TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mari-Lotta Moldau 194105IADB

Rakenduse prototüüp valuutakursside võrdlemiseks

Bakalaureusetöö

Juhendaja: Einar Kivisalu
Magister

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: Mari-Lotta Moldau

14.05.2023

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on luua prototüüp veebirakendusest, mis kuvab erinevate valuutavahetuspunktide vahetuskursse. Rakendus on mõeldud täitma auku turul, kus puudub lahendus võrrelda ühel lehel erinevate võimalike virtuaalsete valuutavahetajate kursse selleks, et teha informeeritud otsus, kus ja millal oma raha vahetada.

Arendusprotsessi käigus luuakse veebirakendus, mis võimaldab kuvada vähemalt seitsme virtuaalse valuutavahetuspunkti kursse ning vähemalt ühte neutraalset ehk turu keskmist kurssi. Lisaks peab rakendus märgistama parima ja halvima vahetuskurssi valuuta kohta ning näitama joondiagrammil valuutakursi muutumist. Rakenduse prototüübi skoopi jääb katta valuutavahetust USA dollarist Peruu Solliks ja vastupidi. Tulevikus on võimalik lisada ka teisi valuutasid.

Lõputöö käigus analüüsitakse mitmeid tehnoloogiaid ja meetodeid veebirakenduse kliendi- ja serveripoolse osa loomiseks. Analüüsitud tehnoloogiatest seast valitakse neist sobivaim püstitatud eesmärgi saavutamiseks ning luuakse nende abil rakendus, mis vastab töös püstitatud nõuetele.

Veebirakenduse põhiosa tugineb veebikaabitsal, mis kaabib andmeid virtuaalsete valuutavahetuspunktide veebilehtedelt ning võimaldab kuvada loodud veebilehel aktuaalset informatsiooni valuutakursside kohta.

Rakenduse toimimise kinnitamiseks testitakse selle funktsioone. Lisaks viiakse läbi kasutajakogemuse uuring, et hinnata üleüldist kasutajakogemust.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 41 leheküljel, 8 peatükki, 17 joonist, 5 tabelit.

Abstract

An Application Prototype for Comparing Currency Exchange Rates

The aim of the current thesis is to create a prototype of a web application, that gathers information from virtual currency exchange institutions about their exchange rates. Said application is intended to fill the void on the market, where there is a lack of a solution to be able to compare exchange rates of different changing institutions on one page in order to make an informed decision about where and when to exchange money.

During development, an application is created that visualizes the exchange rates of at least seven virtual currency exchange companies and one neutral or mid-market rate. In addition, the application is intended to show the best and worst exchange rates per currency and showing the currency exchange rate changes on a line chart per provider. In the scope of the prototype, the application must only show exchange rates from USD to Peruvian Sol and vice versa. Other currencies may be added in the future.

In the process, many possible technologies and methods for developing the client- and server-sides of the application are analysed. The best-suited technologies for accomplishing the aim of the application are chosen to be used in the development process to create an application that fits the established requirements.

The main part of the application is based on a web scraper that collects data from virtual currency exchange websites and allows visualization of the current data on the website.

The proper functioning of the application is ensured by testing the application's functions. In addition, a user experience study is conducted to evaluate the overall user experience.

The thesis is in Estonian and contains 41 pages of text, 8 chapters, 17 figures, 7 tables.

Lühendite ja mõistete sõnastik

ACID	Transaktsioonid andmebaasi terviklikkuse tagamiseks: *Atomicity* (Täielikkus), *Consistency* (Järjepidevus), *Isolation* (Isoleeritus) ja *Durability* (Püsivus)
API	*Application Programming Interface* – rakendusliides (protokollide ja tööriistade komplekt rakendustarkvara ehitamiseks)
ASGI	*Asynchronous Server Gateway Interface* - spetsifikatsioon, mis kirjeldab, kuidas Pythoni asünkroonseid veebirakendusi saab suhelda veebiserveritega
back-end	Serveripoolse tarkvara loomise ja arendamise osa veebiarenduse valdkonnas
Clicjacking	ründetehnika, kus ründaja petab kasutajat klõpsama varjatud nuppude, linkide või objektide peal, mis ei ole seotud tegeliku sisuga
Cross-site request forgery	Ründetehnika, kus ründaja sunnib ohvrit tegema soovimatu toimingu veebisaidil, milleks ohver on autenditud, kasutades varem kogutud autentimisandmeid
Cross-site scripting	Veebiturvalisuse haavatavus, mis võimaldab ründajal süstida pahatahtlikku koodi veebilehele ja võtta kontroll üle ohvri veebibrauseri sessiooni üle
CSS	*Cascading Style Sheets* - veebilehtede küljendamisel kasutatav märgistuskeel
DMBS	*Database Management System* - andmebaasi haldussüsteem
DOM	*Document Object Model* – dokumendi objektimudel on platvormist ja keelest sõltumatu XML, XHTML ja HTML dokumentidega suhtlemise liides
DRY printsiip	*Don't Repeat Yourself* - tarkvaraarenduse põhimõte, mis soovitab vältida koodi kordamist, et tõhusalt hooldatavamat koodi luua.
FaaS	*Function-as-a*-Service ehk funktsioon-teenus on serverivaba arhitektuuri mudel, kus arendaja saab oma rakenduse jaoks kirjutada väikeseid, spetsialiseeritud funktsioone ning seejärel käivitada neid funktsioone dünaamiliselt, vastavalt vajadusele.

Foreign Key	Võõrvõti, andmebaasi mõiste, mis kirjeldab seost kahe olemi vahel
front-end	Kasutajaliidese loomise ja arendamise osa veebiarenduse valdkonnas
full-stack	Terviklik tarkvara süsteem, kus on olemas nii rakenduse, kui kasutaja poolne osa.
HTML	*Hypertext Markup Language* - Veebilehe märguandmise keel, programmeerimis- ja märgistuskeel, mida kasutatakse veebilehtede loomiseks ja vormindamiseks
HTTP	*Hypertext Transfer Protocol* ehk Hüperteksti edastusprotokoll
I/O teade	*Input/Output message* - teade, mis annab kasutajale või arendajale teavet programmi käivitamise ajal
IDE	*Integrated Development Environment* – tarkvarapakett, mis sisaldab integreeritud tööriistu tarkvaraarendajatele
IEEE	*Institute of Electrical and Electronics Engineers* - tehniliste erialade kõige suurem rahvusvaheline organisatsioon
IEEE Spectrum	IEEE poolt välja antav tehniline ajakiri
JavaScript	dünaamiline skriptimiskeel, mis võimaldab luua interaktiivseid veebilehti ja veebirakendusi
JetBrains	Tarkvaraarendusettevõte
LAMP *stack*	(Linux, Apache, MySQL, PHP) s*tack* ehk avatud lähtekoodiga tarkvarakomplekt, mida kasutatakse laialdaselt veebirakenduste arendamisel ja käitamisel.
Microsoft Internet Explorer	Veebibrauser, mille oli valmistanud USA firma Microsoft
MVC	*Model-View-Controller* – tarkvaraarenduses arhitektuurimuster, mida kasutatakse kasutajaliidese loomisel
MVT	*Model-View-Template* - rakendusraamistike disainimuster
NoSQL	*Not Only SQL* – andmebaastehnoloogia, mis toetab erinevaid andmebaasi mudeleid.
ODSC	*Open DataSience Conference*
OLTP	*Online Transaction Processing* ehk online tehingute töötlemine on andmebaasi töötlemise meetod
OOP	*Object-Oriented Programming* – objektorienteeritud programmeerimine
ORM	*Object-Relational Mapping* - tarkvara tehnoloogia, mis võimaldab seostada OOP keeles kirjutatud andmemudeli andmebaasi tabelitega.
PEN	Peruu soll

RDMBS	*Relational Database Management System* - relatsioonilise andmebaasi haldussüsteem
REST	*Representational state transfer* - tarkvaraarhitektuuri laad, mis seab veebirakenduse loomisele kindlad piirid
Serverless	Serverivaba arhitektuur on tarkvaraarenduse mudel, kus arendaja ei pea muretsema serverite, virtuaalmasinate või infrastruktuuri haldamise üle.
SPA	*Single Page App* ehk ühe lehe rakendus
SQL	*Structured Query Language*,- standardiseeritud programmeerimiskeel, mida kasutatakse relatsiooniandmebaaside haldamiseks ja neis olevate andmetega erinevate toimingute tegemiseks
SQL injection	tarkvarahaavatavus, mis tekib, kui pahatahtlik kasutaja sisestab veebivormi või URL-i kaudu pahavara SQL-käsu
Stack Overflow	Programmeerimisega seotud küsimuste ja vastuste veebileht
TIOBE indeks	Programmeerimiskeelte populaarsuse pingerea süsteem.
URL	*Uniform Resource Locator* – veebilehe või mõne muu ressursi unikaalne aadress
URL-routing	Veebirakenduse marsruutimise protsess, mis võimaldab kontrollida, millised lehed või andmed kasutajale kuvatakse.
USA	*United States of America* ehk Ameerika Ühendriigid
USD	USA dollar
VBScript	Microsofti poolt loodud skriptimiskeel
web scraping	Veebi kaapimine - tehnoloogia, mille abil veebist andmeid korjata.
WebDriveri protokoll	Protokoll veebibrauserite automatiseeritud testimiseks
VisualBasic	Microsofti integreeritud arenduskeskkond
WSGI	*Web Server Gateway Interface* - spetsifikatsioon, mis kirjeldab, kuidas veebirakenduste tarkvara võib suhelda veebiserveritega Pythoni keeles
XML	*Extensible Markup Language* ehk laiendatav märgendkeel
XPATH	Keel, mida kasutatakse XML-dokumentide elementide ja atribuutide asukoha määratlemiseks

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tänapäeva globaliseeruvas maailmas on rahvusvaheline kaubandus ja reisimine muutunud üha tavalisemaks ning sageli kaasneb sellega vajadus valuuta vahetamise järele. Siiski on erinevate valuutavahetuspunktide kursid sageli erinevad ning neid võib olla raske võrrelda.

Valuuta vahetamiseks on mitmeid võimalusi: võtta välja sularaha kohalikus valuutas, vahetada sularaha füüsilises rahavahetuspunktis või teha seda kõike arvuti tagant ning vahetada raha virtuaalse valuutavahetuspunkti kaudu. Siiski on erinevate valuutavahetuspunktide kursid sageli erinevad ning neid võib olla raske võrrelda. Seetõttu on oluline, et oleks olemas rakendus, mis kogub ja võrdleb erinevate valuutavahetuspunktide kursse, et aidata kasutajatel leida parimaid kursse ning säästa raha.

Käesoleva lõputöö raames luuakse rakenduse prototüüp, mis toob võimalikult paljude virtuaalsete valuutavahetuspunktide kursid ühele lehele. Rakenduse loomisel on peamine eesmärk luua lihtne ja kasutajasõbralik platvorm, mis võimaldab kasutajatel võrrelda erinevate valuutavahetuspunktide kursse ning valida kõige soodsam.

Lisaks keskendub käesolev lõputöö rakenduse tehnilisele lahendusele ja arhitektuurile, et tagada jätkusuutlik ja laiendatav lahendus, mida oleks võimalik ka tulevikus täiendada. Selleks analüüsitakse erinevaid tehnilisi lahendusi ning valitakse nende seast sobivaimad antud ülesande püstituse jaoks.

Antud lõputöö loodab anda väärtusliku panuse valuutavahetuse valdkonnas, aidates kasutajatel leida parima valuutavahetuse kursi. Lisaks võib see olla kasulik tulevastele uurimustele, mis võivad aidata täiustada valuutavahetusplatvorme ja -lahendusi.

1.1 Probleemi ülevaade

Peruus on mitu veebipõhise valuutavahetuse võimalusega valuutavahetuspunkti. Nendest populaarseimad on DollarHouse, Rextie, Segurex, Tkambio, Kambista, Inkamoney ja TuCambista. Igal ühel neist on oma veebilehekülg.

Valuuta vahetamisel tuleb jälgida vahetuskurssi, mis võib igapäevaselt kõikuda. Valuuta vahetamine madala kursiga võib kaasa tuua rahalisi kaotusi, mis omakorda võib mõjutada inimese finantsseisu.

Ainuüksi populaarseimate valuutavahetuspunktide veebilehekülgede läbivaatamine ning hindade võrdlemine on ääretult ajamahukas. Seetõttu valitakse tihti tehingu läbiviimiseks vahetuspunkt, millega on varasem kogemus. Selline lähenemine ei pruugi olla alati kõige parem, sest valuutakurss varieerub ja iga vahetuspunkt otsustab, millise kursiga valuutat müüb. Valides alati sama valuutavahetuspunkti ilma hindade võrdlemiseta võib juhtuda, et tehingut tehes kaotatakse suurem summa raha võrreldes aktuaalse vahetuskursiga.

Selleks, et hoida kokku aega ning raha, on turul puudu lehekülg, mis aitab kliendil leida kõige parema kursiga vahetuspunkt, mille abil valuutat vahetada.

1.2 Metoodika

Antud lõputöö raames analüüsitakse olemasolevaid lahendusi, et luua parim võimalik lahendus probleemile, mis jääks lõputöö raames käsitletavasse ulatusse, ning võimaldaks ka edasiarendamist ning lisafunktsioonide lisamist.

