloniga. Iga kirje (võtme/väärtuse paar) eraldatakse komaga. [14] JSON-vormingut on väga lihtne sõeluda, selliseid andmeid on lihtne tsüklina sõeluda, lihtsalt saad andmed objektivõtmete kaupa ja sisestad need andmebaasi.

Pärast Prisma veebilehel analüüsimist otsustas autor, et kasutab Pythoni teeki „Requests“, mis saab lihtsalt HTTP-päringuid teha ja andmeid vastu võtta.

Requests teek võimaldab saata HTTP/1.1 päringuid ülimalt lihtsalt. Pole vaja URL-idele päringu stringe käsitsi lisada ega POST-andmeid vormiliselt kodeerida. Elus hoidmine ja HTTP-ühenduse ühendamine on 100% automaatsed. [15]

2.3.2 Maxima

Kogu info Maximas olevate toodete kohta on leitav www.barbora.ee, kus on info ka Barbora veebipoe enda toodete kohta. Autor hakkas analüüsima, kuidas leht andmeid saab. Autor kasutas sama meetodit, nagu Prisma puhul. Autor avas arendustööriistades vaatepaneeli „Võrk“ ja sai päringute põhjal aru, et server tagastab valmis HTML-tulemuse koos tooteandmetega (joonis 5), mistõttu tuli autoril leida Pythoni teek, mis suudab lugeda andmeid HTML-i elementidest.



Joonis 5. Võrguvaade Maxima (Barbora) piimatoodete kategooria veebilehel [16]

HTML on veebilehtede loomise standardne märgistuskeel. See võimaldab luua ja üles ehitada jaotisi ja lõike kasutades HTML-elemente (veebilehe ehitusplokke), nagu sildid ja atribuudid. [17]

HTML-il on palju kasutusjuhtumeid, nimelt :

- Veebiarendus. Arendajad kasutavad HTML-koodi, et kujundada, kuidas brauser kuvab veebilehe elemente, nagu tekst, hüperlingid ja meediumifailid
- Interneti-navigeerimine. Kasutajad saavad hõlpsasti navigeerida ja seotud veebilehtede vahele lisada linki.

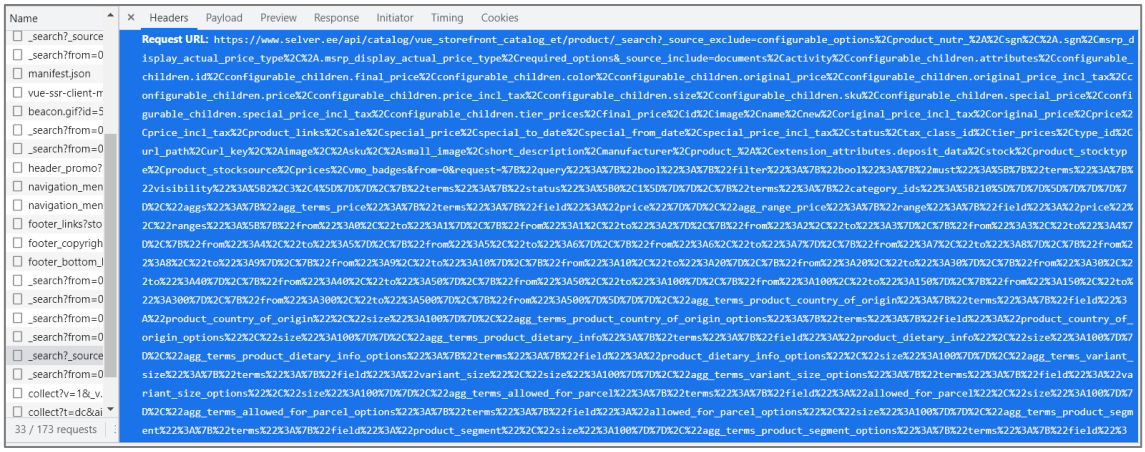
- Veebidokumentatsioon. HTML võimaldab sarnaselt Microsoft Wordiga dokumente korraldada ja vormindada .

Autor leidis väga populaarse teegi, mis suudab koguda andmeid veebilehe HTML-elementidest. Autor otsustas koguda andmeid Pythoni teeki BeautifulSoup abil.

BeautifulSoup on Pythoni teek, mida kasutatakse veebi kraapimiseks, et HTML- ja XML-failidest andmed välja tõmmata. See loob lehe lähtekoodist sõelumispuu, mida saab kasutada andmete hierarhiliseks ja loetavamaks eraldamiseks. [18]

2.3.3 Selver

Kogu info Selveri toodete kohta on leitav www.selver.ee. Samuti asus autor seda veebilehte analüüsima, kuidas andmed veebilehele sisestatakse. Autor kasutas sama meetodit, nagu Prisma puhul. Autor avas arendustööriistades vaatepaneeli „Võrk“ ja sai päringute põhjal aru, et server, nagu ka Prisma veebileht, tagastab tooteteabe JSON-vormingus. Aga aadress, mis andmeid tagastab, on väga pikk (joonis 6) ja ei mahu isegi lehele ära, ka autoril kulus selle aadressi leidmiseks palju aega, sest võrguvaade tagastab palju muid aadresse ning administraatoril, kes need lehed veebi kraapimiseks lisab, ei ole see mugav, administraatoril on lihtsam lihtsalt kopeerida link tootekategooriasse, samuti nagu Prisma ja Maxima (Barbora) jaoks.



Joonis 6. Võrguvaade Selver õunakategooria veebilehel [19]

Kuna Selvera server ei tagasta valmis renderdatud HTML koodi andmetega, nagu see oli Maxima (Barbora) kodulehel, vaid tagastab ainult andmed JSON formaadis ja siis veebilehe esiosa ehk siis JavaScript töötleb neid andmeid ja näitab selle kasutajale, tuli autoril Selveri veebilehtedelt toodeandmete kogumiseks otsida mõni muu meetod.

Autor leidis lõpuks selliste keeruliste veebilehtede jaoks sobiva meetodi. Autor otsustas kasutada Selenium Pythoni teeki, täpsemalt Selenium kasutab veebidraiveri, mis imiteerib brauserit ehk avab brauseri, läheb seejärel veebilehte lehele ja saab juba koguda teavet veebilehe HTML-elementidest. See on umbes sama, kui inimene, kes avab brauseri ja läheb mõnele veebilehele ja näeb kogu teavet.

Selenium viitab paljudele erinevatele avatud lähtekoodiga projektidele, mida kasutatakse brauseri automatiseerimiseks. See toetab kõigi suuremate programmeerimiskeelte, sealhulgas Pythoni sidumist. [20]

Seleniumi kasutab veebibrauserite, näiteks Chrome, Firefox või Safari juhtimiseks WebDriveri protokolli. Selenium saab juhtida nii lokaalselt installitud brauseri eksemplari kui ka seda, mis töötab kaugmasinas võrgu kaudu. [20]

Algselt ja sellest on möödunud umbes 20 aastat oli Selenium mõeldud brauseritevaheliseks, otsast lõpuni testimiseks (vastuvõtutestid). Vahepeal on see aga võetud valdavalt üle üldise brauseri automatiseerimisplatvormina, näiteks ekraanipiltide tegemiseks, mis sisaldab loomulikult ka veebi roomamise ja veebikraapimise eesmärki. [20]

Selenium pakub laia valikut viise veebilehtedega suhtlemiseks, näiteks [20]:

- Nuppude klõpsamine
- Vormide täitmine andmetega
- Lehe kerimine
- Ekraanipiltide tegemine
- Arendaja enda kohandatud JavaScripti koodi käivitamine

Kuid tugevaim argument selle kasuks on võimalus käsitleda veebilehte loomulikul viisil, nagu iga brauser seda teeb. See tuleb eriti esile JavaScripti-rohkete üheleheliste rakendus veebilehtedega. Kui arendaja kraabiks sellise veebilehte traditsioonilise HTTP-kliendi ja HTML-i kraabitsa kombinatsiooniga, oleks arendajal enamasti palju JavaScripti faile, kuid mitte nii palju andmeid, mida kraabida. [20]

2.4 Veebirakenduse arhitektuuri valimine

Pärast veebikraabitsa loomist, mis sisestab tooteandmed andmebaasi autor on otsustanud, et näitab tooteid oma loodavas veebirakenduses. Veebirakendus peab dünaamiliselt kuvama tooteid andmebaasist, veebirakendusel peab olema ka administraatoripaneel, autoriseerimine, autentimine ja erinevad funktsionaalsused. Autoril oli valida dünaamiliste veebilehtede või SPA ja REST vahel. Autoril oli nende veebirakenduse loomise arhitektuuridega juba kogemusi, mistõttu otsustas autor neid võrrelda ja valida endale sobivaima. Arhitektuuri valikul lähtub autor veebirakenduse arendamise paindlikkusest ja mugavusest ning vastutuse jagamisest veebirakenduse tagumise ja esiosa vahel.

2.4.1 Dünaamilised veebilehed

Dünaamilised veebilehed (inglese keeles *Dynamic Web Pages*), selle lähenemisviisi korral koosneb süsteemiklient veebileht, mida kasutaja veebibrauseri kaudu vaatab ja millega suhtleb, HTML-, Javascript- ja CSS-failidest, mis on osaliselt või täielikult serveri poolt renderdatud ja reaalajas veebibrauserisse saadetud. Selle lähenemisviisi õilsad raamistikud hõlmavad [21]:

- ASP.Net
- Spring Boot
- Laravel
- Ruby on Rails

Dünaamiliste veebilehtede puhul on veebibrauser sunnitud laadima täiesti uue HTML-dokumendi (mille server genereeris reaalajas) iga kord, kui kasutaja uuele vaatele navigeerib. Ühelehelised rakendused (SPA), nagu nimigi ütleb, asuvad ainult ühes staatilises, algselt laaditud HTML-dokumendis ja kuvavad kogu selle sisu, muutes seda dokumenti reaalajas. Selle tulemuseks on palju sujuvam, tundlikum, manustatud rakenduse laadne kasutuskogemus ning see suurendab ka jõudlust, kuna brauseril on allalaadimiseks vähem faile ja nende genereerimisega ei ole üldse seotud lisakulusid. Pannakse tähele ka seda, et kui klient on staatiliste tavaliste HTML, CSS ja JavaScript-failide kujul, teeb juurutamise ülilihtsaks. [21]

Kui server ja klient on tihedalt seotud (nagu dünaamiliste veebilehtede puhul), tähendab see rohkem koodi ühes kohas. Ja rohkem koodi tähendab alati rohkem segadust ja suuremat spagetikoodi juhtumise ohtu. Peale selle, kuna serveri ja kliendi programmeerimine nõuab täiesti erinevat lähenemist, võib arendaja lõpuks moodustada mitu arendajat (või isegi meeskonda), kellel on mitu erinevat tehnoloogiatausta ja programmeerimisstiili, mis kõik töötavad ühel koodibaasil. Ilmselgelt tähendab see probleeme. [21]

2.4.2 SPA ja REST

SPA ja REST lähenemisega klient (veebileht, mida kasuta vaatab ja millega veebibrauseri kaudu suhtleb) koosneb staatilistest HTML, CSS ja Javascript failide komplektist (ühe HTML-peafailiga), mis suhtleb veebiserveriga kasutades REST-arhitektuur (tavaliselt HTTP-protokolli kaudu). Kuigi veebiserverit saab rakendada peaaegu igas programmeerimiskeeles, kasutavad arendajad tavaliselt spetsiaalseid raamistikke, et rakendada SPA kliendi poolel ja RESTful API-d serveri poolel. RESTful API raamistike näited on [21]:

- ASP.Net Web API
- Spring Boot
- Laravel

SPA raamistike näited on [21]:

- Vue.js
- React.js
- Angular

RESTful API on rakendusprogrammi liidese (API) arhitektuurne stiil, mis kasutab andmetele juurdepääsuks ja nende kasutamiseks HTTP-päringuid. Neid andmeid saab kasutada andmetüüpide GET, PUT, POST ja DELETE saamiseks, mis viitab ressurssidega seotud toimingute lugemisele, värskendamisele, loomisele ja kustutamisele. RESTful API põhineb esinduslikul seisundiülekandel (REST), mis on arhitektuuriline stiil ja lähenemine suhtlusele, mida sageli kasutatakse veebiteenuste

arendamisel. REST-tehnoloogiat eelistatakse üldiselt teistele sarnastele tehnoloogiatele. See kipub nii olema, kuna REST kasutab vähem ribalaiust, mis muudab selle tõhusaks interneti-kasutuseks sobivamaks. RESTful API-sid saab ehitada ka programmeerimiskeeltega, nagu JavaScript või Python. [22]

Enamik suuremahulisi rakendusi on tänapäeval ehitatud ühelehelise rakendusena ehk SPA-na. SPA arhitektuur seisneb selles, et modellid elavad vaadetest eraldi. See on peaaegu nagu kaks erinevat kontrollerit, mis suhtlevad üksteisega: üks, mis käsitleb klientide päringuid ja teine andmeid. See on suhtlus kliendi ja RESTful API vahel. [23]

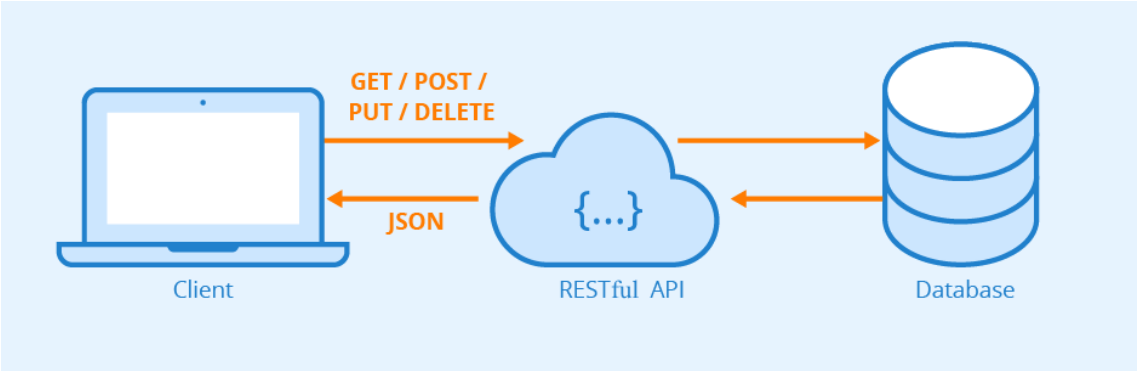
SPA ja REST tehnoloogiapinu puhul on arendaja klient ja server eraldi üksused, mis ei ole seotud millegi muuga kui RESTful API dokumentatsiooniga. See võimaldab arendajatel neid iseseisvalt arendada, testida ja juurutada, mis on eelis, mida dünaamilise veebilehtede tehnoloogiavirna kasutajad ei naudi. Näiteks vana, aegunud kujunduse või halva kasutuskogemusega kliendi saab serveri koodi puudutamata asendada uue läikivaga. Teisest küljest saab serveri poolel rakendada uusi RESTful API lõpp-punkte, mis on seotud kolmanda osapoole süsteemikommunikatsiooniga, ilma et peaks klienti uuesti testima. [21]

SPA ja REST teevad selge jaotuse serveri maailmade (st tehingud, olemid, äridomeenid, taustaülesanded, järjepidevus) ja kliendi (st vormid, diagrammid, stiilid, reageerimisvõime, selgus) vahel. See omakorda muudab arendaja lahenduse puhtaks. [21]

2.4.3 Veebirakenduse valitud arhitektuur

Pärast autori ülaltoodud analüüsi otsustas autor valida SPA ja REST arhitektuuri, kuna see on väga paindlik, siis saab autoril olla eraldi andmebaasiga suhtleva veebirakenduse tagumine osa, kuhu saab saata taotlusi andmete lisamiseks, kustutamiseks ja värskendamiseks, mille jaoks saab luua suvalise kliendi (veebirakenduse esiosa), st võib veebirakenduse tagaküljel olla palju erinevaid kliente erinevatel tehnoloogiatel, mis on väga mugav, kui näiteks veebirakenduse kasutajad ei meeldi veebirakenduse esikülg, nad saavad lihtsalt kasutada RESTful API ja saada andmeid toodete kohta ning luua selle RESTful API jaoks mis tahes kliendi, see tähendab, et veebirakendus muutub väga paindlikuks ja vastutus on jagatud veebirakenduse tagumise osa ja veebirakenduse esiosa vahel. Samuti, kui tulevikus on sellel projektil rohkem kui üks arendaja, siis saab hõlpsasti töö ära jagada, keegi teeb veebirakenduse esiosa ja keegi veebirakenduse tagumise osa.