Analüüsi vältel tuuakse välja rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded. Lisaks analüüsitakse mitmeid tehnilisi lahendusi, ning põhjendatakse nende eelistusi ja valikut.

Arhitektuuri loomine toimub vastavalt püstitatud nõuetele. Lisaks luuakse rakendusele ka kasutajaliides.

Kõik lõputöös kasutatud refereerimata tabelid ja joonised on autori poolt loodud.

2 Olemasolevad lahendused

2.1 Best Exchange Rates

Best Exchange Rates (BER) on loodud 2009. aastal. Selle eesmärk on aidata väike-firmadel valuutavahetusega seotult kuludelt kokku hoida, kuvades neile erinevate vahetuspunktide kursse. [1]

BER veebilehel saab valida valuuta, mille vahel vahetada, vahetusviisi ning eesmärk. Valiku põhjal näitab leht erinevaid valuutavahetuspunkte. Lisaks on võimalik lisada ennast kursi jälgijaks, mis võimaldab hoida silm peal valuutakurssidel, et kasutada ära parimaid hetki valuuta vahetamiseks.

2.2 Wize – compare exchange rates.

Wize on 2011 aastal sündinud Eesti ettevõte, mis tegeleb erinevate valuutade haldamisega sama konto alt. Wize on loonud ka valuuta võrdluse rakenduse, kus on võimalik võrrelda Wize-i enda vahetuskurssi teiste samalaadsete valuutavahetajatega. [2]

Wize-i veebirakendus *Compare Exchange rates*, võrdleb valuuta kursse, vaid nende valuutade vahel mille vahel vahetamist võimaldab Wize-i konto. Neid on kokku 21 ning nende hulgas on ka USD ja PEN, kuid valuuta vahetust ning kursi võrdlust võimaldab Wize platvorm vaid ühes suunas.

2.3 Loodava lahenduse skoop

Olemasolevad rakendused näitavad küll üle maailma erinevate valuutavahetuspunktide kursse, kuid ei ole võimalik näha lokaalseid ehk kohalike riigi piiridesse jäävate valuutavahetuspunktide kursse. Seetõttu on lehed peamiselt suunatud ettevõtetele või isikutele, kes tegelevad tehingutega, mis nõuavad igapäevaselt erinevate valuutadega arveldamist ning maksete tegemist üle riigi piiride.

Lahendust, mis võrdleks vaid ühe riigi piires valuuta vahetuspunktide kursse ei olegi olemas, mistõttu on ainus võimalus leida endale sobiv valuutakurss valuutavahetajate lehti järjest avades ning kursse käsitsi võrreldes.

Selle lõputöö raames pakub autor välja luua veebirakendus, mis kiirendaks parima valuutakursi leidmise protsessi, ning võimaldaks kasutajal lihtsasti visualiseerida erinevate valuuta vahetajate kurssi. Rakendus on plaanitud keskenduma Peruu valuutavahetuspunktidele. Kindlasti peab rakendus võimaldama kliendil liikuda lihtsalt sobiva valuutavahetuspunkti kodulehele, kust edasi saab kasutaja sooritada soovitud tehingu. Lisaks on oluline kuvada umbkaudne tehingu läbiviimise aeg juhul kui valuutavahetuspunkt seda võimaldab.

Mitmed valuutavahetuspunktid võimaldavad kiiremat tehingusooritusaega, kui raha ülekanne tehakse valitud pangast, kuid antud rakendus piirdub siiski vaid üldise tehingu sooritamise aja kuvamisega. Samuti jääb rakenduse skoobist välja veebikraabitsate monitooring, mis võimaldab veebilehe omanikul saada teateid, juhtudest, kus veebikraabitsa töö katkeb näiteks valuutavahetuspunkti veebilehe aluskoodi muutumisega.

Arvestades, et valdav enamus virtuaalseid valuutavahetuspunkte töötavad vaid brauseri-põhistena, otsustas autor ka luua rakenduse vaid brauseripõhisena, sest see ei nõua oodatud põhikasutajatelt suurt keskkonna muutust ning võimaldab tehinguid sama lihtsalt sooritada. Samas peab lisaks arvestama, mobiilseadmed on tänapäeval väga populaarsed ning enamik inimesi sooviks lahendust kasutada ka mobiili teel. Seetõttu on oluline luua ka mobiilne lahendus, kuid käesolev lõputöö keskendub siiski ainult brauseripõhise lahenduse loomisele ning ei hõlma mobiilirakenduse arendamist.

Antud rakenduse eesmärgiks ei ole tulu teenimine. Rakenduse põhieesmärk on lihtsustada valuutavahetamise protsessi ning aidata kasutajatel säästa aega ning raha, võimaldades kuvada ühel lehel erinevate veebipõhiste valuutavahetuspunktide vahetuskursse.

3 Veebirakenduse analüüs

Veebirakenduse arendamiseks on eelnevalt vaja paika panna eesmärgid ning nõuded, mida rakendus täitma peab. Lisaks tuleb analüüsida võimalikke viise rakenduse elluviimiseks ja leida parim lahendus soovitud tulemuse saavutamiseks.

Veebirakenduse arendus koosneb kahest suuremast osast: teenusepoolne (*back-end)* ja kliendipoolne (*front-end*) arendus. [3] Teenusepoolne osa vastutab äriloogika, andmebaasiga suhtluse eest. Kliendipoolne osa vastutab kasutajakogemuse eest.

3.1 Nõuete määramine

Loodava lahenduse skoop on aluseks nõuete määramisele. Lisaks võetakse arvesse ka olemasolevate rakenduste vead ja voorused.

Rakenduse põhieesmärk on kuvada klientidele informatsioon, mille aluseks on teised veebilehed ning lihtsustada nendele veebilehtedele liikumist, mistõttu ei ole vajalik luua kasutajasüsteemi.

Rakendusel on kaks kasutaja tüüpi: tavakasutaja ja administraator. Tavakasutajaks loetakse rakenduse kasutajat ning administraatoriks loetakse rakenduse haldajat.

3.1.1 Funktsionaalsed nõuded

Veebilehe kliendipõhised funktsionaalsed nõuded tulenevad vajadusest leida parim vahetuskurss kiirelt ning mugavalt. Funktsionaalseid nõudeid on kliendi poolel kuus:

- Leht peab kuvama neutraalset kurssi USD ja PEN vahel.
- Leht peab visualiseerima vähemalt seitsme virtuaalse valuutavahetuspunkti kurssi.
- Leht peab visualiseerima USD nii müügi- kui ostukurssi.
- Iga valuutavahetuspunkti juures peab olema võimalik liikuda nupuvajutusega antud valuutavahetuspunkti kodulehele.

- Iga valuutavahetuspunkti juures peab olema informatsioon, kui kaua võtab aega vahetatud valuutas raha jõudmine kliendini.
- Kasutajal peab olema võimalus soovituste jagamiseks administraatoriga.
- Igal valutavaatel peab olema visualiseeritud joondiagrammi abil vähemalt viimase nädala valuutakursside muutused valuutavahetuspunktide kaupa.

3.1.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded kirjeldavad rakenduse kvaliteeti ning kehtestavad sellele piirangud ning nõuded. [4]

Mittefunktsionaalsed nõuded kasutaja perspektiivist on järgmised:

- Tavakasutajana soovin aru saada veebilehe sisust kui räägin inglise keelt.
- Tavakasutajana soovin, et lehe laadimine ei võtaks aega kauem kui kolm sekundit.
- Tavakasutajana soovin veebilehte sirvida ükskõik millise levinud brauseriga.
- Tavakasutajana soovin, et veebilehe kasutamine oleks intuitiivne.
- Tavakasutajana soovin, et veebileht oleks lihtsasti navigeeritav.

Mittefunktsionaalsed nõuded veebilehe administraatori perspektiivist:

- Administraatorina soovin, et koodibaas oleks mugavalt hallatav ning jaotatud funktsionaalsuse põhjal osadeks.
- Administraatorina soovin, et koodibaas oleks rahvusvahelises keeles, juhuks kui arendusse on vaja kaasata rohkem inimesi.
- Administraatorina soovin, et koodibaas oleks arusaadav ning dokumenteeritud.
- Administraatorina soovin, et koodi testitaks regulaarselt ning veendutaks, et see vastab kõikidele funktsionaalsetele ja mittefunktsionaalsetele nõuetele.

3.1.3 Tulevikus loodavad funktsionaalsused

Selles peatükis kirjeldatavad funktsioonid jääval selle lõputöö skoobist välja, kuid näitavad kuidas on võimalik edaspidi rakendust edasi arendada ja parendada.

Kindlasti on vaja lisada rakendusele veebikraabitsa monitooring, mis võimaldab administraatoril teada saada, kui veebikraabits ei suuda enam veebilehelt informatsiooni lugeda. See informatsioon on oluline, et rakenduse tagasi töökorda saamine oleks kiire ning veebilehel olev informatsioonipuudus ei tekitaks kasutajatele pikalt ebamugavusi.

Rakendusele on võimalik lisada teiste valuutade vahel vahetamise võimalus. Selle lõputöö raames luuakse tugi vaid USD ja PEN vahel.

Kolmandaks lisafunktsiooniks on lisada kuvatavatele graafikutele interaktiivsus, mis võimaldab kasutajatel täpsemalt võrrelda, kuidas on valuutakurss aja jooksu valuutavahetuspunktis muutunud.

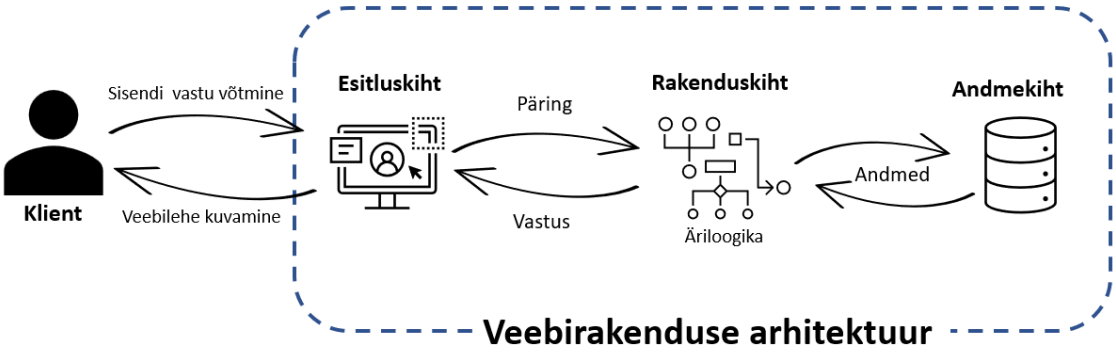
3.2 Arhitektuur

Veebirakenduse arhitektuur koosneb kolmest põhikomponendist: Esitlus kiht, rakenduskiht ning andmekiht (Joonis 1).

Esitluskiht (*Presentation layer*) on osa rakendusest, mis on kasutajale nähtav. See osa rakendusest vastutab informatsiooni kuvamise ja kasutaja sisendi vastu võtmise eest. Kiht töötab veebibrauseris.

Rakenduskiht (*Application layer*) on osa rakendusest, mis töötleb päringuid esitluskihilt, teostab äriloogikat ning vastutab andmebaasiga suhtluse eest.

Andmekiht (*Data layer*) on rakenduse osa, mis vastutab andmete lugemise ja salvestamise eest. Selleks võib kasutada nii relatsioonilist, mitterelatsioonilist või kombineeritud andmebaasi.



Joonis 1 Veebirakenduse arhitektuur

3.3 Teenusepoolsete tehnoloogiate valik

Antud lõputöö eesmärk on luua veebirakendus, mis kogub informatsiooni erinevatelt veebilehtedelt. Veebirakendus on serveritarkvara, mida saab kasutada üksnes veebibrauseri kaudu [5]. Informatsiooni kogumiseks kasutatakse veebikaapimise tehnoloogiat.

3.3.1 Veebikaapimine

Veebikaapimine ehk *web scraping* on tehnoloogia, mille abil on võimalik veebis olevad struktureerimata andmeid viia kujule, kus neid on võimalik analüüsida ja andmebaasis või tabelis hoida. Veebikraapimise tehnoloogiaid on mitmeid, nende hulka kuuluvad näiteks tavaline kopeeri-ja-kleebi meetod, regulaaravaldiste sobitamine, HTTP programmeerimine, HTML analüüs, DOM analüüs, veebikaapimise tarkvarad, vertikaalsed koguplatvormid, semantiliste annotatsioonide tuvastamise ning arvutinägemise veebilehe analüüsimise tarkvara. Kopeeri-ja-kleebi meetod võib osutuda väga aeganõudvaks, eriti kui andmeid, mida kraapida on palju. See tõttu on veebikaapimise tarkvara kasutamine kõige lihtsam viis veebikaapimiseks, sest kõik muud meetodid peale kopeeri-ja-kleebi nõuavad suuremal määral tehnilisi teadmisi. [6]

Tänapäeval on olemas tuhandeid veebikaapimise tarkvarasid. Veebikaapimise tarkvara kasutatakse manuaalse kopeerimise ja kleepimise töö hõlbustamiseks. See tarkvara võib proovida automaatselt ära tunda veebilehe andmestruktuure või pakkuda funktsioone, mille abil kaapida veebilehelt andmeid, muuta nende kuju ja hoida saadud andmeid lokaalses andmebaasis.