See võimaldab arendajatel neid iseseisvalt arendada, testida ja juurutada. Veebirakenduse skeem on toodud joonisel 7.



Joonis 7. Veebirakendusi arhitektuuri skeem [24]

2.5 Veebirakenduse tagakülje raamistiku valimine

Kuna autor otsustas, et loob veebirakenduse tagaküljele RESTful API, pidi autor valima raamistiku, millel seda teeb. Autoril oli kogemus selliste raamistikega nagu Spring Boot ja Laravel, mistõttu otsustas autor valida raamistiku sellest raamistike loendist. Selles osas analüüsib autor neid raamistikke ja lõpuks valib endale sobivaima. Raamistiku valikul lähtub autor raamistiku lihtsusest, veebirakenduse arendamise kiirusest.

2.5.1 Spring Boot

Spring kasutatakse laialdaselt skaleeritavate rakenduste loomiseks. Veebirakenduste jaoks Spring pakub Spring MVC, mis on Springi laialdaselt kasutatav moodul, mida kasutatakse skaleeritavate veebirakenduste loomiseks. Kuid Springi projektide peamine puudus on see, et konfigureerimine on tõesti aeganõudev ja võib uutele arendajatele pisut üle jõu käia. Rakenduse tootmisvalmidus võtab veidi aega, kui arendaja on Springi uus kasutaja. [25]

Lahendus sellele on Spring Boot. Spring Boot on avatud lähtekoodiga Java-põhine raamistik. Spring Boot on ehitatud vedru ülaosale ja sisaldab kõiki Springi funktsioone. Ja on tänapäeval muutumas arendajate lemmikuks, kuna see on kiire tootmiseks valmis keskkond, mis võimaldab arendajatel konfigureerimise ja seadistamise kallal vaeva nägemise asemel keskenduda otse loogikale. [25]

Spring Boot eelised [26]:

- See võimaldab vältida Springis esinevat XML-i rasket konfigureerimist.
- See pakub lihtsat hooldust ja REST-i lõpp-punktide loomist.

Spring Boot puudused [26]:

- Üldiselt on Spring Booti arendusfaili suurus selle kujunduse tõttu suurem. Arendajad ei saa faili suurust kontrollida. Spring Booti kujundus lisab tarbetuid sõltuvusi, mis on enamasti kasutamata. Kõik need liigsed sõltuvused suurendavad rakenduste binaarset suurust.
- Muudatused ei ole Spring Booti puhul kerge ülesanne. Kui arendajal pole Spring-süsteemide ja ajaloo kohta tugevaid teadmisi, ei saa arendaja Spring Booti muuta ega tõrkeotsingut teha.

2.5.2 Laravel

Laravel on väljendusrikka ja elegantse süntaksiga PHP veebirakenduste raamistik. Veebiraamistik pakub arendajarakenduse loomise struktuuri ja lähtepunkti. Laravel püüab pakkuda hämmastavat arendajakogemust, pakkudes samal ajal võimsaid funktsioone, nagu põhjalik sõltuvuse süstimine, väljendusrikas andmebaasi abstraktsioonikiht, järjekorrad ja ajastatud tööd, üksuste ja integratsiooni testimine ning palju muud. Laravel võib toimida ka API taustaprogrammina JavaScripti ühelehelisele rakendusele või mobiilirakendusele. [27]

Laraveli eelised [28]:

- Lihtne kodeerimine: Laravel muudab ülesande palju lihtsamaks. Selle põhjuseks on asjaolu, et tarkvaral on ulatuslik eelprogrammeeritud funktsioonide raamatukogu, mis vähendab vajaliku kodeerimise hulka. Selle tulemusena on tarkvara abil lihtsam luua tugevaid veebirakendusi.
- Skaleeritavus: projekti suurused on erinevad. Väga skaleeritav tarkvara võimaldab arendajal sõltuvalt vajadustest lahendada mis tahes suurusega projekte. Laravel on üks selline skaleeritav raamistik, mis muudab selle kasutamise väikeste ja keskmise suurusega veebirakenduste jaoks lihtsaks.

- Lihtne andmete migreerimine: Laraveli kasutamisel on andmete migreerimine palju lihtsam. Sellisena vähendate oma projekti lõpuleviimiseks kuluvat üldist aega.

- Lihtne õppida: paljud veebiarendajad nõustuvad, et Laravel on üks ligipääsetavamaid veebiraamistikke. Seda tänu põhjalikule kasutajadokumentatsioonile, mis on saadaval kõige lihtsamal kujul.

Arvestades Laraveliga kaasnevaid hämmastavaid eeliseid, tundub peaaegu võimatu tarkvara kahjustada. Kuid nagu varem mainitud, ei ole ükski veebiraamistik, hoolimata sellest, kui suurepärane see välja näeb, immuunne mõne puuduse suhtes. [28]

Laraveli puudused [28]:

- Sagedased värskendused: Laravelil on regulaarsed värskendused, mis on suurepärane. Kuid selle negatiivne külg on see, et toote vanemad versioonid muutuvad kiiresti lollakaks. Samuti võivad veebiarendajad mõnikord tarkvara uute versioonide värskendamisel raskusi ette valmistada.

2.5.3 Valitud raamistik veebirakenduse tagaosa jaoks

Pärast nende raamistike üksikasjalikku analüüsi otsustas autor, et Laraveli raamistik oleks talle kõige sobivam. Spring Boot raamistik sobib ka RESTful API rakenduse ehitamiseks, kuid RESTful API arendamiseks kulub rohkem aega ja autoril pole Spring-süsteemide ja ajaloo kohta tugevaid teadmisi.

PHP on tõlgendatud programmeerimiskeel, st pole vaja oodata, kuni rakendus on kompileeritud, et kontrollida selle toimimist, mis muudab rakenduste arendamise palju kiiremaks. PHP on dünaamiliselt tüübitud keel, see on keel, mis suudab hõlpsasti luua erinevat tüüpi muutujaid, see viitab neile programmeerimisskriptidele, mis ei nõua muutuja tüübi määratlemist, seega muutub väikese rakenduse arendamine PHP-ga palju kiiremaks kui Java-ga. Kuna veebirakenduse tagumine ots on väike ja selle kallal töötab ainult üks autor, on PHP selle prototüübi arendamiseks parim valik.

Samuti sobib see raamistik väikeste rakenduste jaoks hästi, rakenduse tagaosa pole suur ja sellel pole palju funktsionaalsust, seega sobib see raamistik väga hästi selle rakenduse

jaoks, mida autor areneb. Lisaks on PHP veebirakendused veebis hõlpsasti avaldatud ja need veebimajutajad on üsna odavad ning tasuta veebimajutajaid on palju.

2.6 Veebirakenduse esiotsa tehnoloogia valimine

Pärast veebirakenduse tagumise osa valimist asus autor analüüsima ja valima veebirakenduse esiotsa jaoks tehnoloogiat, millega kasutaja hakkab kasutama autori veebirakendust ja mis saadab päringuid veebirakenduse tagaküljele (RESTful API-le). Peamiselt on autoril kogemusi kahe väga populaarse tehnoloogiatega, mida peamiselt kasutatakse projektides, nendeks on React.js ja Vue.js. Tehnoloogiate valikul lähtub autor tehnoloogia lihtsusest, kolmanda osapoole ja valmistööriistade kombinatsioonide tasakaalust.

2.6.1 React.js ja Vue.js võrdlus

Tänapäeval on iga ettevõtte eesmärk luua lühikese aja jooksul kvaliteetne veebilahendus. Selle elluviimiseks valib arendajate kogukond paljude JavaScripti teekide ja raamistike ning arutelude hulgast, et leida parim. Vue.js ja React.js on esiotsa arendamiseks kõige kiiremini arenevad tööriistad mitmel põhjusel. Kuigi need tehnoloogiad võivad aidata luua samu tooteid ja rakendusi, on neil mõlemal plusse ja miinuseid. Mõlemal veebiarendustööriistal on küpsed kogukonnad, lai tugi ja populaarsus, kuid Vue on raamistik ja React on teek. [29]

Üks võtmetegureid Reacti ja Vue võrdlemisel oli see, et Vue.js raamistiku looja Evan You kasutas React.js-i uue raamistiku arendamiseks inspiratsiooniallikana. Need veebiarendustööriistad on üksteisega eriti sarnased. Seda saab näha isegi ametlikus Vue.js dokumentatsioonis, kus sellised sarnasused on näidatud. Peamised esialgsed ühised punktid on [29]:

- Virtuaalne DOM
- Reaktiiv- ja komponentstruktuur
- JavaScripti kasutamine
- TypeScripti tugi

- Sujuv versiooni migratsioon
- Tagasiühilduvus
- Suur valik raamatukogusid ja tööriistu
- Paindlikkus, jõudlus ja kiirus
- Suured ja aktiivsed kogukonnad

Kui rääkida Reacti ja Vue peamistest erinevustest, peaks autor mainima terminite erinevust. Vue on raamistik, React aga teek. Raamistik on kooditeekide kogum, mis pakuvad veebiarendajale igapäevaste programmeerimistoimingute jaoks eelnevalt kirjutatud koodi. Raamistikel on konkreetne kontekst ja abi veebirakenduste loomisel selles kontekstis. Raamistikud suunavad arendajat ka arhitektuuri ja sellele järgneva projekti osas õiges suunas. Raamistik võib vabastada arendaja peavalust, mida kasutada ja mida mitte kasutada, kuidas arendaja rakendust tellida ja kuidas seda kujundada. See on väga kasulik, kui on vaja kliendi jaoks midagi kiiresti välja töötada. Teekide kasutamine seevastu võimaldab arendajal kujundada oma rakenduse, mis on kohandatud spetsiaalselt arendaja vajadustele, kuid projekti struktuuri ja arhitektuuri väljamõtlemine, sõltuvuste säilitamine, üksikute teekide värskendamine ja mõne teise tõttu purunemise tuvastamine võib olla äärmiselt aeganõudev. [30], [31]

Teine erinevus tuleneb lähenemisest sisu renderdamisele DOM-ile. Kui React kasutab erandkorras JSX-i, siis Vue kasutab peale JSX-i ka HTML-malle. Lõpuks erinevad React ja Vue eelehitatud ja kolmanda osapoole tööriistade poolest. Reactiga kaasneb pädev arhitektuur, DOM-i manipuleerimine ja komponentide olekuhaldus. Kõik muud funktsioonid on loodud ja toetatud kogukonna liikmete poolt. Selle lähenemisviisi tulemusena saavad arendajad rohkem vabadust. Samal ajal võivad algajad kolmandate osapoolte instrumentide rohkus väljakutseid tekitada. [29]

JSX tähistab Javascript XML-i ja see on Reacti arendajatele väga kasulik tööriist. JSX on JavaScripti keele laiendus, mis võimaldab struktureerida komponentide renderdamist, kasutades HTML-ile sarnast süntaksit. JSX annab meile võimaluse kirjutada HTML-i elemente Javascriptis ja paigutada need DOM-i, teisendades HTML-sildid React-elementideks, ilma et oleks vaja kasutada muid meetodeid, nagu createElement() või

appendChild(). See Javascripti ja HTML-i kombinatsioon toob kaasa võimsamad rakendused, millel on suurem jõudlus. [32]

Kui rääkida Vue-st, siis selle laialt levinud tööriistad ja teeke on välja töötanud selle põhimeeskond. Lisaks neile on olemas ka kogukonnapõhised lahendused. Kokkuvõtteks võib öelda, et Vue-l on tasakaalustatum kombinatsioon kolmandate osapoolte ja eelehitatud tööriistadest, mis rahuldab nii kogenud arendajate kui ka algajate vajadusi. [29]

2.6.2 Valitud tehnoloogia veebirakenduse esiotsa jaoks

Pärast nende tehnoloogiate võrdlemist ja analüüsimist otsustas autor, et Vue raamistik oleks autorile sobivam. Pole need tehnoloogiad väga erinevad, mõlemad on väga populaarsed ja mõlemal on palju sarnasusi. React on teek, mitte raamistik nagu Vue, mille tõttu võib töötava prototüübi loomine võtta palju aega. React nõuab tugevaid JavaScripti oskusi, samas kui Vue.js on rohkem suunatud algajatele arendajatele ja Vue-s on tasakaalustatum segu kolmanda osapoole ja valmistööristadest.

Samuti otsustas autor, et kirjutab Vue.js raamistiku abil koodi Typescript programmeerimiskeeles. Lühidalt öeldes on TypeScript sama JavaScript, see on lihtsalt objektorienteeritud ja tüübitud. TypeScript on JavaScripti süntaktiline superkomplekt, mis lisab staatilise tippimise. Põhimõtteliselt tähendab see, et TypeScript lisab JavaScripti peale süntaksi, võimaldades arendajatel tüüpe lisada. TypeScripti süntaktiline superkomplekt tähendab, et see jagab sama põhisüntaksit kui JavaScript, kuid lisab sellele midagi. [33]

JavaScript on dünaamiliselt tüübitud keel. Võib olla raske mõista, mis tüüpi andmeid JavaScriptis edastatakse. JavaScriptis ei ole funktsiooni parameetritel ja muutujatel teavet. Seega peavad arendajad vaatama dokumentatsiooni või oletama juurutamise põhjal. TypeScript võimaldab määrata koodi sees edastatavate andmete tüübid ja anda teada vigadest, kui tüübid ei ühti. Näiteks TypeScript teatab veast stringi edastamisel funktsiooni, mis eeldab numbrit. JavaScript ei tee seda. TypeScript kasutab kompileerimisaja tüübi kontrollimist. Mis tähendab, et see kontrollib, kas määratud tüübid sobivad enne koodi käivitamist, mitte koodi käivitamise ajal. [33]

Autor valis veebirakenduse esiküljeks staatiliselt tüübitud kompileerimiskeele, kuna kompileerimine toimub kohe pärast koodirea lisamist ning kompileerimine on üsna kiire ega mõjuta kuidagi arenduskiirust, võrreldes veebirakenduse tagakülje raamistiku valikuga, kus autor võrdles PHP-d ja Java-d.

2.7 Andmebaasi haldussüsteemi valimine

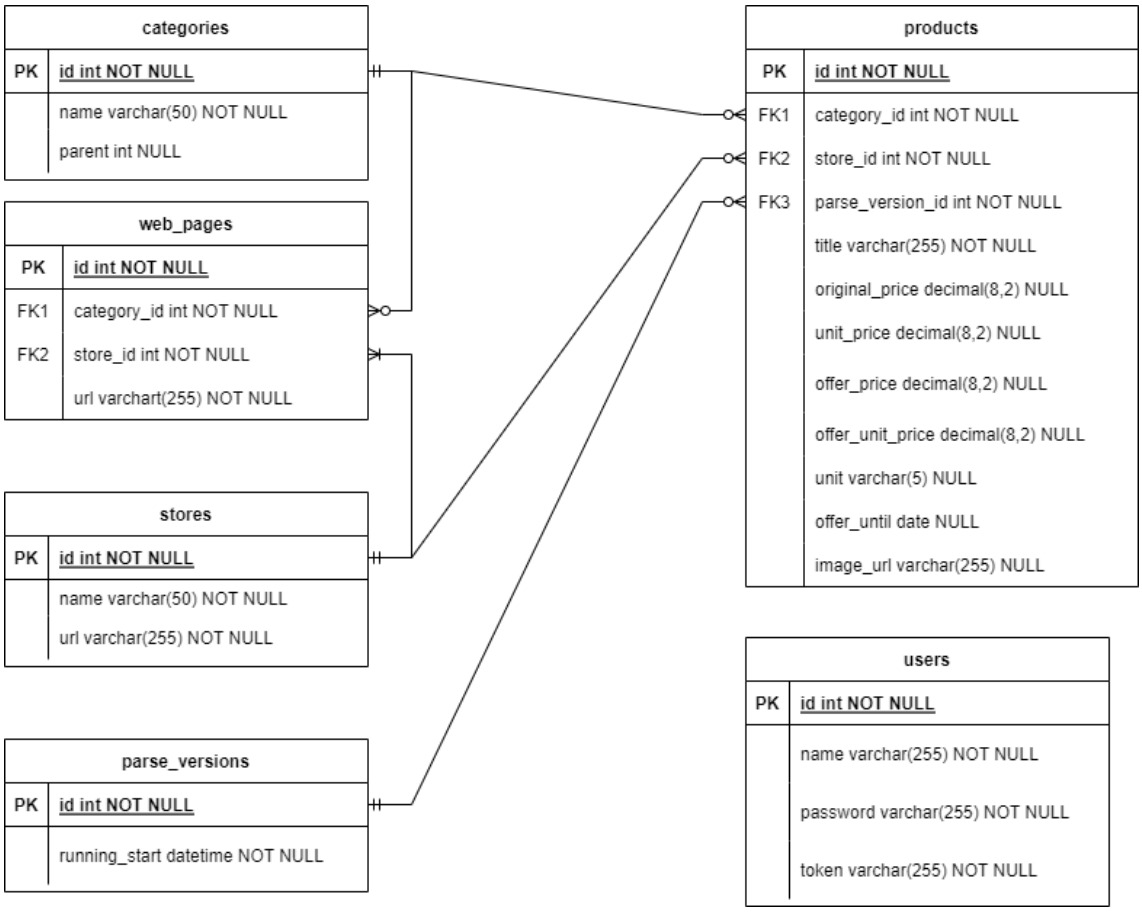
Autor valis oma rakenduse andmebaasihaldussüsteemiks MySQL-i, kuna see süsteem on PHP programmeerimiskeele kasutamisel kõige levinum. [34] Samuti toetab autori rakenduste andmebaasi majutav veebimajutus ainult MySQL-i andmebaasihaldussüsteemi. Autori loodav rakendus tuleks käsitleda kui prototüüpi, seetõttu tuleks tehnoloogiate valikul tehtavatesse otsustesse suhtuda osalise skepsisega. Töö käigus hangitakse lisaandmeid ja saab üle vaadata tehnoloogiate valiku. Hetkel on kõige levinumaks valitud MySQL, kui esineb jõudlusprobleeme ja hooldusraskusi, siis muudab autor andmebaasi haldussüsteemi.

3 Valminud praktilise osa analüüs

Selles osas räägib autor juba valmis tulemusest, rakendusest, mis koosneb veebikraabitsast ja veebirakendusest.

3.1 Andmebaasi analüüs

Andmebaasi skeemi järgi saab hõlpsasti aru, kuidas töötab kogu rakendus, mis koosneb veebikraabitsast ja veebirakendusest (joonis 8).



Joonis 8. Rakenduste andmebaasi skeem

Peamine tabel on siin „products“ (eesti keeles „tooted“), mis sisaldab tooteandmeid, veebikraabitsa abil kogutakse tooteandmed ja sisestatakse need tabelisse „products“, tabelis „products“ on sellised andmed nagu: toote nimetus, toote tavahind, ühikuhind (€/kg, €/l, jne), pakkumise hind, toote ühiku pakkumise hind, ühiku pakkumise periood,

ühik (kg, l, jne) ja link toote fotole, kui need andmed on veebikraabitsa andmete kogumise ajal olemas. Järgmises peatükis on erinevad fotod ja on selgem, kuidas kõik välja näeb.

Tabel „parse_versions“ (eesti keeles „veebikraabitsate versioonid“) on tabel, mis sisaldab kuupäeva, millal autor veebikraabitsa käivitas, iga kord, kui veebikraabits käivitatakse, luuakse uus versioon, iga versioon sisaldab tooteandmeid veebikraabitsa käivitamise ajal. Kuna toodete hinnad muutuvad pidevalt, saavad kasutajad vahetada versioonide vahel, st kuupäevad, millal veebikraabits toodete kohta andmeid kogus, nii et kasutajad saavad jälgida, kuidas muutuvad konkreetsete neid huvitavate toodete hinnad ja samuti saavad kasutajad näha, kui palju konkreetne toode konkreetsel kuupäeval maksab.

Tabel „stores“ (eesti keeles „poed“) on tabel, kus nime ja lingi toidupoe veebilehtele, et oleks selge, millistest toidupoodidest tooteandmeid kogutakse.

Tabel „categories“ (eesti keeles „kategooriad“) on tabel, mis sisaldab autori veebirakenduses saadaolevaid kategooriaid. Veebirakenduses nendesse kategooriatesse minnes saab näha veebikraabitsa poolt kogutud vastavaid tooteid erinevatest toidupoodidest.

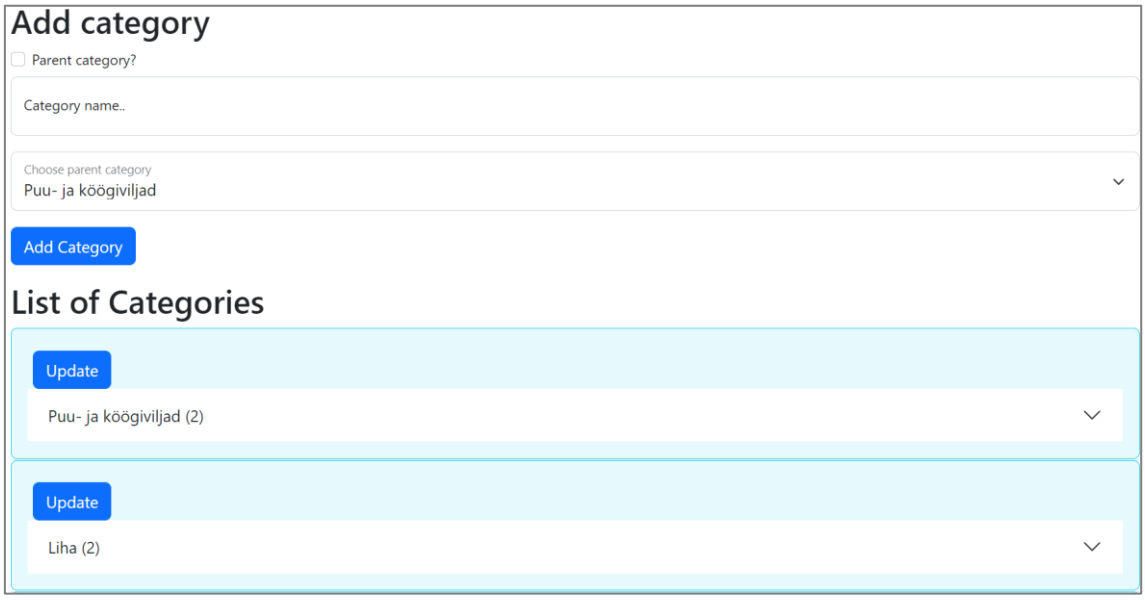
Tabelis „web_pages“ (eesti keeles „veebilehed“) on salvestatud veebilehed, lingid erinevate toidupoodide toodetega kategooriatele, millest veebikraabits peab koguma andmeid ja sisestama need andmebaasi tabelisse „tooted“, veebirakenduse administraator peab lisama neid lehti ja seejärel kogub veebikraabits nendelt veebilehtedelt andmeid.

Tabelis „users“ (eesti keeles „kasutajad“) on salvestatud ainult üks veebirakenduse administraator, kes saab autori veebirakenduse abil andmebaasi kategooriaid ja veebilehti lisada, samuti saab veebist andmeid uuendada ja kustutada ainult selle veebirakenduse administraator, mistõttu otsustas autor lisada veebirakendusele autentimise ja autoriseerimise, et tavakasutajad ei saaks andmebaasis vajalikke andmeid kuidagi kustutada, uuendada ja lisada.

3.2 Kogu rakenduse üksikasjalik kirjeldus

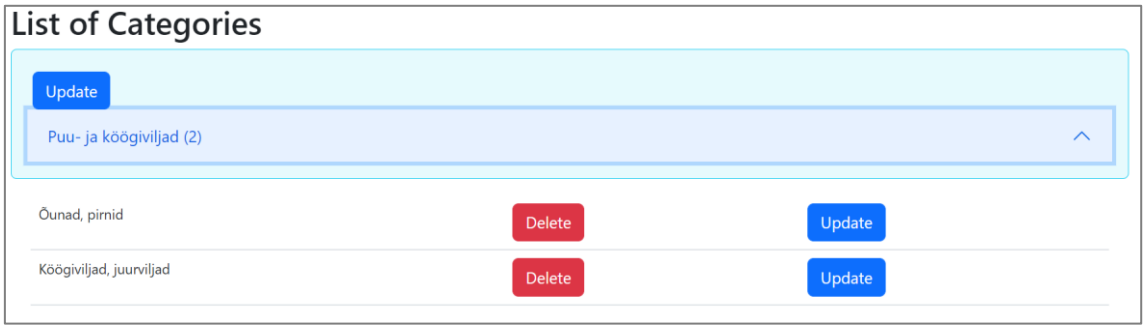
Veebirakendusel on administraatori paneel, kuhu veebirakenduse administraator saab sisse logida. Administraatori paneeli minnes saab administraator lisada ülemkategooriaid ja alamkategooriaid (joonis 9), samuti saab administraator kategooriaid uuendada ja

kustutada, ülemkategooriaid saab kustutada ainult siis, kui neil pole alamkategooriaid. Autori veebirakenduses on kategooriad, mille alla on toodud tooted, mille veebrikraabits on kogunud toidupoodide Prisma, Selveri ja Maxima (Barbora) veebilehtedelt.



Joonis 9. Kategooriate haldamise leht veebirakenduse administraatoripaneelil

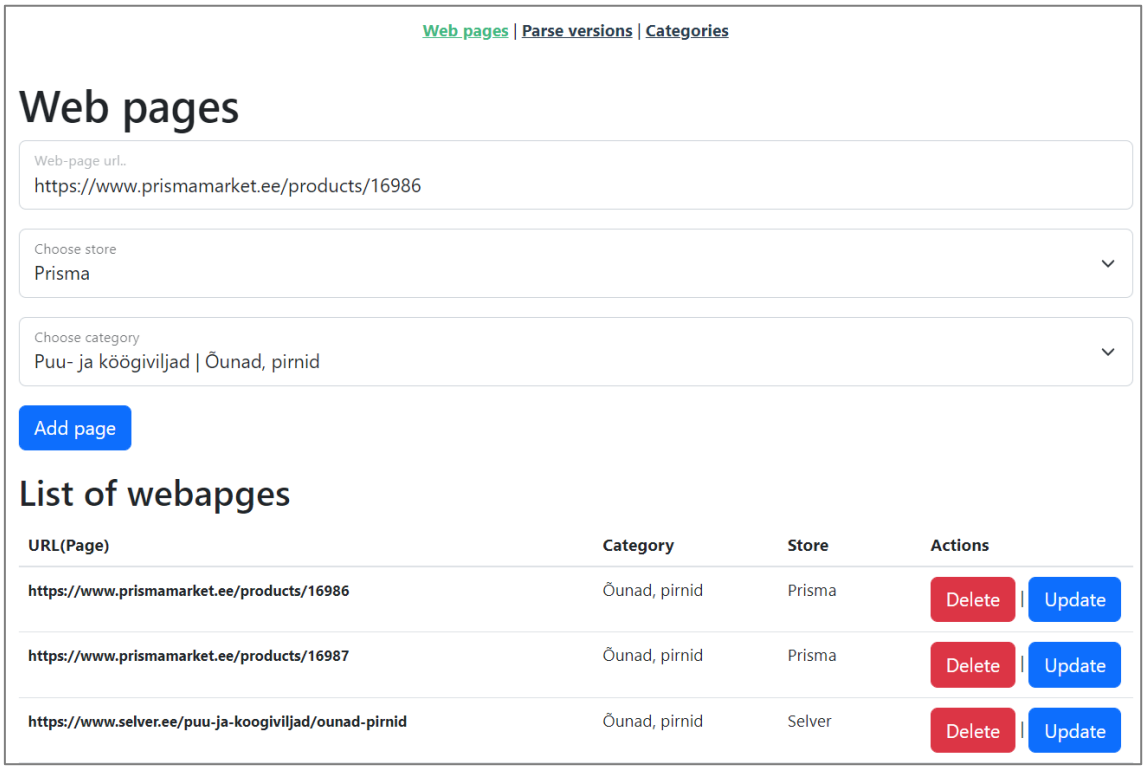
Kategooriate loendis on ülemkategooriate all alamkategooriad (joonis 10).



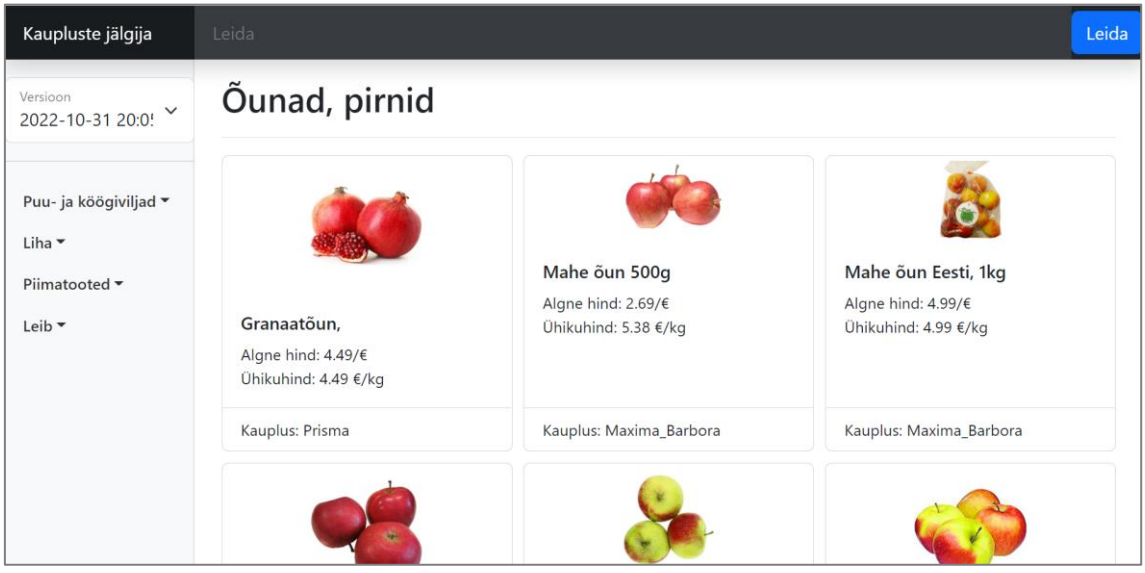
Joonis 10. Konkreetse ülemakategooria all olevate alamkategooriate loend

Samuti saab ja peab administraator lisama veebilehti, linke erinevate toidupoodide kategooriate veebilehtedele, seega saab veebikraabits teada, millistelt konkreetsetelt veebilehtedelt (kategooria veebilehtedelt) on vaja tooteandmeid koguda (joonis 11). Samuti peab administraator valima õige kaupluse ja kategooria, kuhu see veebileht sobib, nii et pärast veebikraapimist hakkavad autori veebirakendusel olevad kategooriad sisaldama tooteid erinevatest toidupoodidest (joonis 12). Kui administraator soovib lisada Prisma õunte kategooria veebilehte, siis peab administraator valima selle veebilehe jaoks

sobiva kategooria, mis on autori veebirakenduses ning valima ka nimekirjast sobiva toidupoe (joonis 11).