Veebikaabitsa loomiseks on vaja valida programmeerimiskeel, mis toetaks veebilehtede lugemist ning sealt informatsiooni otsimist. Olulised aspektid veebikaabitsa jaoks programmeerimiskeele valikul on: keele kasutatavus, veebikaapimise teekide olemasolu ning jõudlus.

3.3.2 Programmeerimiskeelte analüüs ja valik

ODSC (*Open DataSience Conference*) 2022. aasta artiklis „*5 Preferred Programming Languages for Web Scraping*“ võrreldakse viite programmeerimiskeelt veebikaapimiseks. Aluseks võetakse lisaks tõhususele ka koodi kirjutamise intuitiivsus, selle hallatavus ja paindlikkus. Lisaks kõigele võetakse arvesse ka programmeerimisekeele populaarsus. Selles artiklis väljatoodud viieks programmeerimiskeeleks on Python, Ruby, JavaScript, C++ ja Java.

- Python – populaarseim programmeerimiskeel veebikaapimiseks. Ning lisaks sellele ka ajakirja IEEE Spectrum 2021 kõige populaarseim programmeerimiskeel. Selle objektorienteeritud programmeerimiskeelega käib kaasa suur arv teeke ning kaasaarvatus moodulid masinõppeks. Lisaks on Python avatud lähtekoodiga.

- Ruby – dünaamiline avatud lähtekoodiga programmeerimiskeel, mis põhineb Perl keele süntaksil, mis muudab Ruby väga kasulikuks veebilehtede analüüsimisel. Ruby teek Nokigiri on väga kasutajasõbralik, ning võimaldab töötada HTML ja XML failidega.

- Javascript – objektorienteeritud programmeerimiskeel, mida kasutatakse peamiselt veebilehtede skriptimiseks. [7] Koos Node.js käivituskeskkonnaga loetakse seda eelistatus programmeerimiskeeleks veebikaapimiseks dünaamilise koodiga veebilehtesid. Koos teiste teekidega näiteks ExpressJS ja Request on võimalik luua veebikaabits, mis suudab koguda informatsiooni nii veebi- kui mobiilirakendustest.

- C++ – üldotstarbeline objektorienteeritud programmeerimiskeel, mis on loodud eesmärgiga muuta programmeerimine nauditavamaks [8]. Tuntud skaleeritavuse ning paindlikkuse poolest. Keel on staatiline, mistõttu ei sobi ta olukordadeks, kus

on vaja dünaamilist keelt. C++ sobib lihtsaks veebikaapimiseks kuid URL-listide genereerimiseks ja muude veebiämblike tegevuste jaoks ei ole ta nii mugav.

- Java – üldotstarbeline, tüübi- ja klassipõhine programmeerimiskeel. TIOBE indeksi järgi on keel 2023. aasta aprilli kuus kolmandal kohal. [9] Java alla kuuluvad mitmed teegid, mida saab kasutada veebikaapimiseks. Näiteks JSoup, HTMLUnit jt. Java ei sobi väga keerukate veebikaapimise projektide jaoks, kuid võimaldab siiski luua võimsaid veebikaapijaid mitmeks otstarbeks.

Tabelis 1 on välja toodud autori hinnang programmeerimikeelte oskusele ning nende õppekeerukusele.

Tabel 1 Teenusepoolsete programmeerimiskeelte keerukus

Programmeerimiskeel	**Autori Pädevus**	**Õppimiskeerukus**
Python	Hea	Madal [10]
Ruby	Nõrk	Madal [11]
JavaScript	Keskmine	Keskmine [10]
C++	Nõrk	Kõrge [12]
Java	Hea	Keskmine [13]

Võttes arvesse töö autori varasemat kogemust analüüsitud programmeerimiskeeltega piirdutakse edaspidi vaid Pythoni, JavaScripti ja Javaga, vastasel juhul võtaks teiste keelte õppimine liiga kaua aega.

Nende kolme keele vahel otsustamiseks tuleb vaadata nende sobivust veebikaabitsa loomiseks. Tabelis 2 on näha viis olulist tegurit veebikaabitsa keele valimisel ning valitud keelte võrdlused nende tegurite põhjal.

Tabel 2 Programmeerimiskeelte sobivus veebikaabitsa loomiseks [14]

Keel	**Kasutaja-sõbralikkus**	**Teekide dokumentatsioon**	**Populaarsus**	**Kiirus**	**Dünaamiliste lehtede kaapimine**
Python	Hea	Hea	Hea	Keskmine	Hea
JavaScript (Node.js)	Keskmine	Hea	Keskmine	Hea	Hea
Java	Hea	Hea	Keskmine	Madal	Keskmine

JavaScripti raamistiku Node.js-i suureks eeliseks on selle kiirus, kuid kasutajasõbralikkus on teistel keeltel tugevam. Java jääb teistele keeltele all veebikaapimise kiiruse ning dünaamiliste lehtede kaapimise koha pealt. Pythoni suureks eeliseks on keele süntaksi lihtsus ja veebikaapimise teekide kasutajasõbralikkus ja dokumenteeritus. Keel võimaldab ka dünaamiliste veebilehtede kaapimist ning lisaks on Python veebikaapimise projektide seas kõige populaarsem.

3.3.3 Raamistiku valik

Python programmeerimiskeelel on palju erinevaid raamistikke ja teeke pea iga tehnilise valdkonna jaoks. Raamistikud automatiseerivad mitmete koodi osade rakendamist ning loovad rakenduse arendamisele struktuuri. Iga raamistikuga tuleb kaasa valik mooduleid, mis lühendavad märgatavalt arendamise aega. Pythonil on kolme tüüpi raamistikke [15]:

- *Full-stack* ehk täistsükli arendus – Sisaldab kõiki veebiarenduse nõudeid, sealhulgas vormide loojad näiteks ORM ja andmebaasi tugi, mallide kavandid, vormide valideerimine jt. See annab projektile kindlama struktuuri, mida on keerulisem muuta.

- Mikro – Need raamistikud nõuavad arendajalt rohkem käsitöö tegemist. Jälgitakse minimalistlikku veebiarenduse stiil, mis tähendab, et struktuuri osas on arendajal rohkem võimalusi, kuid kui projekt laieneb on vaja lisakoodi kirjutamist, et rakenduse kasvule kohanduda. [16]

- Asünkroonne – Sellised raamistikud kaustavad Python teeki *asyncio*, selleks et jooksutada mitmeid protsesse samaaegselt. Need saavad hakkama mitmete samaaegsete ühendustega.

Tabelis 3 on välja toodud raamistike tüüpide võrdlus mitmete oluliste aspektide alusel.

Tabel 3 Python raamistike tüüpide võrdlus [17]

	Mikro raamistik	***Full-stack* raamistik**
Keerukus	Koosneb minimaalsest koodist, mis on vajalik väiksemate rakenduste arendamiseks.	Koosneb kõigist tööriistadest ja pakettidest, mis on vajalikud täisväärtuslike rakenduste väljatöötamiseks.
Komponendid	Vähemalt päringu marsruutimine.	Päringu marsruutimine turvalisus, ORM, MVC, mallimootor, ja vormi valideerimine.
Eelistatus kasutusalad	Väikesed rakendused, enamasti RESTil põhinevad rakendused väikese programmeerimise vajadusega.	Suurte rakenduste arendamisel.
Õppekeerukus	Väikese mahu tõttu lihtsam õppida.	Suurema koodibaasi tõttu keerulisem õppida.
Arenduse kestus	Eelistatud lühikese arendusperioodi jaoks.	Eelistatud pikema arendusperioodi jaoks.
Koodi maht	Vähem kui 5000 koodirida.	Rohkem kui 5000 koodirida.

Antud rakenduse jaoks sobib kõige paremini *full-stack* raamistik, sest võimalusi rakenduse edasi arendamiseks on mitmeid. *Full-stack* raamistiku komponendid tagavad rakenduse turvalisuse ning kõik vajalikud tööriistad ja paketid rakenduse edukaks loomiseks.

JetBrains-i Python programmeerijate 2021. aasta küsitluse andmetel on populaarseimad Python *full-stack* raamistikud Django ja web2py [18].

- Django – JetBrains-i uuringu järgi kõige populaarsem Python *full-stack* raamistik. See on avatud lähtekoodiga, jälgib DRY printsiipi ja kasutab ORM-i andmeid andmebaasi kaardistamiseks. Raamistikuga on kaasas suur hulk teeke, autentimine ning *URL-routing*. Võrreldes teiste raamistikkudega on Django raamistik turvalisem. [15]

- Web2py – avatud lähtekoodiga raamistik. Platvormist sõltumatu, mis tähendab, et suudab joosta kõigil populaarsematel operatsioonisüsteemidel. Lisaks on veebipoolne arendus lihtsustatud omaenda veebi-põhise IDE-ga, millel on oma koodiredaktor, -siluja jms. Raamistik suudab lugeda mitmeid protokolle, omab rolli-põhist juurdepääsukontrolli ning on tugev ka andmeturvalisuse poolest. [15]

Arvestades JetBrainsi küsitlust on Django mitmeid kordi populaarsem raamistik kui web2py. Nimelt on Django kasutajaid 40% ja web2py kasutajaid vaid 3%. Lisaks on Django kiirem, parema jõudlusega ning turvalisem. [19]

3.3.4 Andmebaasi haldussüsteemi valik

Andmebaas on koht arvutis või pilves, kuhu rakenduse andmed kogutakse ja korrastatakse. Andmebaasi haldust veebirakenduses tehakse andmebaasihaldus-süsteemides (DBMS). Andmebaasil on mitmeid eesmärke. Üheks oluliseks neist on andmete loomine, hoidmine, uuendamine ja haldamine.

Selleks, et valida andmebaasi, tuleb kõigepealt analüüsida andmebaasi tüüpe. On olemas relatsioonilised ehk SQL ja mitterelatsioonilised ehk NoSQL andmebaasid.

- Relatsiooniline andmebaas – andmebaas struktureeritud andmete jaoks. Andemete hoidmiseks jälgitakse kindlat skeemi, mis määrab kuidas andmeid

andmebaasi kirjutatakse. Täpsemalt kirjeldatakse andmete struktuuri ja tüüpi. Andmeid hoitakse enamasti tabelites ja ridades, mis on omavahel seotud.

- Mitterelatsiooniline andmebaas – andmebaas struktureerimata andmete jaoks. Kasutaja ei pea skeemi kindlaks määrama, vaid võivad hoida erinevate struktuuridega andmeid, mis ei pea tingimata seotud olema.

Arvestades, et rakenduse andmed on omavahel seotud ning struktureeritud, sobib paremini relatsiooniline andmebaas.

Relatsioonilise andmebaasi haldussüsteeme (RDBMS) on mitmeid. Stack overflow 2022. aasta arendajate küsitluse järgi on populaarseimad andmebaasi haldussüsteemid MySQL, PostgreSQL ja SQLite [20]. Kõik kolm andmebaasi on avatud koodiga ja tasuta.

- MySQL – relatsiooniline andmebaas, mida on lihtne kasutada. Saab hakkama suuremas koguses andmete lugemisega. Päringuid tehes ei ole tõusutundlik. [21] Enim kasutatav andmebaas algajate programmeerijate seas [20].

- PostgreSQL – relatsiooniline andmebaas, mis on funktsioonirikas ning saab hakkama keeruliste päringutega ja suurte andmebaasi mahtudega. Rohkem toetatud andmetüüpe ja indekseid. Saab hakkama suures koguses andmete lugemise ja kirjutamisega. Keel on päringuid tehes tõusutundlik. [21] Enim kasutatav andmebaas professionaalsete arendajat seas [20].

- SQLite – C-keeles kirjutatud andmebaasimootor. See ei ole eraldi rakendus, vaid teek, mida on võimalik projektides kasutada. Ei sobi suure hulga andmete ega kasutajate jaoks.

Lisaks eelmainitud andmebaasidele on veel olemas kaks suuremat andmebaasisüsteemi: MS SQL ja Oracle. Need andmebaasisüsteemid on jäetud võrdlusest välja nende listsenside maksumuse pärast. Kuigi mõlemad pakuvad tasuta versiooni ei ole kumbki neist vabavaraline ning lihtsa veebi rakenduse jaoks ei ole nii keerulist süsteemi ega funktsionaalsusi vaja.

Tabelis 4 on välja toodud kolme kõige populaarsema andmebaasi võrdlus.