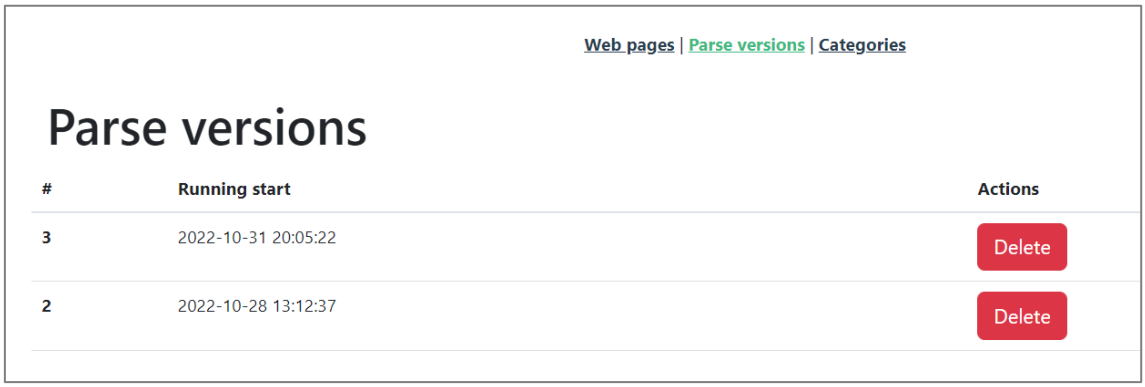


Joonis 11. Veebilehtede haldamise leht veebirakenduse administraatoripaneelil



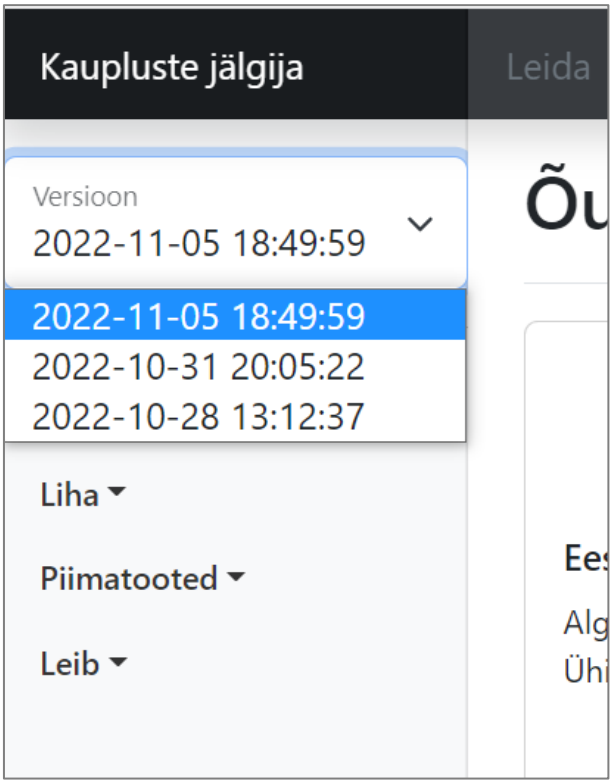
Joonis 12. Autori veebirakenduse õuna kategooria veebileht

Samuti saab administraator kustutada veebikraabitsa andmete kogumise erinevaid versioone (joonis 13), mistõttu kasutajad ei näe veebikraabitsaga kogutud tooteid kindlal kuupäeval, kuna need andmebaasist eemaldatakse.



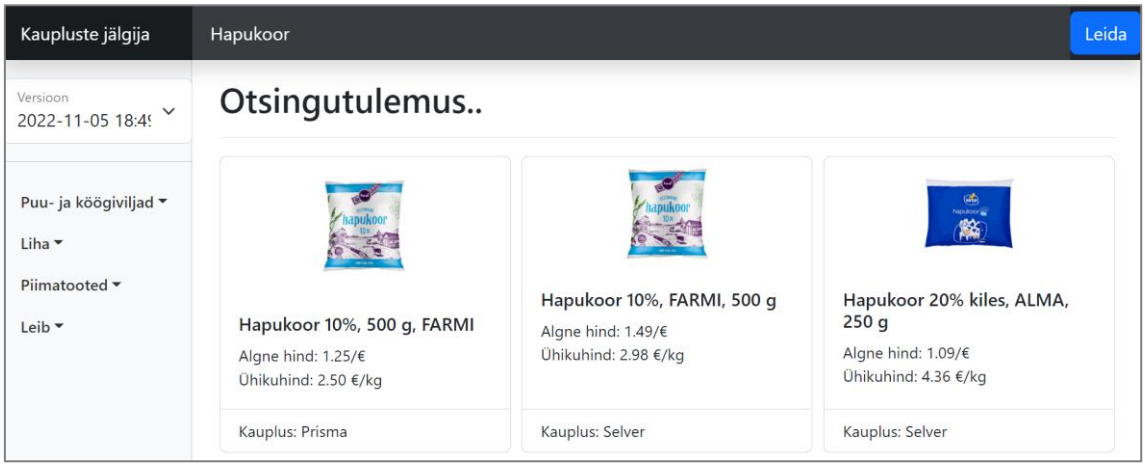
Joonis 13. Veebikraabitsate versioonide haldusleht veebirakenduse administraatoripaneelil

Autori veebirakenduses saavad kasutajad ka vahetada versioone (joonis 14) ja näha toodete infot konkreetsetel päevadel, mil veebikraabits kogus andmeid toidupoodide veebilehtedelt.



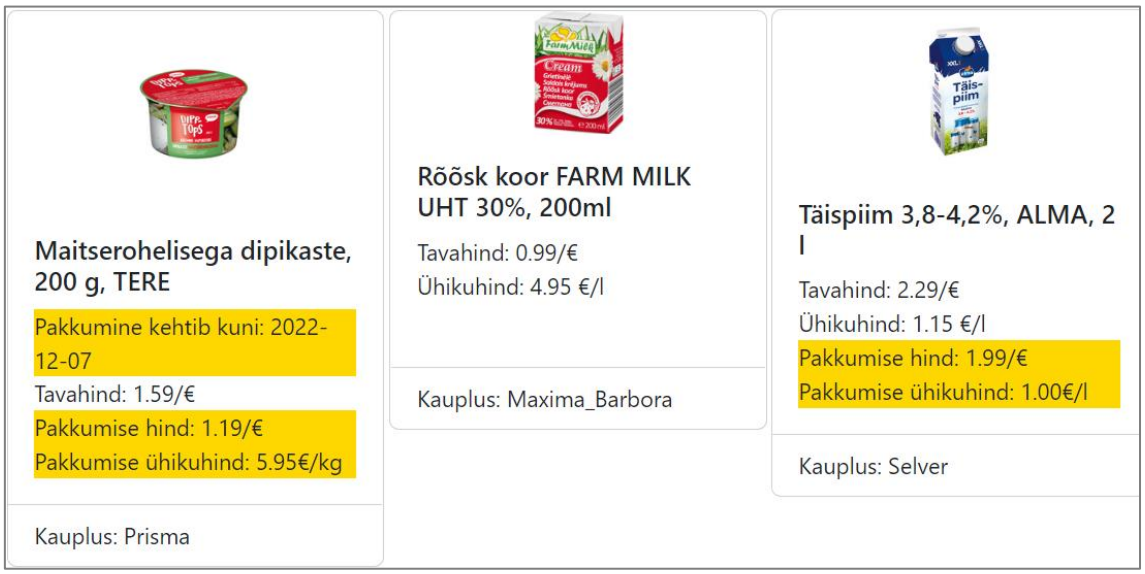
Joonis 14. Versioonide valimine veebirakenduses

Kasutajad saavad tooteid otsida ka nime järgi, sisestades otsingukasti vähemalt 3 tähte ja klõpsates nuppu „Leida“ (joonis 15).



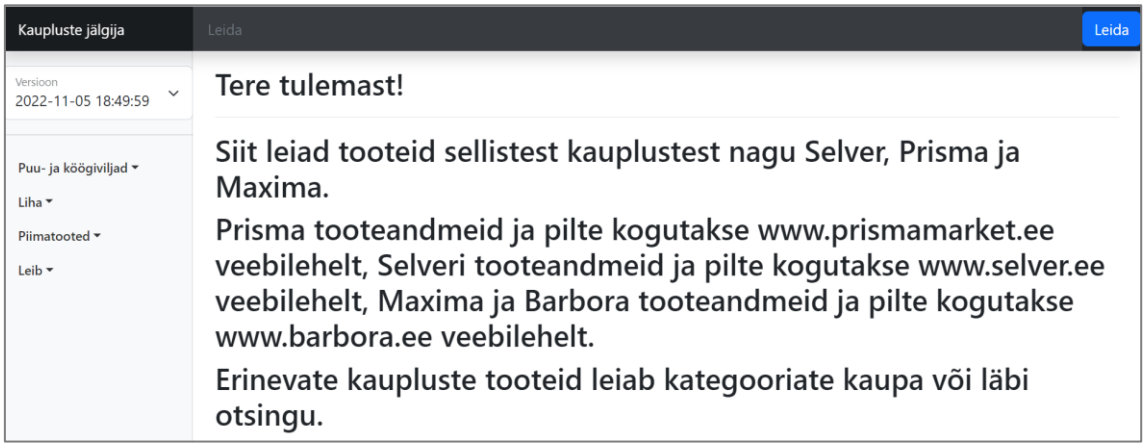
Joonis 15. Toodete otsing autori veebirakenduses

Samuti saavad autori veebirakenduse kasutajad näha toodete soodushindu, kui tootel on allahindlus. Samuti näevad kasutajad, mis kuupäevani allahindlus kehtib, kui veebilehel, kust veebikraabits tooteandmeid kogus, oli see teave olemas. Tootel on sellised andmed nagu: toote nimetus, toote tavahind, ühikuhind (€/kg, €/l, jne), pakkumise hind, toote ühiku pakkumise hind, pakkumise periood, ühik (kg, l, jne) ja tootefoto, kui need andmed on veebikraabitsa andmete kogumise ajal olemas, ja millisest toidupoest toode pärit on. Joonisel 16 on näha 3 erinevat erineva infoga toodet.



Joonis 16. Erineva infoga toodet autori veebirakenduses

Veebrakenduse avalehel saavad kasutajad näha teavet autori veebirakenduse kohta (joonis 17). Samuti märkis autor, kust autor toodete kohta infot kogub.



Joonis 17. Veebirakenduse avaleht

Autori veebirakendusel on ka kohanemisvõimeline kujundus mobiilseadmetele (joonis 18).

Joonis 18. Kujundus mobiilseadmes

3.3 Veebikraabitsa kirjeldus

Autor lõi Pythoni programmeerimiskeele versioonis 3.10.7 veebikraabitsa, mis oli veebikraabitsa kirjutamise ajal Pythoni keele uusim versioon. Iga toidupoe veebilehte jaoks valis autor kõige sobivama Pythoni teeki. Ja nende teeke abil kraabis autor andmeid toidupoodide veebilehtedelt.

Esiteks loob see veebikraabits ühenduse rakenduste andmebaasiga, seejärel loob veebikraabits uue versiooni kuupäevaga, millal autor veebikraabitsa käivitas (joonis 19).

```
def start_new_version(dbcon):
    cursor = dbcon.cursor()
    running_start = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    cursor.execute("INSERT INTO parse_versions (running_start) VALUES (%s)",
(running_start,))

    dbcon.commit()
    return cursor.lastrowid
```

Joonis 19. Uue versiooni loomine

Seejärel käivitatakse meetodid, mis koguvad andmeid toidupoodide kategooriate veebilehtedelt (joonis 20) ja lisavad need andmebaasi. Veebikraabits hangib veebilehed andmebaasist iga toidupoe identifikaatori (joonis 21) järgi. Igal meetodil on oma spetsiaalne andmekogumiskood, mis kasutavad erinevaid teeke. Kood saadab nendele meetoditele kategooriate veebilehed, kust nad peavad koguma tooteandmeid, andmebaasi ühendusstringi ja versiooni identifikaatori koos kuupäevaga ajal, mil autor veebikraabitsa käivitas.

```
if __name__ == '__main__':
    dbcon = db.db_connect()
    version_id = start_new_version(dbcon)
    #parse SELVER
    print('start selver')
    pages_selver = selver.get_urls(dbcon)
    selver.parse(pages_selver, dbcon, version_id)
    # parse Prisma
    print("start prisma")
    pages_prisma = prisma.get_urls(dbcon)
    prisma.parse_data(pages_prisma, dbcon, version_id)
    # parse Maxima
    print('start maxima')
    pages_maxima = maxima.get_urls(dbcon)
    maxima.parsing(pages_maxima, dbcon, version_id)
```

Joonis 20. Kraabimise meetodeid

```
 def get_urls(dbcon):
    myCursor = dbcon.cursor(buffered=True, dictionary=True)

    myCursor.execute("SELECT id, category_id, store_id, url FROM web_pages
WHERE store_id = %s", (Stores.SELVER.value,))

    pages = []
    for row in myCursor:
        pages.append({'category': row['category_id'], 'store':
row['store_id'], 'web_page': row['id'],
                      'next_url': row["url"]})
    return pages
```

Joonis 21. Kood kategooriate veebilehtede andmebaasist kogumiseks

Esmalt käivitatakse veebikraabits kategooriate veebilehtede jaoks, millel on Selveri toidupoe tooteandmed, seejärel Prisma ja lõpuks Maxima (Barbora).

Veebikraabits kogub selliseid andmeid nagu: toote nimetus, toote tavahind, ühikuhind (€/kg, €/l, jne), pakkumise hind, toote ühiku pakkumise hind, pakkumise periood, ühik (kg, l, jne) ja link tootefotole, kui need andmed on veebikraabitsa andmete kogumise ajal olemas.

Samuti lisas autor igale toidupoodide veebikraabitsale funktsionaalsuse, mis kontrollib, kas kategooria veebilehes on järgmine lehekülg samasse kategooriasse kuuluvate toodetega, kui on, siis lähevad veebikraabitsad järgmisele lehele ja koguvad sealt andmeid ja nii edasi, kuni need jõuavad veebilehe viimasele lehele. Iga andmebaasi veebileht on toidupoe kategooria veebilehe, millel on teatud kategooriasse kuuluvaid tooteid.

3.3.1 Prisma veebikraabits

Iga andmebaasist võetud Prisma kategooria veebilehe kohta saadab veebikraabits Requests Pythoni teeki kasutades HTTP päringu päisega „Accept: application/json“, mille tulemusena saab veebikraabits Prismalt JSON-tulemuse koos kõigi selle kategooria tooteandmetega. JSON-objekti võtmete abil saab hankida toote andmed (joonis 22). Kui kategooria veebilehes on järgmine leht, siis sisaldab saadud JSON-tulemus ka linki veebilehe järgmisele lehele (joonis 23). Seega liigub veebikraabits neid lehti tsükliliselt läbi ja kogub tooteandmeid ning sisestab need andmebaasi.

```
"entries": [
    {
        "ean": 4770074472077,
        "name": "Pruun roosuhkur 750 g",
        "subname": "DANSUKKER",
        "quantity": 0.75,
        "unit_name": "kg",
        "deliverable": true,
        "orderable": true,
        "entry_attributes": [],
        "aisle": "Kuivained",
        "aisle_number": 13,
        "commentable": true,
        "entry_user_notes": [],
        "images": 1,
        "image_guid": "ad2421d891c1c9b9fb150929fee3baaf",
        "product_class": "package",
        "price": 2.59,
        "comp_price": 3.45,
        "comp_unit": "kg",
        "sales_unit_name": "tk",
        "contains_alcohol": false,
        "locker_deliverable": true
```


Joonis 22. JSON-objekt, mis sisaldab tooteandmeid


```
"pagination": {
    "prev_class": "disabled",
    "prev_url": null,
    "prev_link_class": "js-category-item",
    "next_class": "",
    "next_url": "/products/17316/page/2",
```


Joonis 23. JSON-objekt, mis sisaldab järgmist kategoorialehte

Osa Prisma veebikraabitsa koodist on toodud lisas 2.

3.3.2 Maxima veebikraabits

Iga andmebaasist võetud Maxima (Barbora) veebilehe kohta saadab veebikraabits BeautifulSoup Pythoni teegi abil HTTP päringu, mille tulemusena saab veebikraabits igalt kategooria veebilehelt HTML elemendid koos kõigi andmetega. kus on ka tooteandmed. Samuti, kui veebilehel on järgmine lehekülg toodetega, siis veebikraabits leiab veebilehe HTML elementide hulgast lingi järgmisele lehele ehk võtab lingi nupult, mis viib järgmisesse kategooriasse lehele ja läheb sellele ka tooteandmete kogumiseks. Seega liigub veebikraabits neid lehti tsükliliselt läbi ja kogub tooteandmeid ning sisestab need andmebaasi. Veebikraabits leiab andmed kategooria veebilehelt, kasutades HTML-

elemente ja nende atribuute. Osa Maxima (Barbora) veebikraabitsa koodist on toodud lisas 3.