Tabel 4 Andmebaasisüsteemide võrdlus [22]

	MySQL	**PostgreSQL**	**SQLite**
Arhitektuur	Klient/Server	Klient/Server	Failipõhine
Serveri OS	FreeBSD, HP-UX, Linux, NetBSD, OpenBSD, OS X, Solaris, Unix, Windows	FreeBSD, Linux, OS X, Solaris, Windows	*Serverless*
Andmebaasi tehingute järjepidevus	ACID	ACID	ACID
Baaskoodi keel	C++	C	C
Toetatud programmeerimis-keeled	ActionScript, Ada, Basic, C, C#, C++, D, Delphi, Forth, Fortran, Haskell, Java, JavaScript, Lisp, Lua, MATLAB, PHP, PL/SQL, Python	C, C++, Delphi, Perl, Java, Lua, .NET, Node.js, Python	NET, C, C++, Delphi, Java, Perl, PHP, Python, Tcl
Populaarsed kasutused	Madala-keskmise liiklusega veebisaidid, Testimine ja arendus	Veebilehed, veebirakendused, LAMP Stack ja OLTP-põhised rakendused	Analüütika, andmete kaevandamine, andmehoidlad, ärianalüüs, Hadoop ning veebi-, mobiil-, georuumilised ja analüütika rakendused
Autori pädevus	Keskmine	Keskmine	Hea

Üleüldiselt on kõige populaarsem avatud lähtekoodiga andmebaasisüsteem MySQL, kuid PostgreSQL andmebaas toetab programmeerimisel valitud Python programmeerimiskeelt. Ainus *serverless* andmebaas on valitutest SQLite, mille arhitektuur põhineb FaaS tehnoloogial, kus kood laetakse pilveplatvormile ja käivitatakse sinult siis kui see on vajalik. Klient/server arhitektuur võimaldab mitme kliendi ühendamist samasse andmebaasi ning selle kasutamist erinevatel platvormidel ja seadmetel. Nii PostgreSQL kui MySQL andmebaase kasutatakse veebirakenduste loomisel. Nende kahe andmebaasi vahel otsustamiseks tuleb arvesse võtta autori varasem kogemus.

3.4 Kliendipoolse tehnoloogia valik

Kliendipoolne osa veebirakendusest on see, mida kuvatakse kliendi seadmes kaasaarvatud kõik, mida klient näeb (pildid, tekst ja üleüldine kasutaja kogemus). Lisaks käib kliendipoolse osa alla ka rakenduse tegevused kasutaja brauseris. [23]

Kliendipoolse osa arendamiseks kasutatakse spetsiifilisi programmeerimiskeeli, milleks on HTML, CSS ja JavaScript. [24]

- HTML ehk (*HyperText Markup Language*) – märgistuskeel, mida kasutatakse veebilehe struktureerimiseks ja selle sisu kuvamiseks. Keel koosneb elementidest, mida kasutatakse sisu ümbritsemiseks, et see veebilehel teatud asukohta paigutada. [25]

- CSS ehk (*Cascading Style Sheets*) – stiililehe keel, mida kasutatakse erinevate HTML elementide stiliseerimiseks. [26]

- JavaScript – programmeerimiskeel, mis lisab veebilehele interaktiivsuse. [27] See on üks kõige tuntumaid skriptimis-keeli veebilehtede loomisel, kuid seda kasutavad ka mitmed brauseri-välised keskkonnad. [28]

Tabelis 5 on võrreldud kliendipoolsete programmeerimikeelte oskust ning nende õpikeerukust.

Tabel 5 Kliendipoolsete programmeerimiskeelte võrdlus

Programmeerimiskeel	Autori pädevus	Õppimiskeerukus
HTML	Hea	Madal [29]
CSS	Hea	Keskmine [30]
JavaScript	Keskmine	Keskmine [31]

Eelnevalt valitud Pythoni raamistik Django kasutab oma mallimootoris HTML malle veebilehtede loomiseks ning seetõttu on HTML programmeerimiskeele kasutamine vältimatu. Lisaks on võimalik Django rakendusele lisada CSS stiilileht ning JavaScript fail veebilehele kujunduse ja interaktiivsuse lisamiseks. Järgnevates peatükkides vaadeldakse lähemalt CSS ja JavaScript failide vajalikkust ning kasutust.

3.4.1 CSS

CSS on programmeerimiskeel, mis määrab kuidas dokumenti kasutajale kuvatakse – milline on tema stiil, paigutus jms. Dokumendiks nimetatakse tavaliselt HTML tekstifaili ning selle kasutajale kuvamine tähendab selle muutmist sellisesse vormi, mis on publikule kasutatav. [32]

Brauserid on loodud selleks, et dokumente visuaalselt kuvada. CSS on loodud väga elementaarseks teksti stiliseerimiseks, näiteks värvi ja suuruse muutmiseks. Lisaks saab sellega muuta ka elementide asukohta ja kohati luua ka animatsioone. [32]

Stiilifaili olemasolu on oluline veebirakenduse kujunduse vaatenurgast. Veebilehe kujundamine on oluline, sest suurendab veebilehe atraktiivsust ning muudab lehe kliendile kasutajasõbralikumaks.

3.4.2 JavaScript

JavaScript on moodsa interaktiivse veebirakenduse tuum. Tuntud JavaScripti raamistikud, mida on võimalik Django veebirakenduses kasutada on React, Angular ja VueJS. Lisaks on võimalik kasutada ka puhast JavaScripti, minimaalse funktsionaalsuse lisamiseks. [33]

Selleks, et otsustada kas ja millist JavaScripti raamistikku kasutada, tuleb vaadata erinevaid Django rakenduse kliendipoolse osa arhitektuure.

Esimeseks on traditsiooniline lähenemine, kus mallid renderdatakse serveri pool. Siin lisatakse JavaScript Django mallidele väikeste osadena. Sellega saab kasutada kõiki Django sisse ehitatud funktsioone. Lähenemine sobib väiksematele tiimidele ning võimaldab kiiret prototüübi loomist. Võimalikuks negatiivseks küljeks on see, et kliendipoolse osa kood võib kasvades olla segane ja organiseerimata. [33]

SPA ehk *single page app* eesliides ja Django REST API-na, kus Django on kasutusel vaid serveripoolses osas ning suhtleb vaid läbi API. Selline lähenemine sobib tiimidele, kus on eraldi eesliidese ja tagapoole arendajad, sest nii saavad tiimis samaaegselt töötada ilma üksteise tööga vastuollu sattumata. Siin lihtsustab JavaScripti raamistiku kasutamine suures osas keeruliste eesliidese rakenduste ehitamist. Negatiivseks küljeks on see, et mitmeid Django sisseehitatud funktsioone ei saa kasutada, näiteks vaateid, malle jne. [33]

Lisaks on võimalik kasutada hübriidi kahest eelnevast arhitektuurist, kus JavaScripti kasutatakse vaid kohtades, kus seda vaja ning ülejäänud kordadel kasutatakse Django malle. See lähenemine võtab mõlemast arhitektuurist parima. Django malle kasutatakse lihtsamate veebilehtede kuvamiseks, näiteks avaleht, ning kliendipoolset JavaScripti kasutatakse lehtedel, kus on vaja luua keerulisemaid kasutajakogemuse elemente. [33]

Kuna antud rakendus on staatiline ning ei nõua kliendilt palju lehega suhtlemist sobib kasutamiseks klassikaline lähenemine, kus JavaScript on staatiline ning genereeritud serveri pool.

3.5 Arenduskeskkonna valik

Arenduskeskkonna valik tuleb jagada kaheks osaks: koodihaldus- ja koodiarendus keskkond. Koodi halduskeskkonda kasutatakse koodi ja muu projektiga seonduvate dokumentide, näiteks dokumentatsiooni, testide ja skriptide hoidmiseks. [34] Koodi arenduskeskkond on kollektsioon erinevatest protsessidest ja vahenditest, mida kasutatakse programmi või tarkvaratoote arendamiseks. [35]

Kaks kõige populaarsemat koodi halduskeskkonda Pythoni arendajate seas on GitHub ja GitLab. [18]

GitHub on neist keskkondadest suurim. Seda kasutab üle 100 miljoni arendaja ja selles on hoiul üle 330 miljoni hoidla ehk repositooriumi. [36] Autor omab rahuldavat varasemat kogemust.

GitLab on sisult GitHubiga üsna sarnane. 2023. aasa alguses jõudis ka GitLab 100 tuhande kasutajani. Autor omab head varasemat kogemust.

Koodi arenduskeskkonna valikul tuleb arvestada programmeerimiskeele toega. Python keelt toetavad populaarseimad integreeritud arenduskeskkonnad (IDE) on IDLE, PyCharm ja Visual Studio Code. [37]

IDLE on tasuta arenduskeskkond, mida on võimalik kasutada MacOSi, Windowsi ja Linuxi peal. Seda on lihtne kasutada ja sobib hästi algajale programmeerijale. Parimateks funktsioonideks on mitme faili otsing, vea- ja i/o teated, põhilised tekstiredaktori funktsioonid ja võimas koodisilur. [37] Autoril puudub varasem kogemus.

PyCharm on laialdaselt kasutatud JetBrainsi poolt loodud arenduskeskkond. Selle funktsioonide hulka kuuluvad JavaScripti, CSSi ja TypeScripti tugi, tark koodinavigatsioon, kiire ja turvaline koodiredaktsioon ning lisad, nagu läbi keskkonna andmebaasile ligi pääsemine. [37] Autoril PyCharmi kasutamisel hea varasem kogemus.

Visual Studio Code on Microsofti poolt loodud avatud koodiga arenduskeskkond. Selle eelisteks on parim tark koodi lõpetamine, hea koodihalduskeskkonnaga ühendumine. Lisaks on võimalik alla laadida lisafunktsioone näiteks teemasid või muid teenuseid. [37] Autoril on rahuldav varasem kogemus.

Arvestades loodava rakenduse vajadusi ning autori varasemat kogemust valitakse koodihaldus keskkonnaks GitLab ja arendus keskkonnaks PyCharm.

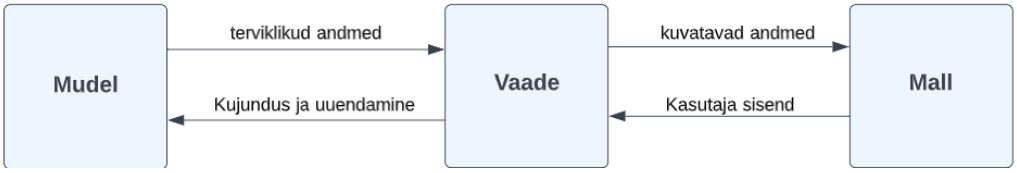
4 Veebiteenusepoolne lahendus

Rakenduse veebiteenusepoolne osa ehk *back-end* on ehitatud Django raamistiku peale. Django on Pythoni raamistik, ning on loodud veebirakenduste loomiseks. Django jälgib MTV arhitektuuri, mis erineb MVC arhitektuurist selle poolest, et Django hoolitseb ise kontrolleri eest. [38]

4.1 Django rakenduse arhitektuur

MVT ehk *Model-view-Template* struktuuril on kolm osa (Joonis 2):

- Mudel ehk *Model* – liides, mis vastutab kogu andmetega seonduva informatsiooni eest. See hoiab kõiki andmete hoidmisega seotud vajalikke välju ja käitumismustreid. Enamasti vastab iga mudel ühele tabelile andmebaasis. [39]
- Vaade ehk *View* – kasutajaliides, mis koosneb Pythoni programmist, mis võtab vastu veebipäringuid ja saadab vastu veebivastuse. [40]
- Mallid ehk *Templates* – staatiline osa soovitud HTML väljundist ja dünaamiline osa, mis kirjeldab kuidas dünaamiline sisu lehele sisestatakse. [41]



Joonis 2 Django struktuuri skeem

Kui luuakse uus projekt siis luuakse Django juurkataloogi fail maganage.py. See fail hoolitseb käsurea funktsioonide eest. Nagu näiteks *runserver*, mis käivitab Django programmi, *makemigrations*, mis loob andmebaasi migratsiooni skeemi ja *migrate*, mis teeb migratsioonifaili põhjal andmebaasi skeemis muudatusi. [42]

Lisaks luuakse juurkataloogi projekti kaust, mille sisse luuakse projekti failid:

- __init__.py – tühi fail, mille olemasolu annab Pythoni interpreteerijale teada, et tegemist on Pythoni projektiga. [42]

- settings.py – fail, mis sisaldab Django projekti seadeid. Seda faili loetakse kõige olulisemaks ning seda kasutatakse rakenduste ja vahevara rakenduste lisamiseks. Siin see on ka andmebaasi konfiguratsioon (algselt SQLite3 andmebaas, mille saab asendada mistahes teise andmebaasiga). [42]

- urls.py – universaalne ressursiotsija, mis sisaldab kõiki veebilehe jaoks vaja minevaid lõpp-punkte. [42]

- wsgi.py – WSGI ehk *Web Server Gateway Interface* kirjeldab, kuidas suhtlevad serverid rakendustega. [42]

- asgi.py - ASGI ehk *Asynchronous Server Gateway Interface.* Väga sarnane WSGI-ga, kuid omab mõningaid lisafunktsionaalsusi. [42]

Luues Django projekti, tuleb sinna sisse luua ka rakendus. Rakendusi võib ühes projektis olla üks või rohkem, mis suudavad töötada samaaegselt. Rakenduse sisse luuakse failid:

- __init__.py – fail, millel on sama eesmärk, mis projekti sees oleval samanimelisel failil. Selle olemasolu tähendab et rakendus on pakett. [42]

- admin.py – fail, mida kasutatakse Django mudelite registreerimiseks Django administratsiooni paneeli. Sellel on kolm eesmärki: mudelite registreerimine, superkasutaja loomine ja sisselogimine ning veebirakenduse kasutamine. [42]

- apps.py – fail, mille eesmärgiks on aidata lisada rakenduse seaded oma rakenduse jaoks. [42]

- models.py – kõige olulisem fail, siin kirjeldatakse veebirakenduse mudelid klassi vormis. Mudelid defineerivad andmebaasi struktuuri. [42]

- views.py – fail, milles on kirjeldatud vaated, mille abil rakenduse kasutaja rakendusega suhtleb. Vaated on failis klasside vormis [42]

- tests.py –fail rakenduse testide loomiseks. [42]

4.2 Veebikaabits

Suur osa loodava rakenduse äriloogikast põhineb veebikaabitsal. Veebikaapimine on tehnika, mille abil kogutakse automaatselt veebilehtedelt andmeid. Loodavas rakenduses kasutatakse kahte veebikaabitsa teeki: Selenium ja BeautifulSoup.