3.3.3 Selver veebikraabits

Alustuseks avab veebikraabits Selenium Pythoni teeki kasutades brauseri ja läheb Selveri toidupoe põhiveebi, seejärel võtab veebikraabits andmebaasist ükshaaval veebilehed ehk lingid kategooriate veebilehtedele. ning nende linkide abil otsib Selveris kategooriaid, millel on lingid nendele kategooriate veebilehtedele ja klikib neid ükshaaval ning läheb seega nendele kategooriatele ehk siis veebilehtedele, millel on vastavad tooted ja veebikraabits kogub andmeid veebilehe HTML-elementidest.

Kategooriad Selveri veebilehel on need nupud, mis sisaldavad linke vastavate toodetega kategooriate veebilehtedele, veebikraabits võtab andmebaasist veebilehe, mis on link kategooriale ja veebikraabits otsib sarnase lingiga kategooriat, leiab, klõpsab ja läheb sellele veebilehele. Autor otsustas seda teha selleks, et mitte iga kord veebilehti uuesti avada, kuna Seleniumi teek simuleerib brauseriga töötamist ehk siis veebikraabits avab ja sulgeb iga kord brauseri, minnes toodetega Selveri veebilehtedele , selle asemel, et avada üks kord brauser ja liikuda Selveri kategooria veebilehele. Samuti, kui kategooria veebilehel on selles kategoorias järgmised tooted, siis veebikraabits otsib nuppu ja klõpsab sellel nupul, mis läheb järgmisele lehele ning kogub ka nendelt andmeid, seega liigub veebikraabits neid lehti tsükliliselt läbi ja kogub tooteandmeid ning sisestab need andmebaasi. Osa Selveri veebikraabitsa koodist on toodud lisas 4.

3.3.4 IP-aadressi blokeerimise vältimine

Et vältida autori IP-aadressi blokeerimist veebilehtedelt andmete kogumisel, pani autor koodi sisse pause. See tähendab, et enne uuelt veebilehelt andmete kogumist ootab veebikaabits mõnda aega, enne kui saadab päringuid andmete kogumiseks järgmistele veebilehtedele.

Samuti kasutas autor veebilehtedele päringute saatmisel spetsiaalseid HTTP päiseid, üks HTTP päistest oli kasutajaagent (inglese keeles „User-Agent“). Kasutajaagendi päringu päis on iseloomulik string, mis võimaldab serveritel ja võrgukaaslastel tuvastada rakenduse ja operatsioonisüsteemi. [35] Seega saavad toidupoodide veebilehed, kus veebikraabits päringuid saadab, aru, et tegemist on päris kasutajaga, mitte robotiga.

Kui autor on IP-aadressi tõttu blokeeritud, peaks autor kasutama puhverserverit. Puhverserver toimib kasutaja ja interneti vahelise väravana. See on vaheserver, mis eraldab lõppkasutajaid veebilehtedest, mida nad sirvivad. Proxy serverid pakuvad erinevat funktsionaalsust, turvalisust ja privaatsust sõltuvalt kasutaja kasutusjuhtumist, vajadustest või ettevõtte poliitikast. Kui kasutaja kasutab puhverserverit, voolab internetiliiklus läbi puhverserveri teel soovitud aadressile. Seejärel tuleb taotlus sama puhverserveri kaudu tagasi ja seejärel edastab puhverserver veebisaidilt saadud andmed kasutajale. [36]

Kui autori IP-aadress oleks blokeeritud, teeks autori veebikraabits teise IP-aadressi all päringuid, kasutades selleks puhverserverit, mida toidupoe veebileht pole veel blokeerinud.

3.4 Veebirakenduse tagakülje kirjeldus

Veebirakenduse tagaotsa ehk RESTful API jaoks kasutas autor PHP versiooni 8.1.6 programmeerimiskeelt, mis oli veebirakenduse kirjutamise ajal PHP uusim versioon. Samuti kirjutas autor kogu veebirakenduse tagaosa Laravel 9 PHP raamistikus, mis on veebirakenduse kirjutamise ajal Laraveli uusim versioon.

3.4.1 Veebirakenduse tagakülje kujundusmuster

Veebirakenduse tagaküljel kasutas autor teenusehoidla (inglese keeles „Service-Repository“) kujundusmustrit, kuna see on puhas ja jätkusuutlik. Hoidlate ja teenuste kontseptsioon tagab korduvkasutatava koodi kirjutamise ja aitab hoida kontrolleri võimalikult lihtsana, muutes need loetavamaks. [37]

Hoidlad on tavaliselt rakenduse mudeli tavaline ümbris ja koht, kuhu arendaja kirjutab rakenduse andmebaasi erinevaid päringuid. Teisest küljest on teenus kogu rakenduse loogika käsitlemise kiht. Kogemuste põhjal on mudeli loogika ja ümbrise eraldamine tõesti soodne, eriti kui töötate meeskonnas või suurtes projektides. [37]

Kui veebirakenduse administraator saadab veebirakenduse tagaküljele DELETE-päringu ülemkategooria kustutamiseks, käivitub esmalt kontroller, mis võtab päringu vastu, käivitab teenuse ja saadab päringu andmed teenusele (joonis 24).

```
 public function destroy(Category $category): JsonResponse
{
    //service
    $delete = $this->categoryService->deleteCategory($category);

    if($delete){
        return response()->json($delete);
    }
    return response()->json(['error' => 'Bad request'], 400);
}
```

Joonis 24. Kontroller kategooria kustutamiseks

Seejärel toimub teenuses äriloogika, see teenus kontrollib esmalt, kas andmebaasis on alamkategooria, mis on seotud ülemkategooriaga, mida administraator soovib kustutada, kui ei, siis käivitub teine hoidla, mis kustutab ülemkategooria, seejuures kasutab teenus oma loogika rakendamiseks mitmeid hoidlaid (joonis 25).

```
public function deleteCategory(Category $category): ?Category
{
   // 1 repository
   $childCategory = $this->categoryRepository->getCategory($category);
   if(!$childCategory){
       // 2 repository
       return $this->categoryRepository->delete($category);
   }
   return null;
}
```

Joonis 25. Teenus kategooria kustutamiseks

Iga hoidla teeb ainult ühe toimingu: tagastab andmed, kustutab andmed, lisab andmeid ja teenus saab kasutada nende hoidlate kombinatsiooni, olenevalt sellest, kui keeruline on projekti äriloogika. Hoidla kasutab samal ajal mudelit, mudel on tabel andmebaasis, seda mudelit kasutades pääsete andmebaasi juurde, selles näites kategooria kustutamiseks (joonis 26).

```
public function delete(Category $category): Category
{
   $category->delete();

   return $category;
}
```

Joonis 26. Hoidla kategooria kustutamiseks

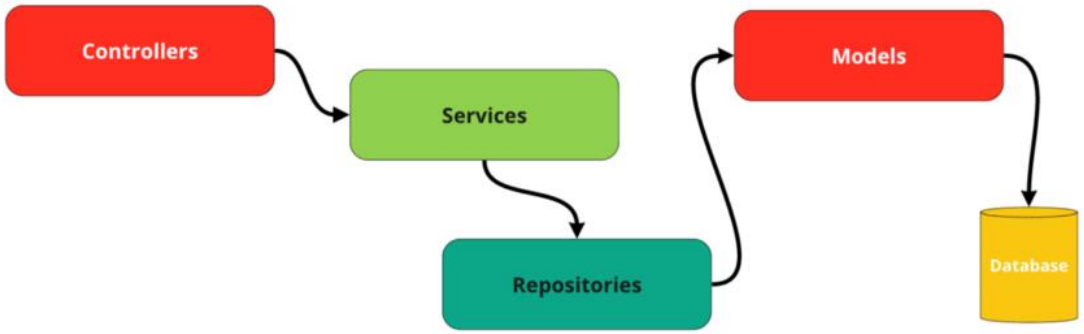
Iga mudel on seotud olemiga, st andmebaasis oleva tabeliga, lühidalt öeldes see mudel on see tabel andmebaasis, antud juhul kategooriatabel (joonis 27). Samuti saab seda mudelit

kasutades hõlpsasti aru, et veebirakenduse kategoorias on palju tooteid, st tooteandmeid, mida veebikraabits kogub.

```
class Category extends Model
{
   const UPDATED_AT = null;
   const CREATED_AT = null;

   public $table = "categories";

   public function products(): HasMany
   {
       return $this->hasMany(Product::class);
   }
}
```

Joonis 27. Kategooria mudel

Joonisel 28 visualiseerib autor seda mustri, mida autor eespool kirjeldas ja mida autor kasutab kogu veebirakenduse tagaküljel ehk siis RESTful API-l.



Joonis 28. Teenusehoida kujundusmustri skeem [38]

Selle kujundusmustri rakendamiseks kasutas autor Laraveli raamistiku tööriistu, nagu Eloquent ORM ja automaatne sõltuvuse süstimine.

Laravel sisaldab Eloquenti, objektide suhte kaardistajat (ORM), mis muudab arendaja andmebaasiga suhtlemise nauditavaks. Eloquenti kasutamisel on igal andmebaasitabelil vastav „mudel“, mida kasutatakse selle tabeliga suhtlemiseks. Lisaks kirjete otsimisele andmebaasi tabelist võimaldavad Eloquent mudelid sisestada, värskendada ja kustutada ka kirjeid tabelist. [39]

Autori veebirakenduse tagakülje struktuur, mis näitab teenusehoidla kujundusmustrit on toodud lisas 5.

Joonisel 29 on kood, mis näitab RESTful API marsruute, kuhu saab päringuid saata, iga marsruudi jaoks kasutatavaid kontrollereid, päringute tüüpe ja kontrollerites kasutatavaid meetodeid. Joonis 29 näitab kõiki veebirakenduse tagaküljel olevaid iseärasused.

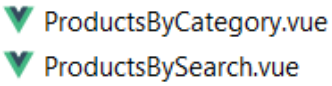
```
Route::get('categories', [CategoryController::class, 'index']);
Route::get('category/{category}/{parseVersionId?}',
[CategoryController::class, 'show']);
Route::post('category/add', [CategoryController::class, 'store']);
Route::put('category/update/{category}', [CategoryController::class,
'update']);
Route::delete('category/delete/{category}', [CategoryController::class,
'destroy']);

Route::get('parse-versions', [ParseVersionController::class, 'index']);
Route::delete('parse-versions/delete/{parseVersion}',
[ParseVersionController::class, 'destroy']);

Route::get('products/search/{title}/{parseVersionId?}',
[ProductController::class, 'search']);

Route::get('web-pages', [WebPageController::class, 'index']);
Route::post('web-page/add', [WebPageController::class, 'store']);
Route::put('web-page/update/{webPage}', [WebPageController::class,
'update']);
Route::delete('/web-page/delete/{webPage}', [WebPageController::class,
'destroy']);

Route::post('login', [UserController::class, 'login']);
Route::post('authorize', [UserController::class, 'isAuthorized']);
```

Joonis 29. RESTful API marsruute

3.5 Veebirakenduse esiosa kirjeldus

Veebirakenduse esiosa tegi autor Vue.js raamistiku uusima versiooni peal TypeScript programmeerimiskeeles. Seda klienti (veebirakenduse esiosa) kasutades saab saata päringuid RESTful API-le (veebirakenduse tagaküljele). Nii saavad kasutajad kasutada autori veebirakenduse toodete vaatamiseks, toodete otsimiseks, erinevate toidupoodide toodete erinevate versioonide vaatamiseks, kust veebikraabits on andmeid kogunud. Samuti saab seda klienti kasutava veebirakenduse administraator lisada, värskendada, kustutada kategooriaid, lisada, värskendada, kustutada veebilehti, millelt veebikraabits tooteandmeid kogub, ja kustutada toodete erinevaid versioone. Samuti saab seda klienti kasutav administraator sisse logida, et pääseda administraatoripaneelile, kus saab ülaltoodud toiminguid teha.

Vue.js raamistikus koosneb kogu projekt komponentidest, seega püüdis autor luua selliseid komponente, mis oleksid korduvkasutatavad. See tähendab, et autor kirjutas koodi nii, et komponendid oleksid korduvkasutatavad. Veebirakenduses on lehed, mis näitavad tooteid kategooriates ja otsingu järgi, st need on kaks erinevat lehekülge ja need on samuti kaks erinevat komponenti (joonis 30).



Joonis 30. Kaks komponenti, mis näitavad tooteid

Et mitte kirjutada igale komponendile koodi, mis kuvab toote kohta teavet, lõi autor ühe komponendi nimega „ProductComponent“ (eesti keeles „Tootekomponent“) (joonis 31) ja kasutab seda kahes teises komponendis. Seega muutub kood väiksemaks ja projekti on palju lihtsam hooldada.

```
<template>
  <div class="card">
    <img v-if="product.image_url" :src="product.image_url" class="card-img-
top" alt="..." width="100px">
    <img v-else src="/dummyimage.png" class="card-img-top" width="100px">
    <div class="card-body">
      <h6 class="card-title">{{ product.title }}</h6>
      <p class="card-text offer-box" v-
if="product.offer_until_date">Pakkumine kehtib kuni: {{
product.offer_until_date }}</p>
      <p>Tavahind: {{ product.original_price }}/€</p>
      <p v-if="product.unit_price">Ühikuhind: {{ product.unit_price }}
€/<template v-if="product.unit">{{product.unit}}</template>
      </p>
      <div class="offer-box">
        <p v-if="product.offer_price">Pakkumise hind: {{ product.offer_price
}}/€</p>
        <p v-if="product.offer_unit_price">Pakkumise ühikuhind: {{
product.offer_unit_price }}€/<template v-if="product.unit">{{ product.unit
}}</template>
        </p>
      </div>
    </div>
    <ul class="list-group list-group-flush">
      <li class="list-group-item">Kauplus: {{ stores[product.store_id]
}}</li>
    </ul>
  </div>
</template>
```

Joonis 31. „ProductComponent“ kood

3.6 Veebirakenduse administraatori paneeli turvalisus

Autentimine on protsess, mis kontrollib, et keegi või miski on see, kelleks nad end ütlevad. Tehnoloogiasüsteemid kasutavad rakendusele või selle andmetele juurdepääsu tagamiseks tavaliselt mõnda autentimise vormi. Näiteks kui kasutajal on vaja juurdepääsu veebirakendusele või võrguteenusele, peab kasutaja tavaliselt sisestama oma kasutajanime ja parooli. Seejärel võrdleb see kulisside taga sisestatud kasutajanime ja parooli oma andmebaasis oleva kirjega. Parool andmebaasis on salvestanud krüpteeritud kujul ning spetsiaalse meetodi abil võrreldakse kasutaja sisestatud parooli. Kui kasutaja esitatud teave ühtib, eeldab süsteem, et kasutaja on kehtiv kasutaja, ja annab kasutajale juurdepääsu. [40]

Autoriseerimine on turbeprotsess, mis määrab kasutaja või teenuse juurdepääsutaseme. Tehnoloogias kasutavad rakendused autoriseerimist, et anda kasutajatele või teenustele juurdepääs teatud andmetele või teatud toimingu tegemiseks. [40]

Sellel autori veebirakendusel pole registreerimist, on ainult autentimine ja administraatori autoriseerimine, kuna hetkel pole rakenduse kasutajatel mõtet registreeruda, kasutajad saavad tooteid vaadata ja otsida juba ilma igasuguse registreerimiseta.

Veebirakenduse tagaküljel on eriklass ehk vahevara (inglise keeles *middleware*), seda klassi kasutades kontrollitakse administraatori nime ja parooli, kui admin sisestas andmed valesti, siis tagastab autori rakendus JSON-vastus veaga ja olekuga 400, seega saab klient aru, et andmed on valesti sisestatud ja näitab viga administraatorile. Joonisel 32 on kood, mis seda kontrolli veebirakenduse tagaküljel teostab.