4.2.1 BeautifulSoup

BeautifulSoup on Pythoni moodul, mis võimaldab hankida andmeid HTML-ja XML failidest. See töötab koos parteriga, et pakkuda keeleliselt sobivaid viise DOM-puu navigeerimiseks, otsimiseks ja muutmiseks. [43]

BeautifulSoup on kiire, algajasõbralik ning töötab brauserist sõltumatult. Teegi suureks miinuseks on see, et see ei suuda veebilehega suhelda. BeautifulSoup suudab ainult andmeid parsida. Seetõttu on vaja andmete eraldamiseks paigaldada muid mooduleid, nagu näiteks requests või httpx. Selleks, et kraapida JavaScripti poolt renderdatud veebilehti on vaja lisamoodulit, sest BeautifulSoup võimaldab ainult HTML- või XML-failide navigeerimist. [44]

App.DollarHouse.pe on staatiline HTML lehekülg. Joonisel 3 on kujutatud staatilist abimeetodit BeautifulSoup instantsi loomisest, mida hiljem kasutatakse meetodis mis tegeleb veebilehelt andmete kaapimisega.

```
@staticmethod
def get_soup(provider):
    url = getattr(provider, 'website')

    # retrieves  HTML content
    response = requests.get(url)

    #Parsea the HTML content and create a BeautifulSoup object
    soup = BeautifulSoup(response.content, "html.parser")
    return soup
```

Joonis 3 Abimeetod BeautifulSoup instantsi loomisest

Joonisel 4 on näha meetod DollarHouse veebilehe kaabitsast, kus on kasutatus BeautifulSoup moodulit.

```
def scrape_dollarhouse(self):
    provider = next(
        (p for p in self.provider_list if p.name == "DollarHouse")
        , None)
    currency_usd = next(
        (c for c in self.currency_list if c.code == 'USD'), None)
    currency_pen = next(
        (c for c in self.currency_list if c.code == 'PEN'), None)

    soup = self.get_soup(provider)

    #  find the specific HTML element
    exchange_rate_container = soup
                              .body
                              .main
                              .find('div', class_="exchange-rate-container")

    #  extracts the buy and sell prices from the HTML element
    buy_price = exchange_rate_container
                              .find(id="buy-exchange-rate")
                              .text
                              .replace(',', '.')
    sell_price = str(
                round(1 / Decimal(
                              exchange_rate_container
                              .find(id="sell-exchange-rate")
                              .text
                              .replace(',', '.')
                              ), 4)
                          )

    self.save_to_db(provider, currency_usd, currency_pen, buy_price)
    self.save_to_db(provider, currency_pen, currency_usd, sell_price)
```

Joonis 4 Veebikaapimine BeautifulSoupiga

4.2.2 Selenium

Selenium on avatud lähtekoodiga brauseri automatiseerimise tööriist, mida sageli kasutatakse veebi kraapimiseks. See on olnud kasutusel juba üle aastakümne ning selle peamised komponendid on Selenium IDE (kasutatakse toimingute salvestamiseks enne nende automatiseerimist), Selenium WebDriver (käskude käivitamiseks brauseris) ja Selenium Grid (paralleelseks täitmiseks). [44] Antud lõputöö raames on kõige olulisem WebDriver komponent, mida kasutatakse veebikaapimiseks.

Selenium suudab töödelda ka dünaamilisi veebilehti, mis on BeautifulSoupi abil raskesti kraabitavad. Erinevalt BeautifulSoupi raamistikust toetab Selenium ka dünaamilistelt veebilehtedelt andmete kogumist, sest omab veebibrauseris nupuvajutuse, vormi täitmise

ja ühelt lehelt teisele navigeerimise funktsiooni. Lisaks on Seleniumi ka ootamise funktsioon, mis ootab kuni JavaScripti poolt loodav dünaamiline sisu lehele ilmub.

Virtuaalse valuutavahetuspunkti veebilehekülg TuCambista on dünaamiline, mistõttu ei ole võimalik seda ainult BeautifulSoup mooduliga kraapida. Joonisel 5 on näha abimeetodit, mis kasutab veebilehe täielikuks laadimiseks Seleniumi komponenti WebDriver ning ootab lehe laadimisega nii kaua kuni JavaScript osa on veebilehele laetud.

```
@staticmethod
def get_soup_selenium(provider, xpath):
    options = webdriver.ChromeOptions()
    options.add_argument('--ignore-certificate-errors')
    options.add_argument('--incognito')
    options.add_argument('--headless')

    # open the Selenium web drive
    driver = webdriver.Chrome(options=options)
    try:
        # navigate to website
        driver.get(getattr(provider, 'website'))

        # wait for page to load
        WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.XPATH, xpath)))

        # get page source after JavaScript has rendered
        html = driver.page_source

        # parse HTML with Beautiful Soup
        soup = BeautifulSoup(html, 'html.parser')
        pass
    finally:

        # close the Selenium web driver
        driver.quit()

    return soup
```

Joonis 5 Seleniumi kasutamine lehe laadimisel.

Kui ka JavaScripti poolt renderdatud lehe osa on veebilehel näha, saab lehelt informatsiooni otsima hakata. Joonisel 6 on näha veebikaabitsat, mis saadab Seleniumi abil BeautifulSoup objekti loojale XPATH muutuja. See näitab, millise elemendi ilmumist peab WebDriver ootama enne lehe koodi salvestamist.

```
def scrape_tucambista(self):
    provider = next(
        (p for p in self.provider_list if p.name == "Tucambista"), None)
    currency_usd = next(
        (c for c in self.currency_list if c.code == 'USD'), None)
    currency_pen = next(
        (c for c in self.currency_list if c.code == 'PEN'), None)

    # xpath variable - identifies HTML element that is rendered by JavaScript
    xpath = "//div[@class=' flex flex-row items-center']"

    soup = self.get_soup_selenium(provider, xpath)

    # locate the specific HTML elements that contain the price values
    data = soup.find_all('div', class_='flex flex-row items-center')

    buy_price = str(round(Decimal(data[0].find('span').text), 4))
    sell_price = str(round(1 / Decimal(data[1].find('span').text), 4))

    self.save_to_db(provider, currency_usd, currency_pen, buy_price)
    self.save_to_db(provider, currency_pen, currency_usd, sell_price)
```

Joonis 6 Seleniumi kasutamine veebikaabitsas

4.2.3 Valuutakursside statistika

Valuutavahetuskursside statistika kuvamine hetkekursside kõrval on oluline selleks, et kasutaja saaks teha informeeritud otsuse valuuta hinna tõusu või languse põhjal, millal on parim aeg valuuta vahetamiseks.

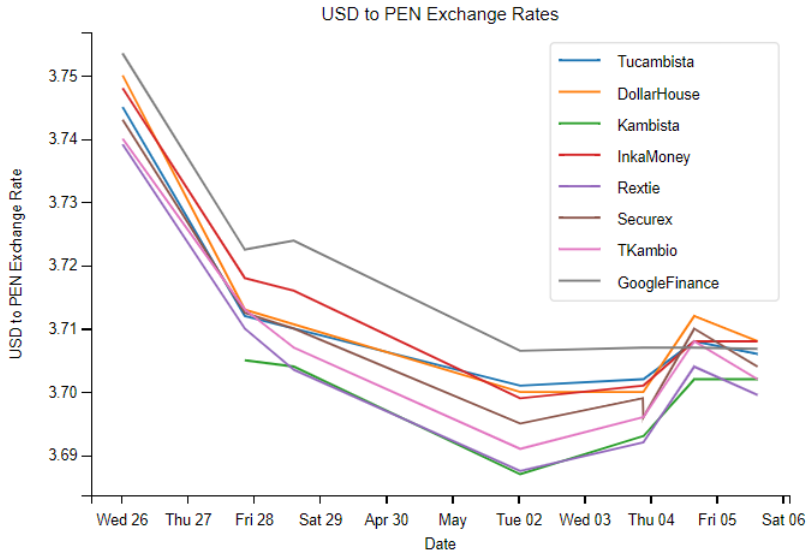
Loodud rakendus kuvab kahte graafikut. Üks graafik kuvab vahetuskursi hindade muutust kui vahetada USA dollareid Peruu sollideks ja teine kuvab vastupidi tehingut, ehk valuuta vahetamist Peruu Sollist USA dollariteks.

Graafikute loomisel kasutati Pythoni teeki Mathplotlib, mis on loodud staatiliste, animeeritud ja interaktiivsete visualisatsioonide loomiseks. Joonisel 7 on näha abimeetodit joondiagrammi loomiseks.

```
def create_rate_chart(form_code, to_code, provider_list):
    fig, ax = plt.subplots(figsize=(6, 4))

    for provider in provider_list:
        # Get the currency rate data for each provider
        rates = CurrencyRate.objects \
                .filter(provider=provider,
                        currency_from__code=form_code,
                        currency_to__code=to_code) \
                .order_by('date_time')
        dates = [rate.date_time for rate in rates]
        rates = [rate.exchange_rate for rate in rates]

        # Add a line to the graph for each provider
        line, = ax.plot(dates, rates, label=provider.name)

        # Add annotation to show provider name on hover
        mplcursors.cursor(line).connect(
            "add", lambda sel: sel.annotation.set_text(provider.name))

    # Add labels and legend
    ax.set_xlabel('Date')
    ax.set_ylabel('{} to {} Exchange Rate'.format(form_code, to_code))
    ax.set_title('{} to {} Exchange Rates'.format(form_code, to_code))
    ax.legend()

    # Convert the plot to an interactive HTML format using mpld3
    interactive_graphic = mpld3.fig_to_html(fig)

    return interactive_graphic
```

Joonis 7 Abimeetod graafikute loomiseks

Loodud graafik on joondiagramm, mis näitab valuutavahetuskursi muutust ajas iga valuutavahetuspunkti kohta. Iga valuutavahetuspunktile kuuluv joon on eri värvi (vt joonist 8).



Joonis 8 Valuutakursi *statistika graafik*

4.3 Veebiteenuse turvalisus.

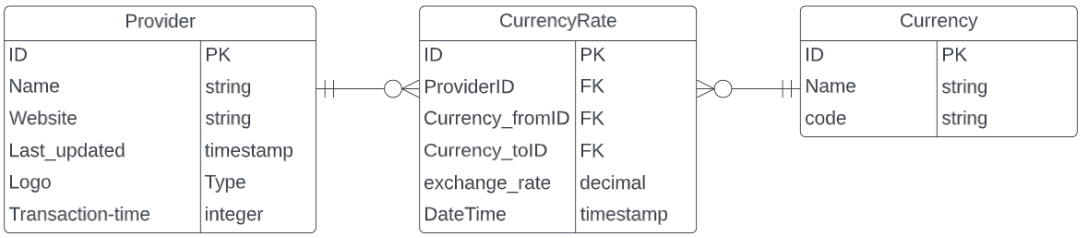
Django raamistikuga tuleb kaasa mitmeid tulvaelemente, näiteks: *cross-site scripting* ehk murdskriptimise kaitse, *cross site request forgery* ehk saitide vaheline päringute võltsimise kaitse, *SQL injection* ehk SQL-süstimise kaitse, *click-jacking* kaitse ja muud. [45]

4.4 Andmebaasi analüüs

Rakenduse juures on oluline osa kliendile näidata, kuidas on valuutakurss aja jooksul muutunud. Kaabitud andmete andmebaasi salvestamine võimaldab säilitada ja töödelda andmeid, ning luua selle abil statistikat valuutakursi muutumise kohta ajas. Lisaks tagab andmebaasi olemasolu kasutajatele juurdepääsu ajakohasele ja täpsele valuutakursile.

Enne andmebaasi loomist tuleb paika panna andmebaasi skeem. Selleks on vaja luua idee, millist informatsiooni peab rakendus hoiustama.

Antud andmebaas peab hoidma valuutavahetus punkte (*Provider*), valuutasid (*Currency*) ja valuuta vahetus kursse (*CurrencyRate*). Joonisel 9 on toodud loodava andmebaasi olem suhte diagramm.



Joonis 9 Andmebaasi skeem

Rakenduse mõistes on kõige olulisem olem valuutavahetuskurss (*CurrencyRate*), sest see hoiab kõige olulisemat informatsiooni rakenduse töötamiseks. Hetkekursiks nimetatakse kõige hilisema *DateTime*'i ehk kirje loomise aja väärtusega kurssi. See peab olema näha nii esilehel kui vastava kursi lehel. Lisaks hoitakse alles ka kõiki eelnevaid valuutavahetuskursse, eesmärgiga koguda informatsiooni valuuta kursi muutumise statistika jaoks, mis on vajalik graafiku loomisel.

Olem valuutavahetuspunkt (Provider) hoiab endas valuutavahetuspunkti nime, kaapimiseks kasutatavat veebilehte, transaktsiooni aega ja kuupäeva, millal viimati uuendati informatsiooni selle valuutavahetuspunkti kohta. Lisaks on võimalik hoida, ka valuutavahetuspunkti logo. Valuutavahetuspunkt on seotud valuutakursiga üks-mitmele suhtega, mis võimaldab, et valuutavahetuspunktil ei ole ühtegi kurssi.

Valuuta (*Currency*) olem hoiab valuutasid, mille vahetuskursse rakendus näitab. Valuuta on defineeritud kahe väljaga: nimi (*Name*), mis hoiab täispikka valuuta nime ning kood (*Code*), mis hoiab valuuta lühikest nime ehk koodi. Valuuta olem on seotutud valuutavahetus kursiga kahe *Foreign Key* ehk võõrvõtmega, üks neist näitab alg- ja teine lõppvaluutat. Ühel valuutal võib olla mitu või mitte ühtegi valuutakurssi.

Andmete salvestamiseks kasutati Django poolt olemasolevat andmemudelit *Django Models* ehk Mudelid, mille abil on võimalik luua andmebaasi olemid ning nendega koodis suhelda. *Models* sisaldab klasside definitsioone, mille abil on võimalik luua andmebaasi tabelid, nende väljad ja suhted. Lisaks võimaldab see ka andmete valideerimist ja andmebaasiga suhtlemist. [46]

Andmebaasi olemite loomiseks on Django loodud fail models.py. Joonisel 10 on näha models.py fail, kus on loodud klassid vastavalt andmebaasi skeemile.

```
from django.db import models

class Provider(models.Model):
    name = models.CharField(max_length=200)
    website = models.CharField(max_length=500)
    last_update = models.DateTimeField()
    logo = models.ImageField(upload_to='logos')
    transaction_time = models.IntegerField()
    is_neutral = models.BooleanField()

    def __str__(self):
        return self.name


class Currency(models.Model):
    name = models.CharField(max_length=200)
    code = models.CharField(max_length=20)

    def __str__(self):
        return self.name


class CurrencyRate(models.Model):
    provider = models.ForeignKey(Provider, on_delete=models.CASCADE)
    currency_from = models.ForeignKey(Currency, on_delete=models.CASCADE,
related_name='curency_from', null=True)
    currency_to = models.ForeignKey(Currency, on_delete=models.CASCADE,
related_name='curency_to', null=True)
    exchange_rate = models.DecimalField(decimal_places=4, max_digits=16)
    date_time = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return "{}: {} to {}  - {} at {}".format(
            self.provider.name,
            self.currency_from.code,
            self.currency_to.code,
            self.exchange_rate,
            self.date_time)
```

Joonis 10 Django mudelid

Veebikaabitsa kogutud andmete salvestamiseks loodi staatiline abimeetod, mis võtab argumentidena sisse valuutavahetuspunkti, algvaluuta, tulemusvaluuta ja vahetuskursi. Meetod loob *CurrencyRate* tüüpi objekti ning salvestab selle andmebaasi (Joonis 11).

```
@staticmethod
def save_to_db(provider, currency_from, currency_to, exchange_rate):
    rate = CurrencyRate(
            provider=provider,
            currency_from=currency_from,
            currency_to=currency_to,
            exchange_rate=exchange_rate)
    rate.save()
```

Joonis 11 Abimeetod andmete salvestamiseks andmebaasi

5 Klientrakenduse lahendus

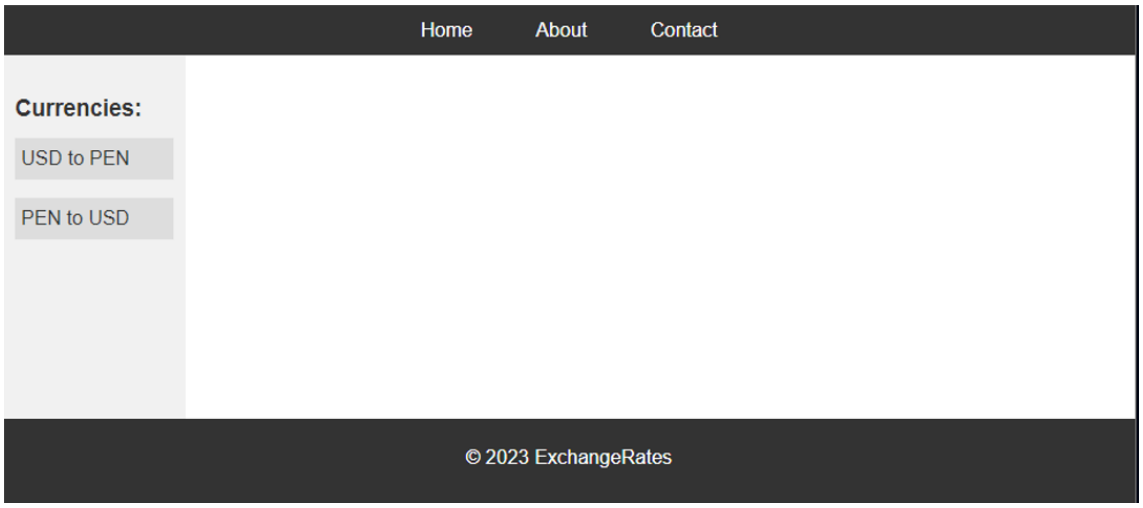
Klientrakenduse lahendusel kasutati Django poolt loodud malle veebilehtede kuvamiseks ning lisafunktsionaalsus lisati JavaScripti funktsioonide abil. Lehe disainimisel kasutati CSS-i.

5.1 Django mallimootor

Django veebirakenduste raamistik sisaldab mitmeid tööriistu, sealhulgas mallisüsteemi. Django mallimootor võimaldab eraldada veebilehtede kujundust ja sisu.

Django mallimootor võimaldab luua malle, kasutades HTML-dokumentide sees olevaid eriliste siltide ja avaldiste kasutamist. Malle saab kasutada ka andmebaasist saadud andmete kuvamiseks.

Mallid salvestatakse tavaliselt „*templates*“ nimelisse kausta ja neid saab laiendada base.html malliga. Malli laiendamise abil saab korduvat koodi vältida ja lehe struktuuri lihtsustada. Mallides saab kasutada ka tingimuslausetega ja tsükliga sarnaseid konstruktsioone. Joonisel 12 on näha lehe osi, mis on genereeritud baasmalli poolt. Joonisel 13 on näha base.html koodi, mis lehte genereerib



Joonis 12 base.html poolt genereeritav leht.

```
{% load static %}     <!-- load all static files  -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}{% endblock %}</title>
    <!-- import style sheet  -->
    <link rel="stylesheet" href="{% static 'convert/main.css' %}">
    <!-- import javascript   -->
    <script src="{% static 'convert/script.js' %}"></script>
   <title>Currency Exchange Rates</title>
</head>
<body onload="highlight_highest_lowest()">
    <nav>
        <ul>
            <!-- links for the main menu  -->
            <li><a href="{% url 'convert:index' %}">Home</a></li>
            <li><a href="{% url 'convert:about' %}">About</a></li>
            <li><a href="{% url 'convert:contact' %}">Contact</a></li>
        </ul>
    </nav>
   <div class="container">
      <div class="side-menu">
        <ul>
            <!-- links for the side menu  -->
            <li><a href="{% url 'convert:currency_rates' currency_from='USD'
currency_to='PEN' %}">USD to PEN</a></li>
            <li><a href="{% url 'convert:currency_rates' currency_from='PEN'
currency_to='USD' %}">PEN to USD</a></li>
        </ul>
      </div>
      <div class="content">
        {% block content %}
                <!-- content form all pages extending this base  -->
        {% endblock %}
      </div>
    </div>
    <footer>
        <p>&copy; 2023 ExchangeRates</p>
    </footer>
</body>
</html>
```

Joonis 13 base.html

5.2 Django staatilised failid

Staatilised failid, nagu CSS ja JavaScript on koodifailid, mis ei muutu sõltuvalt rakenduse käivitamisest. Need failid saadetakse brauserile selleks, et see saaks rakendusele lisada stiili ja interaktiivsuse.

5.2.1 CSS stiilifail

Django rakenduses kasutatakse staatilisi faile, et eraldada veebilehe sisu ja selle stiili. Näiteks võib mitu erineva sisuga veebilehte kasutada sama stiili. Sellisel juhul kasutatakse CSS faili, et luua korduvkasutatav stiilimall, mida on võimalik rakendada kõikidel lehtedel.

Selleks, et lehe baasosa nagu joonisel 12 välja näeks kasutati CSS faili, mis asub Django rakenduse kasutas olevas „*static*“ kaustas. Joonisel 14 on näha osa CSS stiilifailist, mis on seotud lehe baasosa kujundusega.

```css
/* Reset styles */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
/* Basic styles for the body */
body {
  font-family: Arial, sans-serif;
  font-size: 16px;
  line-height: 1.5;
  color: #333;
  background-color: #fff;
/* Header styles */
header {
  background-color: #333;
  color: #fff;
  padding: 20px;
}
/* Navigation menu styles */
nav {
  display: flex;
  justify-content: center;
    background-color: #333;
}
nav ul {
  display: flex;
  list-style: none;
}
nav ul li {
  margin-right: 20px;
}
nav ul li:last-child {
  margin-right: 0;
}
nav ul li a {
  color: #fff;
  text-decoration: none;
  display: inline-block
  padding: 8px 16px;
  background-color: #333;
}
nav ul li a:hover {
  text-decoration: underline;
}
/* Footer styles */
footer {
  background-color: #333;
  color: #fff;
  padding: 20px;
  text-align: center;
}
```

Joonis 14 lehe baasosaga seotud osa stiilifailist

5.2.2 JavaScript fail

JavaScript on skriptimis- või programmeerimiskeel, mis võimaldab lisada veebilehele dünaamilist sisu [47]. Näiteks vormide valideerimist, animatsioone, teatekaste, andmetabelite sorteerimist ja filtreerimist ning palju muud.

Django rakenduses saab JavaScript faile kasutada kliendi poolele funktsioonide lisamiseks. Näiteks kasutaja liidese parandamiseks, et muuta kasutajaliides dünaamilisemaks ja interaktiivsemaks. Need failid võivad olla spetsiaalselt loodud iga Django rakenduse jaoks või võivad ka olla üldises kasutuses mitme rakenduse vahel.

Django raamistik võimaldab mitmeid viise, kuidas JavaScript faile oma rakendusega ühendada. Üheks võimaluseks on kasutada „*script*“ elementi HTML-mallides, et JavaScript faili sisu otse veebilehele lisada. Teiseks võimaluseks on kasutada Django „*static*“ malli märgendit, et viidata JavaScript faili asukohale staatiliste failide kaustas.

Joonisel 15 on näide JavaScript failis olevast funktsioonist, mis värvib kõrgeima ja madalaima kursi hinna vastavalt CSS failis olevale „*highest*“ ja „*lowest*“ märgendile.

```
function highlight_highest_lowest() {
// Find the cell with the highest exchange rate
    let numColumns = document.getElementById("myTable").rows[0].cells.length;
    for (let i = 1; i < numColumns; i++) {
        // Find the cell with the highest exchange rate for this column
        let maxRate = 0;
        let maxRateCell = null;
        let minRate = Number.MAX_VALUE;
        let minRateCell = null;
        let rateCells = document.getElementsByClassName("rate");
        for (let j = i; j < rateCells.length; j += 1) {
            if ((j-i) % (numColumns-1) === 0) {
                let rateValue = parseFloat(rateCells[j]
                                                    .getAttribute("value"));
                if (rateValue > maxRate) {
                    maxRate = rateValue;
                    maxRateCell = rateCells[j];
                }
                if (rateValue < minRate) {
                minRate = rateValue;
                minRateCell = rateCells[j];
                }
            }
        }
        // Add a CSS class to the cell with the highest and lowest exchange
rate for this column
        maxRateCell.classList.add("highest");
        minRateCell.classList.add("lowest");
    }
```

Joonis 15 JavaScripti funktsioon

6 Kasutajaliides

Kasutajakogemus on üks olulisimaid komponente veebirakenduse arenduses. Rakenduse intuitiivne ja lihtne disain, kus kasutajal on kõik vajalik silma all on vajalik, et kasutaja tunneks end hästi rakendust kasutades ning selleks, et ta edaspidi rakenduse juurde ka tagasi tuleks.

Käesoleva lõputöö raames loodud rakendus koosneb kahest põhivaatest. Esileht ja valuutavaade, kus on näha vaid kahe valuuta vaheline ühepidine kurss.