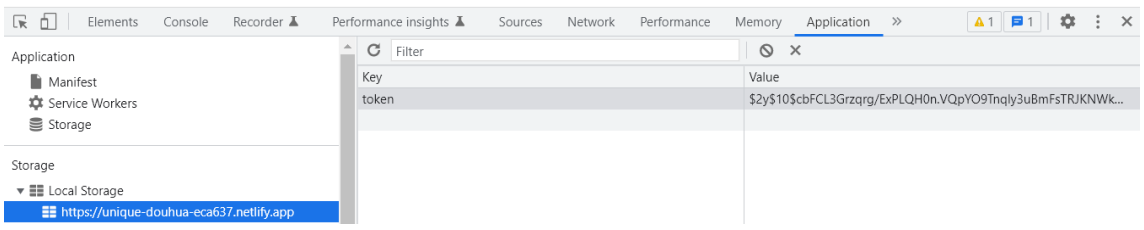
```
class Authenticate
{

    public function handle(Request $request, \Closure $next)
    {
        $userData = $request->only([
            'name',
            'password',
        ]);
        $user = User::where('name', $userData['name'])->first();
        if($user === null || !Hash::check($userData['password'], $user-
>password)){
            return response()->json(['Error'=>"Bad request"], 400);
        }

        return $next($request);
    }
}
```

Joonis 32. Autentimise vahevara klass

Kui administraator sisestas oma andmed õigesti, genereerib veebirakenduse tagaosa spetsiaalse krüptitud sõna, lisab selle andmebaasi ja tagastab selle sõnaga JSON-vastuse, igal autentimisel värskendatakse seda sõna turvalisuse huvides andmebaasis eesmärkidel. Kui veebirakenduse esiosa selle krüpteeritud sõna saab, salvestab see antud sõna kasutaja brauseri mällu (joonised 33, 34).

```
  methods: {
    async submitLogin(){
      this.failMsg = ""
      const response = await HttpService.Post<User>('login', this.user)
      if(response.ok){
        localStorage.setItem('token', (response.data as User)?.token)
        this.$router.push('/admin/web-pages')
      }else{
        this.failMsg = "Wrong password or name!";
      }
    }
  }
```

Joonis 33. Veebirakenduse esiosa autentimise kood



Joonis 34. Krüpteeritud sõna brauseri mällus

Pärast edukat autentimist suunatakse administraator administraatori paneelile, st spetsiaalsele veebilehele, kus administraator saab juba oma asju ajada .

Samuti lisas autor turvalisuse huvides veebirakendusele autoriseeringu, kui mitte admin, vaid mõni teine kasutaja teab kuidagi seda linki, mis viib administraatori paneeli, ehk siis siseneb administraatori paneeli ilma autentimiseta, siis autori veebirakendus peab kasutaja administraatoripaneelilt autentimiseks lehele ümber suunama, st ilma autentimiseta ja ilma autoriseerimata veebirakendus ei luba kasutajal siseneda administraatori paneeli, kus saab kustutada, värskendada ja lisada erinevaid andmeid.

Veebirakenduse esiosa igal marsruudil, mis viivad administraatori paneeli lehtedele (joonis 35), on spetsiaalne kontroll (vahevara) (joonis 36), mis toimub vahetult enne kasutajal veebirakenduse määratud lehele ilmumist.

```
{
  path: '/admin/categories',
  name: 'adminCategories',
  component: CategoriesView,
  beforeEnter: async () => {
    if(! await auth()){
      return 'admin/login'
    }
  }
},
{
  path: '/admin/web-pages',
  name: 'adminWebPages',
  component: WebPagesView,
  beforeEnter: async () => {
    if(! await auth()){
      return 'admin/login'
    }
  }
},
{
  path: '/admin/parse-versions',
  name: 'adminParseVersion',
  component: ParseVersionsView,
  beforeEnter: async () => {
    if(! await auth()){
      return 'admin/login'
    }
  }
},
```

Joonis 35. Veebirakenduse esiosa administraatori paneeli marsruudid

```
export default async function auth() {
    const token = localStorage.getItem('token');
    if (!token) {
        return false
    }

    const authResponse = await HttpService.Post('authorize',null, token)
    return authResponse.ok;
}
```

Joonis 36. Veebirakenduse esiosa autoriseerimise vahevara kood

Veebirakenduse tagaküljel on klass (vahevara), mis autoriseerib kasutajat (joonis 37). Veebirakenduse esiosa abil saadetakse krüptitud sõna, mis on just võetud brauseri salvestusseansist, ja see krüpteeritud sõna saadetakse HTTP päringu abil veebirakenduse tagaküljele koos autoriseerimistaotluse päisega.

```
class Authorize
{
    public function handle(Request $request, \Closure $next)
    {

        $token = $request->bearerToken();
        $user = User::where('token', $token)->first();

        if($token && $user){
            return $next($request);
        }
        return response()->json(['Error' => 'Unauthorized'], 401);

    }
}
```

Joonis 37. Veebirakenduse tagaosa autoriseerimise vahevara kood

Veebirakenduse tagumine ots võtab selle krüptitud sõna vastu ja kontrollib, kas antud krüptitud parooliga kasutaja on olemas, kui jah, siis pääseb administraator juhtimispaneelile, ka ainult õige krüptitud parooliga saab administraator lisada, kustutada ja uuendada andmeid andmebaasist, kuna iga meetodi jaoks kontrolleris, mis suudab erinevaid andmeoperatsioone teha, lisas autor selle „vahevara“ (inglase keeles „middleware“) kontrolli (joonis 38).

Joonisel 38 on näidatud kood, mis käsib kontrolleril autoriseerida kasutajat kõigi meetodite jaoks, välja arvatud „show“ (eesti keeles „näitus“) meetod, mis lihtsalt tagastab kategooriate loendi. See tähendab, et tavaline autoriseerimata kasutaja näeb

veebirakenduses kategooriaid, kuid ei saa andmebaasis andmeid kustutada, uuendada ja lisada, seda saab teha ainult autoriseeritud kasutaja kes on administraator.

```
class CategoryController extends Controller
{
    protected CategoryService $categoryService;

    public function __construct(CategoryService $categoryService)
    {
        $this->categoryService = $categoryService;
        $this->middleware('can', ['except' => ['index', 'show']]); //auth
    }
```

Joonis 38. Kategooria kontroller koos vahevaraga

3.7 Veebirakenduse majutamine

Selles peatükis räägib autor, kuidas majutas veebirakenduse taga- ja esiosa.

3.7.1 Veebirakenduse tagakülje majutamine

Autor kasutas Zone veebimajutust. Zone virtuaalserver sisaldab meilimajutust, veebimajutust ja andmebaaside majutust, DNS-teenust ja muid kasulikke teenuseid. Autor valis selle veebimajutuse, kuna veebimajutus kuulub kohalikule ettevõttele ja sealt on kergesti küsimustele vastuseid saada. Ja autor oli seda veebimajutust juba varem kasutanud, probleeme polnud, samuti oli autoril juba aasta tellimus, enne kui autor oma lõputööd tegema hakkas.

3.7.2 Veebirakenduse esiosa majutamine

Veebimajutamiseks veebirakenduse esiküljel kasutas autor mugavat ja tasuta Netlify platvormi. Netlify majutab rakendust viisil, mis muudab selle arendaja jaoks väga lihtsaks. Põhimõtteliselt on see 3-etapiline protsess, mis tõestab pideva juurutamise teenust [41]:

- Arendage oma rakendust ja lükake GIT-i, autori puhul GitHub’is
- Looge Netlify'is järgu sätted, mis osutavad autori rakenduse GIT-hoidlale
- Juurutage oma rakendus

Suurepärane on see, et niipea, kui arendaja avaldab Netlify jälgitavas GIT-harus uusi muudatusi, tõmbab Netlify peaaegu koheselt välja, loob ja juurutab rakenduse uue versiooni, seda nimetatakse pidevaks juurutamiseks (Continuous Delivery). Netlify platvorm ühendab arendaja hoidlad kõikehõlmava töövooga ülemaailmse CDN-i juurutamiseks, pidevaks integreerimiseks, automaatseks ja tasuta HTTPS-i kasutamiseks. [41]

4 Rakenduse testimine

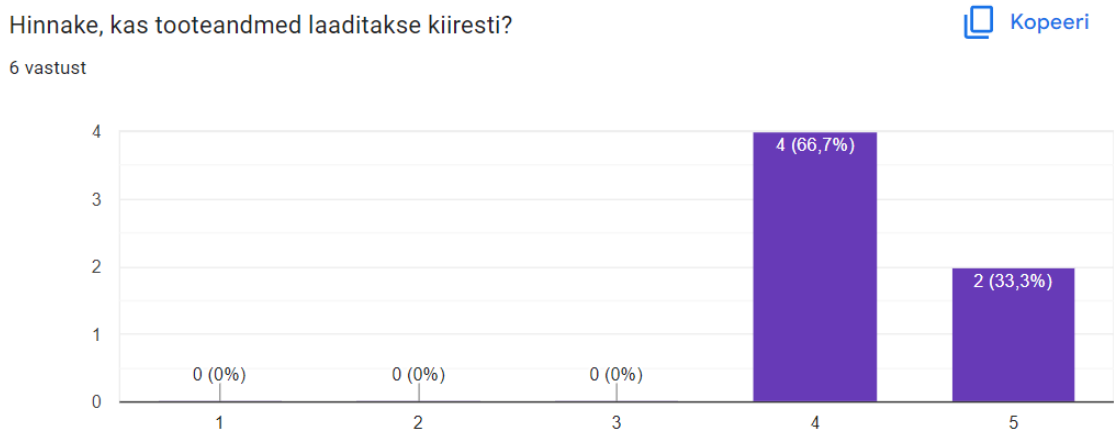
Autor andis oma rakenduse, mis koosneb veebirakendusest ja veebikaabitsast, erinevatele inimestele testimiseks, et veenduda rakenduse korrektses toimimises ja rakenduse täitmises.

4.1 Veebirakenduse testimine

Autor otsustas esmalt uurida, kas kõik funktsioonid töötavad ning kas veebirakendusega on mugav võrrelda ja vaadata erinevate toidupoodide toodete hindu ja infot kauplustes.

Autor saatis testijatele küsimustiku, kuhu nad saavad veebirakenduse kohta oma tagasiside jätta. Autor kasutas küsimustiku koostamiseks Google Formsi. Testijaid oli kokku 6, voim sisaldas 8 küsimust.

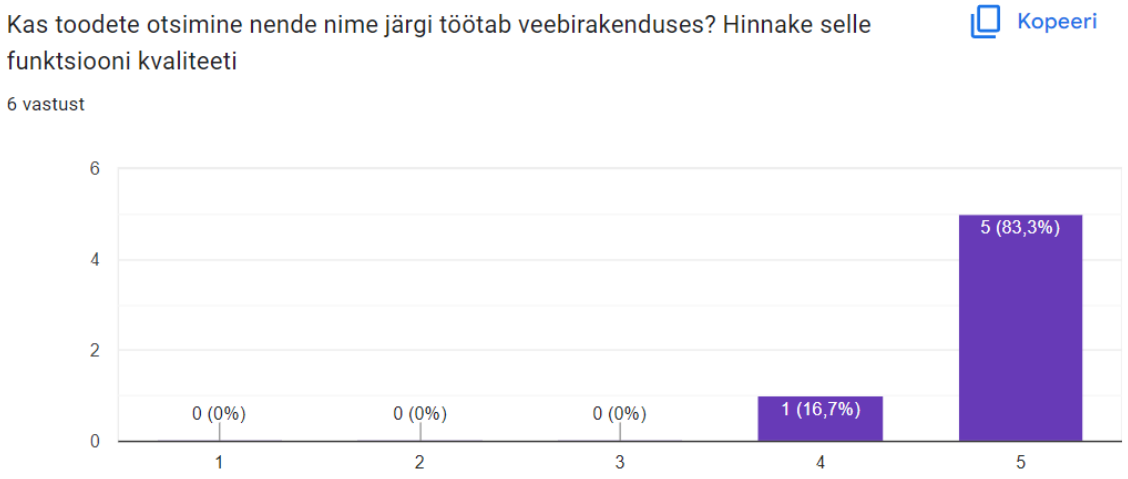
Esimene küsimus ja vastused sellele on toodud joonisel 39. Pärast testimist pidid kasutajad hindama (1 kuni 5) tooteandmete laadimise kiirust autori veebirakenduses



Joonis 39. Küsimustiku esimene küsimus

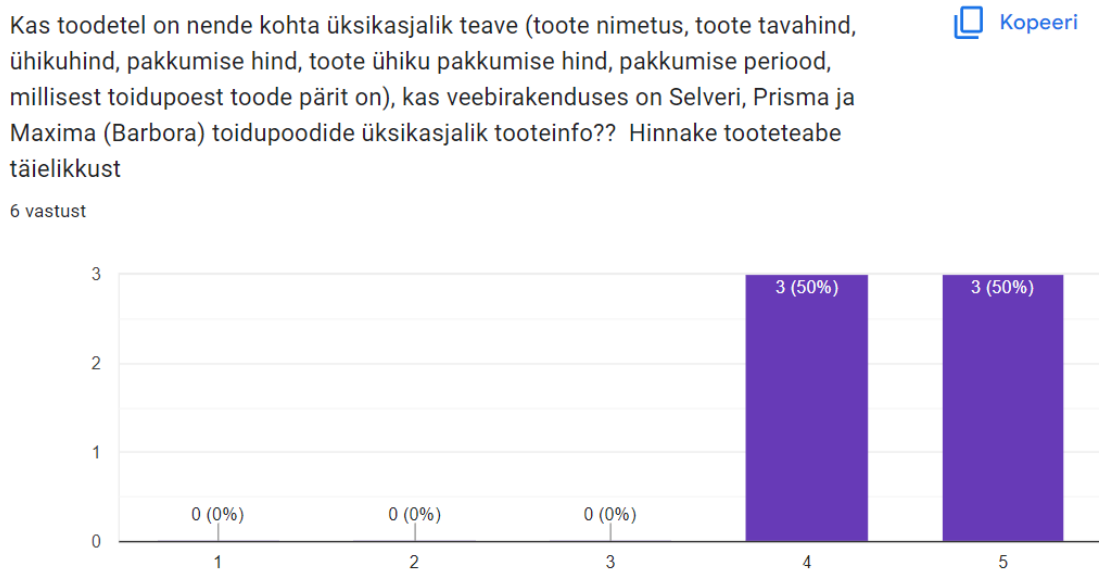
Joonisel 40 on näha teine küsimus ja vastused sellele, pärast testimist pidid kasutajad hindama (1 kuni 5), kui palju meeldis kasutajatele autori veebirakenduses tootenime järgi

otsingufunktsioon.



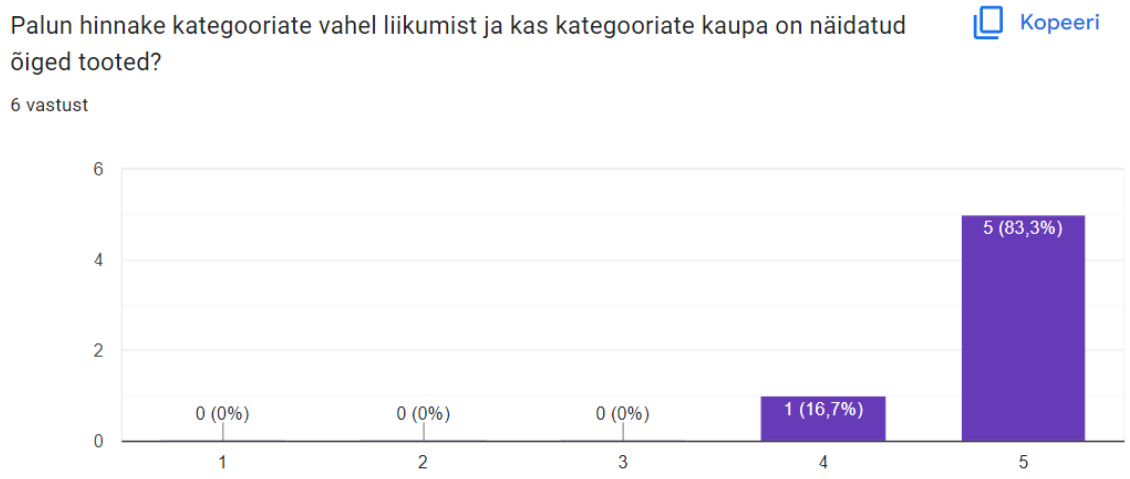
Joonis 40. Küsimustiku teine küsimus

Joonisel 41 on näha kolmas küsimus ja vastused sellele, pärast testimist pidid kasutajad hindama (1 kuni 5), kas autori veebirakenduses olevatel toodetel on piisavalt infot.



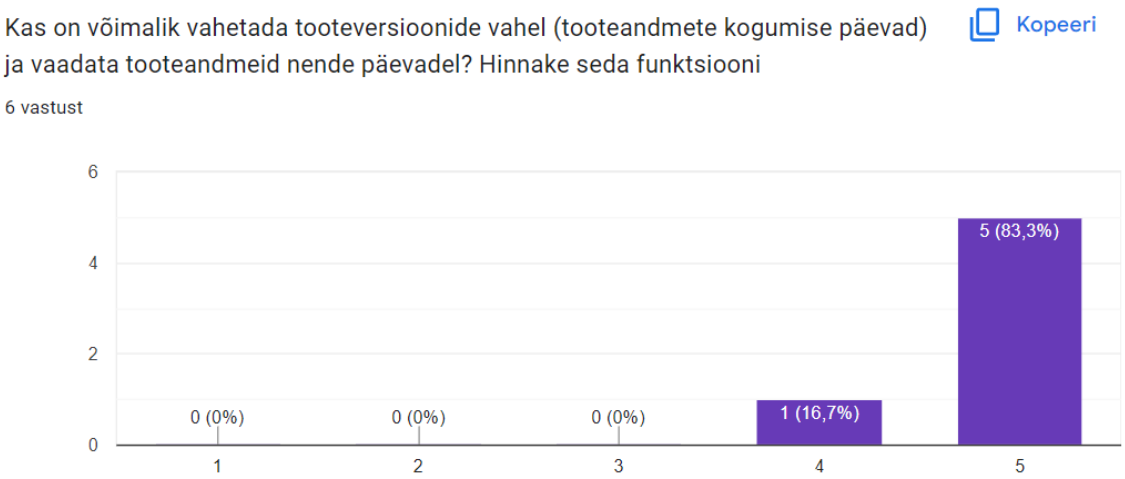
Joonis 41. Küsimustiku kolmas küsimus

Joonisel 42 on näha neljas küsimus ja vastused sellele, pärast testimist pidid kasutajad hindama (1 kuni 5), kui palju kasutajatele veebirakenduse kategooriate vahel navigeerimine meeldis ja kas kategooriatel on vastavad tooted.



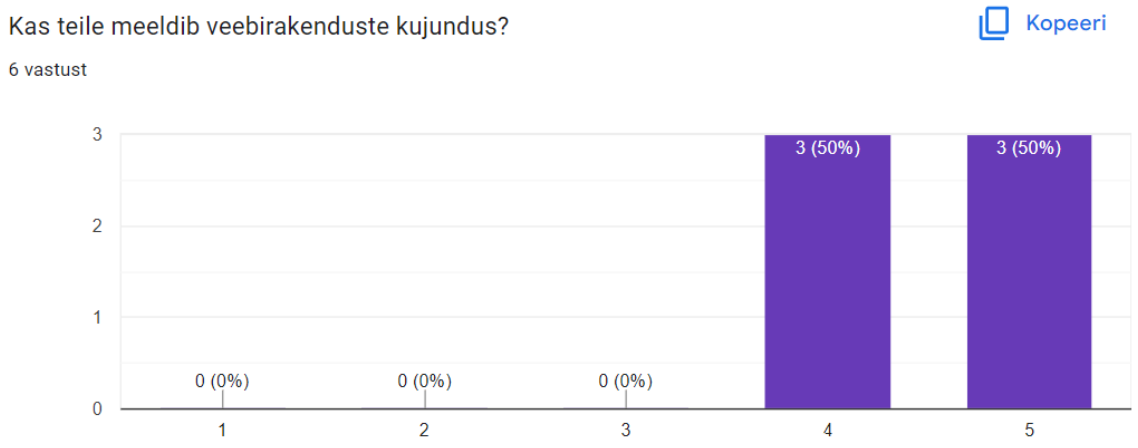
Joonis 42. Küsimustiku neljas küsimus

Joonisel 43 on näha viies küsimus ja vastused sellele, pärast testimist pidid kasutajad hindama (1 kuni 5), kui palju meeldis erinevate tooteversioonide vahel vahetamise funktsionaalsus.



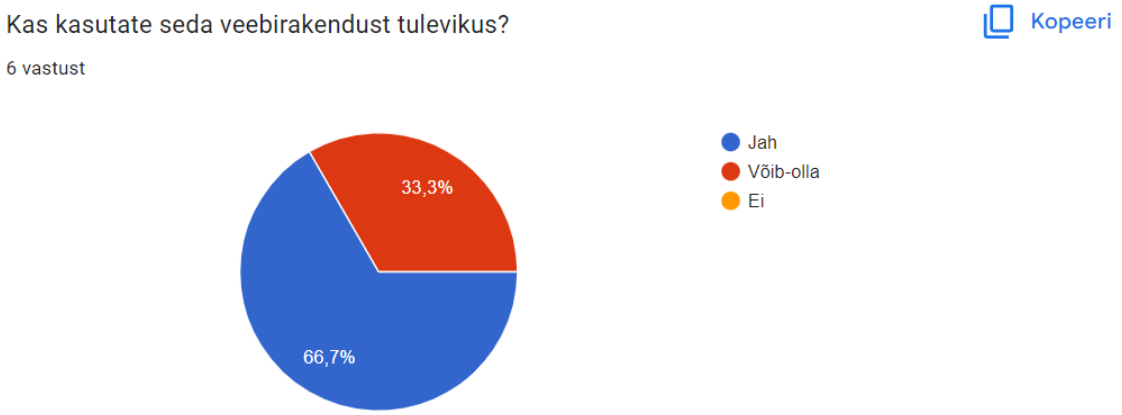
Joonis 43. Küsimustiku viies küsimus

Joonisel 44 on näha kuues küsimus ja vastused sellele, pärast testimist pidid kasutajad hindama (1 kuni 5), kui palju veebirakenduse kujundus meeldis.



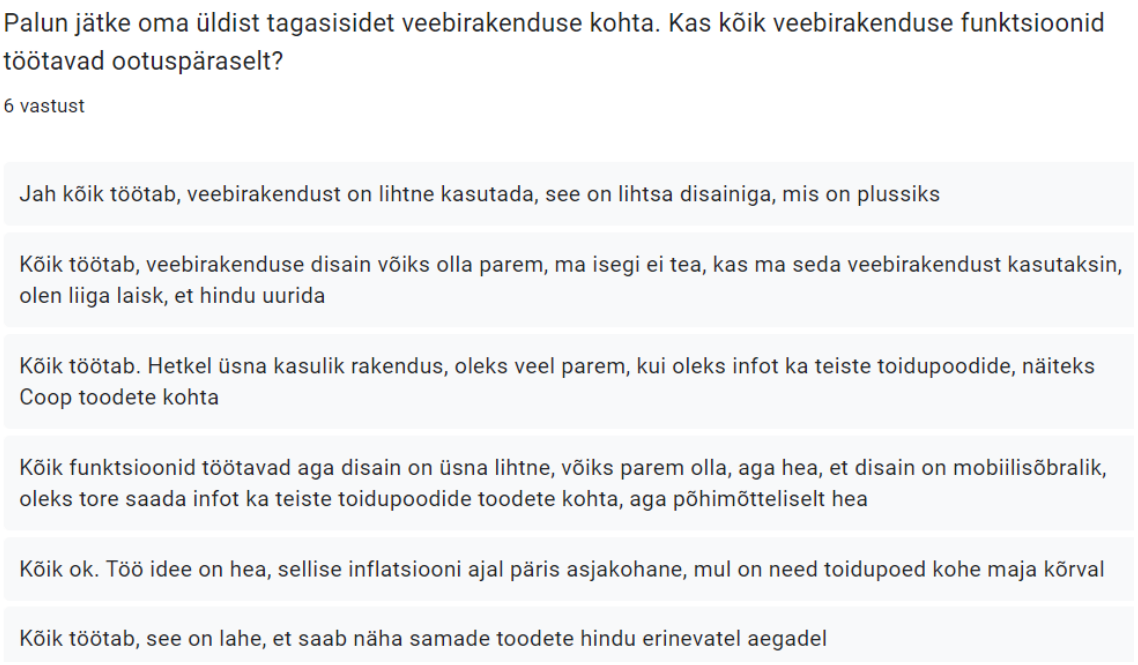
Joonis 44. Küsimustiku kuues küsimus

Joonisel 45 on toodud seitsmes küsimus ja vastused sellele, pärast testimist pidid kasutajad valima nimekirjast ühe vastuse ja seeläbi otsustama, kas pärast küsimustikule vastamist seda veebirakendust ka edaspidi kasutavad.



Joonis 45. Küsimustiku seitsmes küsimus

Joonisel 46 on kujutatud viimane küsimus ja vastused sellele, pärast testimist pidid kasutajad kirjutama üldise tagasiside autori veebirakenduse kohta.



Joonis 46. Küsimustiku kaheksas küsimus

4.2 Veebirakenduse administraatori paneeli ja veebikraabitsa testimine

Seejärel andis autor rakenduse kogenud Pythoni arendajale hindamiseks, et arendaja saaks testida veebirakenduse administraatoripaneeli ja veebikraabitsat. Antud juhul autor enam Google Formsi ei kasutanud, sest vastajaid oli vaid üks. Pärast seda, kui arendaja oli testinud veebirakenduse administraatori paneeli ja veebikraabitsat, koostas autor tabeli kõigi arendajale esitatud küsimuste ja arendajalt saadud tagasisidega (Tabel 2). Tagasiside andmeid võetakse rakenduse edasisel arendamisel arvesse.

Tabel 2. Küsimus ja tagasiside

Küsimus	Tagasiside
Kas autentimine administraatoripaneelil töötab? Kas sinna on võimalik minna?	Jah
Kas administraatoripaneelil on võimalik autori veebirakendusele kategooriaid lisada, ka uuendada ja kustutada?	Jah

Kas administraatoripaneelil on võimalik kategooriate veebilehti lisada, ka kustutada ja uuendada?	Jah
Kas administraatoripaneelil on võimalik versioone kustutada (kui veebikaabits tooteandmeid kogus)?	Jah
Kas veebikraabitsad töötavad toidupoodides nagu Maxima (Barbora), Prisma, Selver?	Jah, need töötavad, aga Selveri toidupoest andmete kogumine võtab kaua aega.
Kas autor on valinud õiged kraapimisteegid?	Jah, iga veebileht on erinev, seega valiti iga toidupoe veebilehe jaoks õiged Pythoni teegid.
Kas pärast seda, kui veebikraabits on toidupoodidest kõik andmed kokku kogunud, kas on võimalik näha tooteandmeid autori veebirakenduses?	Jah, pärast seda, kui veebikraabits on oma töö lõpetanud, saab näha tooteid veebirakenduses.

5 Kokkuvõte

Käesolevas bakalaureusetöös koostas autor rakenduse, mis koosneb veebikraabitsast ja veebirakendusest, mis suhtlevad autori loodud andmebaasiga. Autor on loonud veebikraabitsa, mis suudab koguda tooteandmeid toidupoodide nagu Prisma, Serlveri ja Maxima (Barbora) veebilehtedelt. Autor on loonud veebirakenduse, mis võimaldab kasutajatel vaadata tooteid kategooriate kaupa, otsida tooteid ning võrrelda erinevate toidupoodide toodete hindu interneti vahendusel, samuti on võimalik vaadata tooteid erinevatel kuupäevadel, millal veebikraabits andmeid kogus. Autor lõi veebirakendusse ka administraatori paneeli, et administraator saaks lisada, värskendada ja kustutada kategooriaid ja veebilehti ning administraator saab kustutada tooteversioone ajal, mil veebikraabits kogus andmeid. Samuti lisas autor veebirakendusele administraatori jaoks autentimise ja autoriseerimise, et veebirakendus oleks võimalikult turvaline.

Ka selles bakalaureusetöös analüüsis autor erinevaid tehnoloogiaid ja meetodeid veebirakenduse ja veebikraabitsa loomiseks ning autor analüüsis töö valminud praktilist osa.

Bakalaureusetöö eesmärk on täidetud, tulemuste põhjal võib järeldada, et töötav prototüüp on kasutusvalmis ja avatud täiendustele.

Kasutatud kirjandus

[1] „What is inflation?,“ European Central Bank, 2022, [Võrgumaterjal]. Available: https://www.ecb.europa.eu/ecb/educational/explainers/tell-me-more/html/what_is_inflation.en.html. [Kasutatud 01 10 2022].

[2] D. Barton, I. Nakov, D. Lukáč, „Web Scraping: The Beginner’s Guide.“ Apify, [Võrgumaterjal]. Available: https://blog.apify.com/web-scraping-guide/. [Kasutatud 01 10 2022].

[3] „Python Programming Language,“ GeeksforGeeks, 16 06 2022. [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/python-programming-language/. [Kasutatud 01 10 2022].

[4] ODSC Community, „5 Preferred Programming Languages for Web Scraping,“ Open Data Science, 11 08 2022. [Võrgumaterjal]. Available: https://opendatascience.com/5-preferred-programming-languages-for-web-scraping/. [Kasutatud 02 10 2022].

[5] C. Dilmegani, „Best Web Scraping Programming Languages in 2022 with Stats,“ 31 10 2022. [Võrgumaterjal]. Available: https://research.aimultiple.com/web-scraping-programming-languages/. [Kasutatud 02 10 2022].

[6] „7 Python Libraries For Web Scraping To Master Data Extraction,“ ProjectPro, 30 08 2022. [Võrgumaterjal]. Available: https://www.projectpro.io/article/python-libraries-for-web-scraping/625. [Kasutatud 03 10 2022].

[7] „What is JavaScript used for?,“ Hack Reactor, 26 08 2021. [Võrgumaterjal]. Available: https://www.hackreactor.com/blog/what-is-javascript-used-for. [Kasutatud 05 10 2022].

[8] „Ruby vs. Python: Pros, Cons, and Where to Start,“ Coursera, 11 07 2022. [Võrgumaterjal]. Available: https://www.coursera.org/articles/ruby-vs-python. [Kasutatud 06 10 2022].

[9] P. Carbonnelle, „PYPL PopularitY of Programming Language index,“ PYPL, 31 11 2022. [Võrgumaterjal]. Available: https://pypl.github.io/PYPL.html. [Kasutatud 06 10 2022].

[10] „How do I use the Network view in Dev Tools?,“ FullStory. [Võrgumaterjal]. Available: https://help.fullstory.com/hc/en-us/articles/360020623654-How-do-I-use-the-Network-view-in-Dev-Tools-. [Kasutatud 07 10 2022].

[11] Prisma, [Võrgumaterjal]. Available: https://prismamarket.ee. [Kasutatud 01 07 2022].

[12] G. Romero, „What is Postman API Test,“ Encora, 29 06 2021. [Võrgumaterjal]. Available: https://www.encora.com/insights/what-is-postman-api-test. [Kasutatud 08 10 2022].

[13] „Postman API Platform,“ Postman. [Võrgumaterjal]. Available: https://www.postman.com. [Kasutatud 08 10 2022].

[14] M. Tyson, „What is JSON? The universal data format,“ InfoWorld, 26 08 2022. [Võrgumaterjal]. Available: https://www.infoworld.com/article/3222851/what-is-json-a-better-format-for-data-exchange.html. [Kasutatud 09 10 2022].

[15] K. Reitz, „Requests: HTTP for Humans™,“ Requests. [Võrgumaterjal]. Available: https://requests.readthedocs.io/en/latest/. [Kasutatud 09 10 2022].