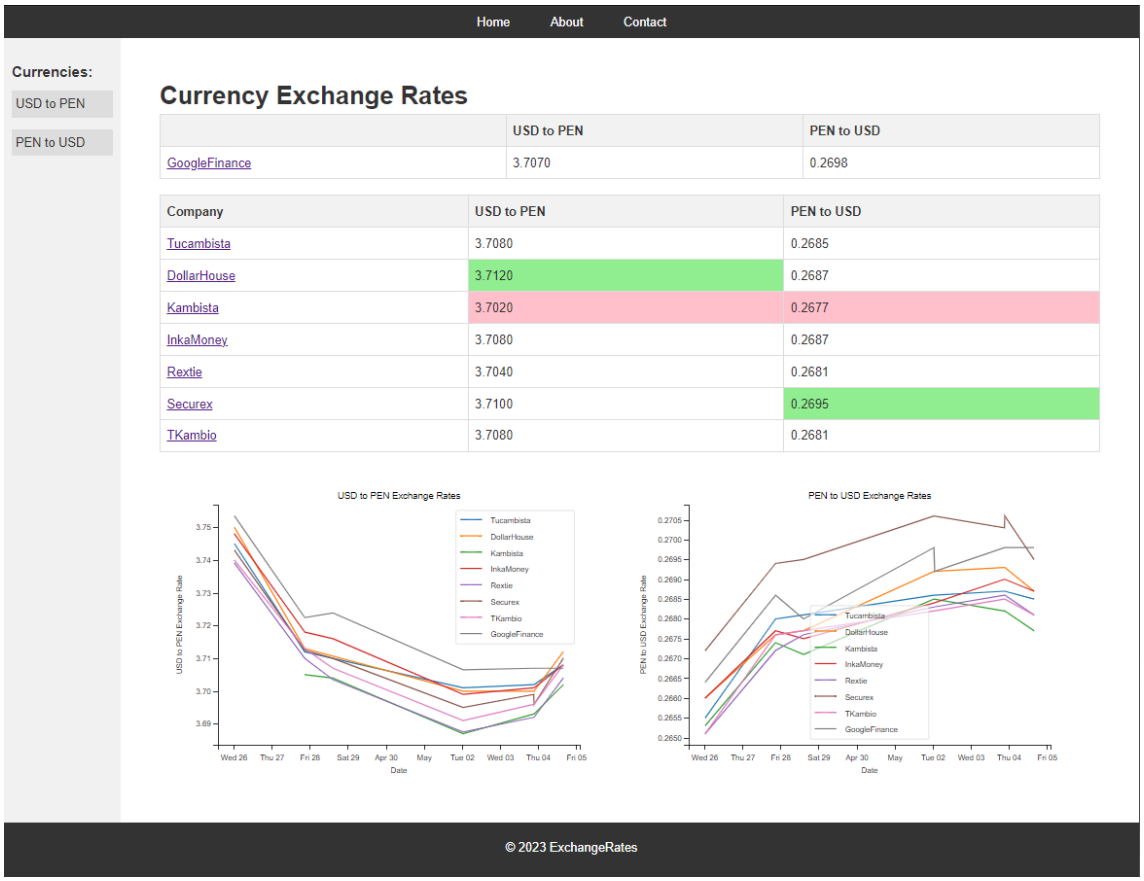
6.1 Esileht

Esilehel on tabel, kus on kuvatud usaldusväärne, neutraalne valuutavahetus kurss, mis enamjaolt ühtib turu keskmise kursiga. See on seal selleks, et kasutaja, ei peaks valuutavahetuspunktide hindade võrdlemiseks otsima teistelt veebilehtedelt neutraalset kurssi.

Teises tabelis on kuvatud valuutavahetuspunktide kaupa viimane saadaolev kurss, vahetamaks raha kas USA dollaritest Peruu sollideks või vastupidi. Parim hind on värvitud roheliseks ning madalaim hind punaseks. Klõpsates valuutavahetuspunkti nimele, viib link automaatselt antud valuutavahetuspunkti kodulehele, kus on võimalik alustada valuuta vahetamist.

Lehe alumises otsas on näha mõlema valuutakursi statistikat visualiseerivat joondiagrammi, kus iga joon vastab ühele valuutavahetuspunktile.

Joonisel 16 on näha veebirakenduse esilehe kujundust.



Joonis 16 Veebirakenduse esileht

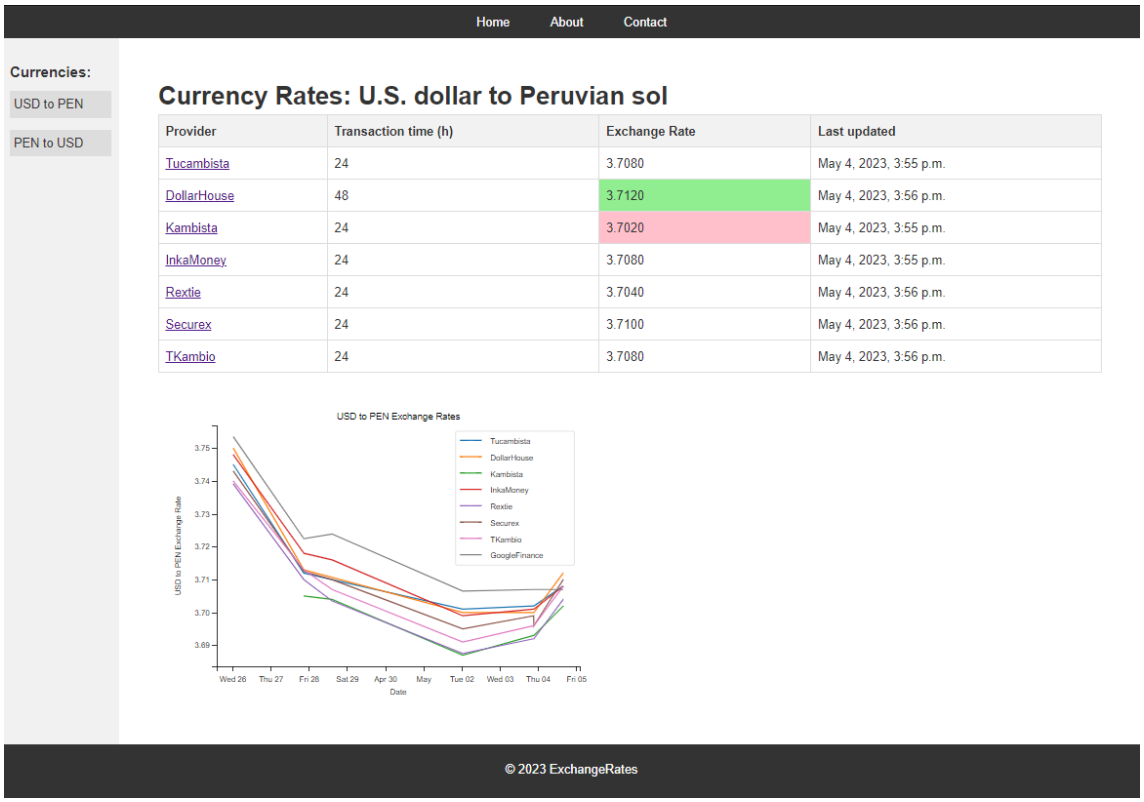
6.2 Valuutavaade

Valuutavaatel on suures plaanis sama funktsionaalsus mis esilehel. Vaate eesmärk on vähendada informatsiooni, mida kasutaja näeb ning lisada lehele ka detailsem informatsioon sellest, mis kuupäeval ja kellajal antud valuutakurssi viimati uuendati ja kui pikk on transaktsiooni aeg.

Valuuta kursi viimane uuendamise aeg on oluline, et kasutaja saaks otsustada kas valuuta kurss on veel asjakohane. Valuuta kursi uuendamise probleem võib olla rakenduse- ja ka valuuta vahetaja poolne.

Lisaks on lehe all osas kuvatud ka joon diagramm, selle valuuta kursi kohta. Diagramm visualiseerib, kuidas on valuuta kurss erinevates valuutavahetuspunktides aja jooksul muutunud.

Joonisel 17 on näha valuutavaadet.



Joonis 17 Valuutavaade USD to PEN

7 Rakenduse testimine

Rakenduse töös ning kasutajasõbralikkuses veendumiseks tuleb rakendust ka testida. Läbi viidi kahte sorti testid: kasutaja kogemuse testid ja rakenduse funktsionaalsuse testimine.

7.1 Rakenduse funktsionaalsuse testimine

Rakenduse funktsionaalsuse testimine on jaotatud kolme suuremasse osasse: Django mudelite testimine, Django vastete testimine ning äriloogika ehk selle lõputöö raames veebikaabitsate testimine.

Testide kirjutamiseks loodi Django rakenduse sisse kaust „*tests*“, mille sisse lisati __init__.py fail, mis muudab kausta mooduliks ning võimaldab teste jooksutada käsurealt. Iga osa jaoks loodi eraldi test fail, mis sisaldab vastava osa teste.

7.1.1 Mudelite testimine

Mudelite testimisel on oluline kontrollida, kas kõik väljad on õigesti defineeritud ning kas kõik lisameetodid tagastavad soovitud tulemuse. Lisaks tuleb testida erinevaid erijuhtumeid, mis võivad koodi katki teha. Mudelite testimise koodi näited on leitavad Lisa 3 alt.

7.1.2 Vaadete testimine

Vaadete testimisel kontrollitakse, kas vaate funktsioonid tagastavad korrektse HTML vastuse ning kas sisendid on õigesti käsitletud. Lisaks tuleb kontrollida, kas lehele saadetud andmed on korrektselt kuvatud ning kas kuvatud andmed on ise korrektsed. Oluline on ka testida, et vaade töötab tõhusalt ega tee tarbetuid andmebaasipäringuid ega muid viivitusi. Vaadete testimise näited on leitavad Lisa 4 alt.

7.1.3 Veebikaabitsa testimine

Veebikaabitsa testimisel tuleb kontrollida, et kaabits eraldab lehelt õiged andmed ning muudab need vajalikule kujule. Lisaks tuleb kontrollida, et eraldatud andmed jõuavad andmebaasi soovitud kujul. Veebikaabitsa ja selle abimeetodite testimise näited on leitavad Lisa 5 alt.

Veebikaabits peab olema valmis ka olukorra jaoks, kui veebilehe kood muutub, ning enam ei ole võimalik informatsiooni sama viisiga kaapida. Seega on oluline testida ka veebikaabitsa veahaldust.

7.2 Veebirakenduse kasutajakogemuse testimine

Kasutajakogemuse testimiseks loodi küsitlus, kus paluti kasutajatel hinnata erinevaid aspekte rakenduse kasutaja vaatepunktist. Küsitlus koosnes üheksast küsimusest, kus kasutaja sai tagasisidet anda rakenduse välimuse, töö ja vajalikkuse kohta. Küsimustikule andis vastuse kümme kasutajat. Küsimustiku vastuseid ja nende statistikat on võimalik näha peatükis Lisa 2.

Kokkuvõttes olid 80% vastanutest rahul rakenduse tööga ning 20% arvasid et rakendus on kasutamiseks piisavalt hea. Lehe jõudluse ja laadimiskiirusega olid rahul 90% vastanutest ning 10 % arvas, et seda peaks veel parandama. Vaid 10 % vastanutest arvas, et leidub sama funktsiooniga rakendusi, kuid kõik vastanud leidsid, et rakendus on neile vajalik. Kasutaja kogemusega olid täielikult rahul 90% vastanutest ning 10% leidsid et seda peaks veel parandama. Kõik vastanutest soovitaksid rakendust ka teistele.

8 Kokkuvõte

Antud lõputöö eesmärk on luua veebirakenduse prototüüp, mis kogub erinevate virtuaalsete valuutavahetuspunktide vahetuskursid ühele lehele. Platvormi olulisus peitub võimaluses võrrelda erinevate valuutavahetuspunktide kursse ning valida neist soodsaim.

Prototüübi tehnoloogilist lahendust analüüsiti põhjalikult ning lahendusteks valiti antud ülesande täitmiseks kõige sobivamad tehnoloogiad. Kasutaja kogemuse testimine näitas, et platvorm on kasutajasõbralik ning lihtne kasutada.

Rakendus täidab kõiki varasemalt püstitatud funktsionaalseid nõudeid. Tulevikus on võimalik rakendust edasi arendada, lisades rohkem funktsionaalsusi ning rakenduse haldust lihtsustavaid võimalusi. Esimeseks sammuks on kindlasti veebikraabitsa monitooringu lisamine, et tagada tõrgeteta rakenduse töö ja hea kasutajakogemus kõigile kasutajatele.

Rakendus saavutas algselt püstitatud eesmärgi hoida kokku kasutaja aega ja raha kogudes ühele lehele vähemalt seitsme virtuaalse valuutavahetuspunkti valuutavahetuskursi. Kasutaja saab kursse omavahel kiirelt võrrelda või valida kurss, mida rakendus parimaks loeb.

Kokkuvõttes annab käesolev lõputöö panuse valuutavahetuse valdkonda, luues praktilise rakenduse, mis hõlbustab leida parima valuutavahetuskursi pakutavate seast. Autor loodab, et loodud rakendusest on kasu ning see innustab ka tulevikus valuutavahetuse valdkonnas edasist arengut ja innovatsiooni.

Kasutatud kirjandus

[1] „BestExchangeRates,“ Best Exchange Rates Pty Ltd, [Võrgumaterjal]. Available: https://bestexchangerates.com/about/. [Kasutatud 16 Aprill 2023].

[2] Wise Payments Limited, „Wize,“ Wise Payments Limited, [Võrgumaterjal]. Available: https://wise.com/gb/about/our-story. [Kasutatud 16 aprill 2023].

[3] „Web Application Development: A Detailed Guide for 2023,“ Fingent, 2023. [Võrgumaterjal]. Available: https://www.fingent.com/blog/web-application-development-a-detailed-guide/. [Kasutatud 13 Aprill 2023].

[4] Altexsoft, „Functional and Nonfunctional Requirements: Specification and Types,“ Altexsoft, 23 Juuli 2021. [Võrgumaterjal]. Available: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/. [Kasutatud 10 Mai 2023].

[5] „Digipädevus,“ Haridus- ja noorteamet, [Võrgumaterjal]. Available: https://digipadevus.ee/sonastik/veebirakendus/ . [Kasutatud 23 Märts 2023].