[16] BARBORA, [Võrgumaterjal]. Available: https://barbora.ee/. [Kasutatud 15 10 2022].
[17] A. S, „What Is HTML? Hypertext Markup Language Basics Explained,“ Hostinger Tutorials, 15 11 2018. [Võrgumaterjal]. Available: https://www.hostinger.com/tutorials/what-is-html. [Kasutatud 16 10 2022].
[18] M. Maithani, „Scrape Beautifully With Beautiful Soup In Python,“ Analytics India Magazine, 04 12 2020. [Võrgumaterjal]. Available: https://analyticsindiamag.com/beautiful-soup-webscraping-python/. [Kasutatud 17 10 2022].
[19] Selver, [Võrgumaterjal]. Available: https://www.selver.ee/. [Kasutatud 25 10 2022].
[20] K. Sahin, „Web Scraping using Selenium and Python,“ ScrapingBee, 25 08 2022. [Võrgumaterjal]. Available: https://www.scrapingbee.com/blog/selenium-python/. [Kasutatud 30 10 2022].
[21] P. Wójcik, „Why Single-page Applications + REST are better than Dynamic Web pages,“ Applandeo, 08 11 2015. [Võrgumaterjal]. Available: https://applandeo.com/blog/single-page-applications-rest-better-dynamic-web-pages/. [Kasutatud 01 11 2022].
[22] A. Gillis, „REST API (RESTful API),“ SearchAppArchitecture. [Võrgumaterjal]. Available: https://www.techtarget.com/searchapparchitecture/definition/RESTful-API. [Kasutatud 02 11 2022].
[23] C. Dupuis, „SPA vs. MVC,“ Medium, 04 12 2020. [Võrgumaterjal]. Available: https://ctdupuis7.medium.com/spa-vs-mvc-9c9b6706a419. [Kasutatud 04 11 2022].
[24] “REST API,” Seobility. [Võrgumaterjal]. Available: https://www.seobility.net/en/wiki/REST_API. [Kasutatud 04 11 2022].
[25] „Introduction to Spring Boot,“ GeeksforGeeks, Aug. 26, 2019. [Võrgumaterjal]. Available: https://www.geeksforgeeks.org/introduction-to-spring-boot/. [Kasutatud 04 11 2022].
[26] R. Mishal, „6 Advantages and Disadvantages of Spring boot | Limitations & Benefits of Spring boot,“ Hitechwhizz, 26 08 2022. [Võrgumaterjal]. Available: https://www.hitechwhizz.com/2022/08/6-advantages-and-disadvantages-limitations-benefits-of-spring-boot.html. [Kasutatud 05 11 2022].
[27] „Laravel - The PHP Framework For Web Artisans,“ Laravel LLC. [Võrgumaterjal]. Available: https://laravel.com/. [Kasutatud 06 11 2022].
[28] Guru Staff, „The Pros and Cons of Laravel - Guru Blog,“ Guru, 01 12 2021. [Võrgumaterjal]. Available: https://www.guru.com/blog/the-pros-and-cons-of-laravel/. [Kasutatud 06 11 2022].
[29] Eugene, „Vue vs React: What is The Best JavaScript Framework in 2022?,“ Codica, 10 09 2021. [Võrgumaterjal]. Available: https://www.codica.com/blog/react-vs-vue/. [Kasutatud 07 11 2022].
[30] „Framework vs Library: Full Comparison,“ InterviewBit, Oct. 23, 2021. [Võrgumaterjal]. Available: https://www.interviewbit.com/blog/framework-vs-library/. [Kasutatud 08 11 2022].
[31] A. Camus, „JavaScript Library vs JavaScript Frameworks - The Differences,“ Microverse, 05 05 2022. [Võrgumaterjal]. Available: https://www.microverse.org/blog/javascript-library-vs-javascript-frameworks-the-differences. [Kasutatud 09 11 2022].

[32] S. Arancio, „What is JSX?,“ Medium, 05 10 2021. [Võrgumaterjal]. Available: https://medium.com/@sjarancio/what-is-jsx-e3dda0af3490. [Kasutatud 10 11 2022].
[33] „TypeScript Introduction,“ W3schools. [Võrgumaterjal]. Available: https://www.w3schools.com/typescript/typescript_intro.php. [Kasutatud 11 11 2022].
[34] „PHP: MySQL Database,“ W3schools. [Võrgumaterjal]. Available: https://www.w3schools.com/php/php_mysql_intro.asp. [Kasutatud 12 11 2022].
[35] „User-Agent - HTTP | MDN,“ Mozilla Foundation, 23 10 2022. [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent. [Kasutatud 12 11 2022].
[36] IndianKid, “What is a Proxy Server and How Does it Work?,” Huawei Enterprise Support Community, 08 03 2021. [Võrgumaterjal]. Available: https://forum.huawei.com/enterprise/en/what-is-a-proxy-server-and-how-does-it-work/thread/706473-867. [Kasutatud 13 11 2022].
[37] J. Balili, „Implement CRUD with Laravel Service-Repository Pattern,“ DEV, 20 05 2020. [Võrgumaterjal]. Available: https://dev.to/safbalili/implement-crud-with-laravel-service-repository-pattern-1dkl. [Kasutatud 15 11 2022].
[38] S. Mayzes, “How Do You Work in Laravel — 2020 Edition,” Medium, 12 06 2020. [Võrgumaterjal]. Available: https://medium.com/swlh/how-do-you-work-in-laravel-2020-edition-7d07b2102a70. [Kasutatud 19 11 2022].
[39] „Eloquent: Getting Started,“ Laravel LLC. [Võrgumaterjal]. Available: https://laravel.com/docs/9.x/eloquent. [Kasutatud 19 11 2022].
[40] „Authentication vs. Authorization: What’s the Difference?,“ OneLogin. [Võrgumaterjal]. Available: https://www.onelogin.com/learn/authentication-vs-authorization. [Kasutatud 20 11 2022].
[41] S. Carr, „VueJS 3 Composition API and Netlify CD,“ Medium, 01 11 2020. [Võrgumaterjal]. https://simonjcarr.medium.com/vuejs-3-composition-api-and-netlify-cd-ebc21d345142. [Kasutatud 21 11 2022].

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

Mina, Ahmed Abdullajev

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Rakendus hindade jälgimiseks toidupoodides“, mille juhendaja on Aleksei Talisainen
   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.

27.11.2022

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 - Osa Prisma veebikraabitsa koodist

```
import requests
import time
import db
from urllib.parse import urlparse

from enums.Stores import Stores

url = "https://www.prismamarket.ee"
IMAGE_PATH = 'https://s3-eu-west-
1.amazonaws.com/balticsimages/images/180x220/'
IMAGE_FORMAT = '.png'
payload = {}
headers = {
    'Accept': 'application/json',
    'Cookie': '_web_session_id=18c2f386974138265f1ed571cb3d8ef8;
in_session=2; is_new_user=1',
    'User-Agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:52.0)
Gecko/20100101 Firefox/52.0'
}


def get_urls(db):
    myCursor = db.cursor(buffered=True, dictionary=True)

    myCursor.execute("SELECT id, category_id, store_id, url FROM web_pages
WHERE store_id = %s", (Stores.PRISMA.value, ))

    # selectors = myCursor.fetchall()
    pages = []
    for row in myCursor:
        pages.append({'category': row['category_id'], 'store':
row['store_id'], 'web_page': row['id'],
                      'next_url': urlparse(row["url"]).path + '/page/1'})
    return pages



def can_parse_prop(entry, prop):
    try:
        entry[prop]
    except KeyError:
        return False
    return True
```

```
def parse_data(pages, dbcon, version_id):
    for page in pages:
        next_url = page['next_url']
        print(page['category'])
        while True:
            json_response = requests.request("GET", url + next_url,
headers=headers,
                                             data=payload).json()
            print(json_response['pagination']['next_url'])
            for entry in json_response['entries']:
                data = {}
                data['store_id'] = page['store']
                data['category_id'] = page['category']
                data['parse_version_id'] = version_id
                data['unit'] = entry['comp_unit']
                data['title'] = entry['name'] + ', ' + entry['subname']
                if can_parse_prop(entry, 'image_guid'):
                    data['image_url'] = IMAGE_PATH + entry['image_guid'] +
IMAGE_FORMAT
                if can_parse_prop(entry, 'entry_ad'):
                    data['offer_price'] = entry['entry_ad']['price']
                    data['offer_unit_price'] =
entry['entry_ad']['comp_price']
                    data['offer_until_date'] = entry['entry_ad']['ends_at']
                    data['original_price'] = entry['original_price']
                else:
                    data['original_price'] = entry['price']
                    data['unit_price'] = entry['comp_price']

                db.add_product(dbcon, data)
            print('---------------------next page----------------------------
')
            next_url = json_response['pagination']['next_url']
            time.sleep(4)
            if not next_url:
                break
        print('----------next url--------------------')
        time.sleep(5)
```

Lisa 3 – Osa Maxima (Barbora) veebikraabitsa koodist

```
import time
import db
from bs4 import BeautifulSoup
import requests
from decimal import Decimal
from re import sub
from enums.Stores import Stores

base_url = 'https://barbora.ee'

headers = requests.utils.default_headers()
headers.update({
    'User-Agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:52.0)
Gecko/20100101 Firefox/52.0',
})


def get_urls(db):
    myCursor = db.cursor(buffered=True, dictionary=True)

    myCursor.execute("SELECT id, category_id, store_id, url FROM web_pages
WHERE store_id = %s", (Stores.MAXIMA.value,))

    pages = []
    for row in myCursor:
        pages.append({'category': row['category_id'], 'store':
row['store_id'], 'web_page': row['id'],
                      'next_url': row["url"]})
    return pages

def parsing(pages, dbcon, version_id):
    num = 0;
    for page in pages:
        current_page = page['next_url']
        # send headers to prevent being blocked as a bot
        while True:
            page_url = requests.get(current_page, headers=headers)
            soup = BeautifulSoup(page_url.text, 'html.parser')
            brics_list = soup.find_all('div', class_='b-product-wrap-img')
            print(len(brics_list))
            for brick in brics_list:
                num += 1
                data = {}
```

```
                title = brick.find('a', class_='b-product-title b-product-
title--desktop b-link--product-info').find(
                    'span')
                current_price = brick.find('span', class_='b-product-price-
current-number')
                old_price = brick.find('del', class_='b-product-crossed-out-
price')
                unit_price = brick.find('div', class_='b-product-price--
extra')
                if current_price:  # to not add products out of stock
                    data['store_id'] = page['store']
                    data['category_id'] = page['category']
                    data['parse_version_id'] = version_id
                    data['title'] = title.get_text(' ', strip=True)
                    data['image_url'] = brick.a.img['src']
                    if old_price:
                        data['offer_price'] = Decimal(sub(r'[^\d.]', '',
current_price.get_text(' ', strip=True)))
                        data['original_price'] = Decimal(sub(r'[^\d.]', '',
old_price.get_text(' ', strip=True)))
                        data['offer_unit_price'] = Decimal(sub(r'[^\d.]', '',
unit_price.get_text(' ', strip=True)))
                    else:
                        data['original_price'] = Decimal(sub(r'[^\d.]', '',
current_price.get_text(' ', strip=True)))
                        data['unit_price'] = Decimal(sub(r'[^\d.]', '',
unit_price.get_text(' ', strip=True)))
                    data['unit'] = unit_price.get_text(' ',
strip=True).split('/')[1]  # KG/l/g
                    db.add_product(dbcon, data)
            next_page = soup.find('ul', class_='pagination').find_all('a')[-
1]['href']

            time.sleep(2)
            print(current_page)
            print(base_url + next_page)
            print("NUM: ")
            print(num)
            if current_page == base_url + next_page:
                print("No next pages")
                break
            else:
                current_page = base_url + next_page
```

Lisa 4 - Osa Selveri veebikraabitsa koodist

```
from price_parser import Price
from urllib.parse import urlparse
from selenium import webdriver
import time
import db
from selenium.common import NoSuchElementException
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from enums.Stores import Stores


def element_exists(driver, elemList, selector):
    driver.implicitly_wait(0)
    try:
        value = elemList.find_element(By.CSS_SELECTOR, selector)
    except NoSuchElementException:
        return None
    return value


def get_urls(dbcon):
    myCursor = dbcon.cursor(buffered=True, dictionary=True)

    myCursor.execute("SELECT id, category_id, store_id, url FROM web_pages
WHERE store_id = %s", (Stores.SELVER.value,))

    pages = []
    for row in myCursor:
        pages.append({'category': row['category_id'], 'store':
row['store_id'], 'web_page': row['id'],
                      'next_url': row["url"]})
    return pages


def get_and_add_data(driver, page_info, dbcon, version_id):
    # go to bottom of the page
    html = driver.find_element(By.TAG_NAME, 'html')
    html.send_keys(Keys.END)
    time.sleep(2)

    ## Scraping Products
    fullData = []
```

```
    productCards = driver.find_elements(By.CSS_SELECTOR, '.ProductCard')
    for product in range(len(productCards)):
        data = {}

        data['store_id'] = page_info['store']
        data['category_id'] = page_info['category']
        data['parse_version_id'] = version_id
        title = element_exists(driver, productCards[product],
'.ProductCard__title > .ProductCard__link')
        if title:
            data['title'] = title.text
        image_url = element_exists(driver, productCards[product],
                                   '.product-image.ProductCard__image >
img').get_attribute('src')
        if image_url:
            data['image_url'] = image_url
        offer_price = element_exists(driver, productCards[product],
'.ProductPrice.ProductPrice--special')
        if offer_price:
            data['offer_price'] =
Price.fromstring(offer_price.text.split('\n')[0]).amount
            offer_unit_price = element_exists(driver, productCards[product],
                                              '.ProductPrice.ProductPrice--
special > .ProductPrice__unit-price')
            if offer_unit_price:
                data['offer_unit_price'] =
Price.fromstring(offer_unit_price.text).amount
            original_price = element_exists(driver, productCards[product],
'.ProductPrice.ProductPrice--original')
            if original_price:
                data['original_price'] =
Price.fromstring(original_price.text.split('\n')[0]).amount
            unit_price = element_exists(driver, productCards[product],
                                        '.ProductPrice.ProductPrice--original
> .ProductPrice__unit-price')
            if unit_price:
                data['unit_price'] = Price.fromstring(unit_price.text).amount
                data['unit'] = unit_price.text.split('/')[1]
        else:
            original_price = element_exists(driver, productCards[product],
'.ProductPrices > .ProductPrice')
            if original_price:
                data['original_price'] =
Price.fromstring(original_price.text.split('\n')[0]).amount
            unit_price = element_exists(driver, productCards[product],
'.ProductPrice > .ProductPrice__unit-price')
            if unit_price:
                data['unit_price'] = Price.fromstring(unit_price.text).amount
                data['unit'] = unit_price.text.split('/')[1]
        db.add_product(dbcon, data)
```

```
def parse_next_pages(driver, page_count, page_info, dbcon, version_id):
    for page in range(1, int(page_count)):
        if page == 1:
            time.sleep(2)
            get_and_add_data(driver, page_info, dbcon, version_id)
            time.sleep(2)
        next_page_elem = driver.find_element(
            By.CSS_SELECTOR, '.sf-pagination.Pagination > .sf-
pagination__item.sf-pagination__item--next')
        next_page_elem.click()
        time.sleep(3)
        get_and_add_data(driver, page_info, dbcon, version_id)
        time.sleep(2)


def parse(pages, dbcon, version_id):
    opt = webdriver.ChromeOptions()

    opt.add_argument("--disable-xss-auditor")
    opt.add_argument("--disable-web-security")
    opt.add_argument("--allow-running-insecure-content")
    opt.add_argument("--no-sandbox")
    opt.add_argument("--disable-setuid-sandbox")
    opt.add_argument("--disable-webgl")
    opt.add_argument("--disable-popup-blocking")

    s = Service('chromedriver.exe')
    driver = webdriver.Chrome(service=s, options=opt)
    driver.get('https://www.selver.ee/')
    for page_info in pages:
        url = urlparse(page_info.get('next_url')).path
        link = driver.find_element(By.XPATH,

"//ul[@class='SidebarMenu__list']//ul[@class='SidebarMenu__list']//a[contains
(@href, '" + url + "')]")
        driver.execute_script("arguments[0].click();", link)
        time.sleep(4)

        try:
            page_count = driver.find_elements(By.CSS_SELECTOR, '.sf-link.sf-
pagination__item')[-1].text
        except IndexError:
            page_count = 1
            print('1 page only')

        if page_count == 1:  # parse only 1 page
            get_and_add_data(driver, page_info, dbcon, version_id)
        else:
            parse_next_pages(driver, page_count, page_info, dbcon,
version_id)  # parse first and then next pages
        time.sleep(5)
```

Lisa 5 - Veebirakenduse tagakülje struktuur, mis näitab teenusehoidla kujundusmustrit