[6] S. d. S. Sirisuriya, „A Comparative Study on Web Scraping,“ %1 *Poceedings of 8th International Research Conference, KDU*, Ratmalana, 2015.

[7] 99corps, „Mozilla developer centre,“ MindTouch Standard, 8 Jaanuar 2009. [Võrgumaterjal]. Available: https://web.archive.org/web/20100106122312/https://developer.mozilla.org/en/About_javascript. [Kasutatud 10 aprill 2023].

[8] B. Stroustrup, „An overview of C++.,“ %1 *Proceedings of the 1986 SIGPLAN workshop on Object-oriented programming*, 1986.

[9] TIOBE, „TIOBE Index for April 2023,“ TIOBE, [Võrgumaterjal]. Available: https://www.tiobe.com/tiobe-index/. [Kasutatud 20 Aprill 2023].

[10] L. M., „BitDegree,“ BitDegree.org, 2 Märts 2023. [Võrgumaterjal]. Available: https://www.bitdegree.org/tutorials/python-vs-javascript/. [Kasutatud 4 Aprill 2023].

[11] T. J. Kidder, „Learn Academy,“ 10 Veebruar 2020. [Võrgumaterjal]. Available: https://learnacademy.org/blog/what-best-way-easy-hard-learn-ruby-rails. [Kasutatud 5 Aprill 2023].

[12] E. Schaffer, 18 Jaanuar 2022. [Võrgumaterjal]. Available: https://www.educative.io/blog/learn-cpp-for-2022. [Kasutatud 4 Märts 2023].

[13] F. Mahboob, „The Tech Boogle,“ 2 Detsember 2022. [Võrgumaterjal]. Available: https://thetechboogle.com/web-scraping-java-vs-python-which-is-better/. [Kasutatud 5 Märts 2023].

[14] „The Best Programming Languages for Web Scraping,“ Scrape-It, [Võrgumaterjal]. Available: https://scrape-it.cloud/blog/web-scraping-languages. [Kasutatud 04 Mai 2023].

[15] N. Duggal, „simplilearn,“ 22 Veebruar 2023. [Võrgumaterjal]. Available: https://www.simplilearn.com/python-frameworks-article. [Kasutatud 7 Märts 2023].

[16] Eduonix Learning Solutions, „Eduonix,“ Eduonix Learning Solutions, 31 Jaanuar 2022. [Võrgumaterjal]. Available: https://blog.eduonix.com/video-tutorials/web-development-tutorials/full-stack-vs-micro-framework-in-web-development/. [Kasutatud 7 Märts 2023].

[17] Techno Lush, „Micro Vs Full-Stack Frameworks,“ Techno Lush, 20 Jaanuar 2019. [Võrgumaterjal]. Available: https://www.technolush.com/blog/micro-vs-full-stack-frameworks. [Kasutatud 4 Mai 2023].

[18] JetBrains, „Python Developers Survey 2021 Results,“ JetBrains, 2021.

[19] CodeAhoi, „CodeAhoi,“ CodeAhoi, [Võrgumaterjal]. Available: https://codeahoy.com/compare/django-vs-web2py. [Kasutatud 10 Märts 2023].

[20] Stack overflow, "2022 Developer Survey," Stack Overflow, 2022.

[21] M. Smallcombe, „Integrate.io,“ 15 Veebruar 2023. [Võrgumaterjal]. Available: https://www.integrate.io/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/. [Kasutatud 15 Märts 2023].

[22] A. Yigal, „Sqlite vs. MySQL vs. PostgreSQL: A Comparison of Relational Databases,“ Logz, [Võrgumaterjal]. Available: https://logz.io/blog/relational-database-comparison/. [Kasutatud 4 Mai 2023].

[23] „What Is the Client Side?,“ Elementor, [Võrgumaterjal]. Available: https://elementor.com/resources/glossary/what-is-the-client-side/. [Kasutatud 10 Aprill 2023].

[24] Indeed Editorial Team, „Career Guide: Client-Side vs. Server-Side: What's the Difference?,“ Indeed, 10 Märts 2023. [Võrgumaterjal]. Available: https://www.indeed.com/career-advice/career-development/client-side-vs-server-side. [Kasutatud 10 Aprill 2023].

[25] „HTML basics,“ Mozilla, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics. [Kasutatud 10 Aprill 2023].

[26] „CSS basics,“ Mozilla, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics. [Kasutatud 12 Aprill 2023].

[27] „JavaScript basics,“ Mozilla, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics. [Kasutatud 12 Aprill 2023].

[28] „JavaScript,“ Mozilla, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Kasutatud 12 Aprill 2023].

[29] Thinkful, „How Hard is it to Learn HTML?,“ Thinkful, [Võrgumaterjal]. Available: https://www.thinkful.com/blog/how-hard-is-it-to-learn-html/. [Kasutatud 4 Mai 2023].

[30] L. D. JamesGallagher, „Learn CSS: Best Courses, Books, and Resources for Learning CSS,“ Careerkarma, 14 August 2022. [Võrgumaterjal]. Available: https://careerkarma.com/blog/learn-css/. [Kasutatud 4 Mai 2022].

[31] M. U. Njoku, „How Hard Is It to Learn JavaScript?,“ CareerKarma, 7 Jaanuar 2022. [Võrgumaterjal]. Available: https://careerkarma.com/blog/is-javascript-hard-to-learn/. [Kasutatud 4 Mai 2023].

[32] „What is CSS for?,“ Mozila, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS#what_is_css_for. [Kasutatud 4 Mai 2022].

[33] „Build a modern web app using Django and Javascript,“ Advantch, 24 Oktoober 2021. [Võrgumaterjal]. Available: https://www.advantch.com/blog/build-a-modern-web-app-using-django-and-javascript/. [Kasutatud 13 Aprill 2023].

[34] Sonatype, „Sonatype,“ Sonatype Inc, [Võrgumaterjal]. Available: https://www.sonatype.com/launchpad/what-are-code-repositories. [Kasutatud 11 Märts 2023].

[35] Suse, „SUSE,“ [Võrgumaterjal]. Available: https://www.suse.com/suse-defines/definition/development-environment/. [Kasutatud 11 Märts 2023].

[36] GitHub, „About,“ GitHub, [Võrgumaterjal]. Available: https://github.com/about. [Kasutatud 13 Märts 2023].

[37] A. Gupta, „Top 15+ Python IDEs in 2023: Choosing The Best One,“ Simplilearn, 12 Aprill 2023. [Võrgumaterjal]. Available: https://www.simplilearn.com/tutorials/python-tutorial/python-ide. [Kasutatud 15 Aprill 2023].

[38] Tutorialspoint, „Django-Overview,“ Tutorialspoint, [Võrgumaterjal]. Available: https://www.tutorialspoint.com/django/django_overview.htm. [Kasutatud 10 Aprill 2023].

[39] „Django Documentation - Models,“ Dajngo, [Võrgumaterjal]. Available: https://docs.djangoproject.com/en/4.2/topics/db/models/#module-django.db.models. [Kasutatud 10 Aprill 2023].

[40] „Django Documentation - Views,“ Django, [Võrgumaterjal]. Available: https://docs.djangoproject.com/en/dev/topics/http/views/#writing-views. [Kasutatud 10 Aprill 2023].

[41] „Django Documentation - Templates,“ Django, [Võrgumaterjal]. Available: https://docs.djangoproject.com/en/4.1/topics/templates/. [Kasutatud 10 Aprill 2023].

[42] TechVidvan, „Django Project Structure and File Structure,“ TechVidvan, [Võrgumaterjal]. Available: https://techvidvan.com/tutorials/django-project-structure-layout/. [Kasutatud 10 Aprill 2023].

[43] „Beautiful Soup Documentation,“ Leonard Richardson, [Võrgumaterjal]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/. [Kasutatud 14 Aprill 2023].

[44] „Selenium vs BeautifulSoup in 2023: Which Is Better?,“ ZenRows, 20 Jaanuar 2023. [Võrgumaterjal]. Available: https://www.zenrows.com/blog/selenium-vs-beautifulsoup#what-are-the-advantages-of-beautifulsoup. [Kasutatud 15 Aprill 2023].

[45] Django, „Django Documentation - Security,“ Django, [Võrgumaterjal]. Available: https://docs.djangoproject.com/en/4.0/topics/security/. [Kasutatud 11 Aprill 2023].

[46] P. B. W. C. Jeff Forcier, Python Web Development with Django, Addison-Wesley Professional, 2008.

[47] „What is JavaScript?,“ Developer Mozilla.org, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-

US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. [Kasutatud 20 Aprill 2023].

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

Mina, Mari-Lotta Moldau

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

14.05.2023

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

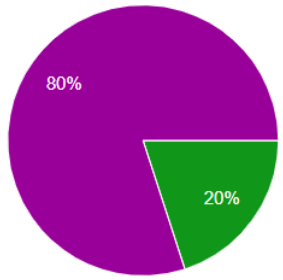
Lisa 2 – Kasutajakogemuse küsitluse vastused

Küsitlus on leitav aadressil: https://forms.gle/73xNMRC7EHhJ8wbg9

Kuidas hindate rakenduse üldist tööd?

Kopeeri

10 vastust



- 1 - ei kasutaks rakendust teist korda
- 2 - rakendus ei vasta ootustele
- 3 - rakendus on rahuldav, kuid on palju, mida parandada
- 4 - rakendus on hea, kuid puuduvad mõned asjad
- 5 - rakendus on väga hea, ning kasutaksin seda ka edaspidi

Kas rakendus täidab teie ootusi ja vajadusi?

Kopeeri

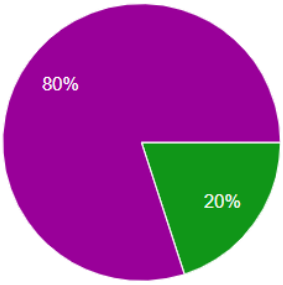
10 vastust

- Jah
- Ei
- Enam vähem

Kuidas hindate rakenduse kasutajaliidest ja disaini

Kopeeri

10 vastust

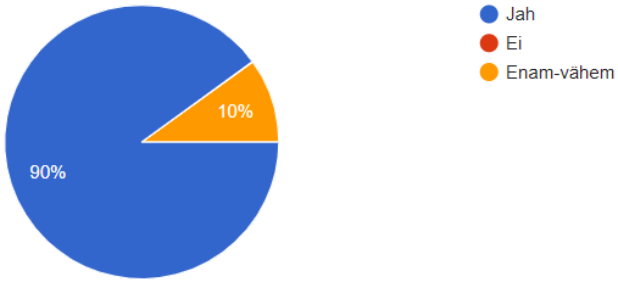


- 1 - rakendus ei ole üldse kasutajasõbralik
- 2 - rakendus on natukene kasutajasõbralik
- 3 - rakendus on enam vähem kasutajasõbralik
- 4 - rakendus on piisavalt kasutajasõbralik
- 5 - rakendus on väga kasutajasõbralik

Kas rakenduse funktsioonid on selged ja lihtsasti kasutatavad?

Kopeeri

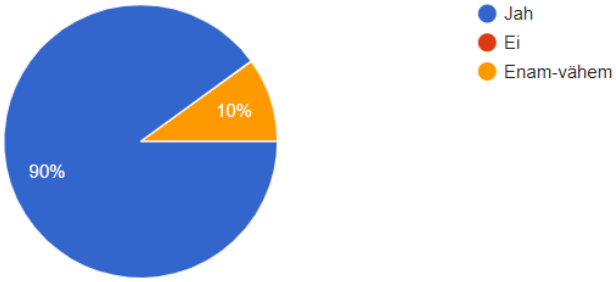
10 vastust



Kas rakenduse navigeerimine on intuitiivne ja lihtne?

Kopeeri

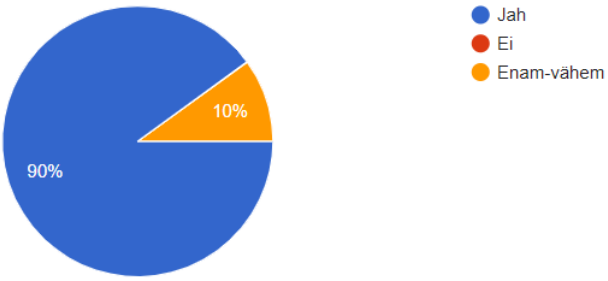
10 vastust



Kas rakenduse laadimiskiirus ja jõudlus vastavad teie ootustele?

Kopeeri

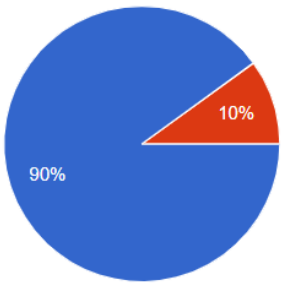
10 vastust



Kas rakendus on teie jaoks vajalik või leiaksite samu funktsioone mujalt?

Kopeeri

10 vastust

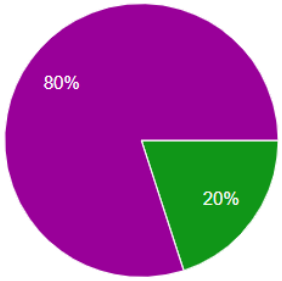


- Jah, rakendus on mulle vajalik
- Rakendus on vajalik, kuid samalaadseid võimalusi leab ka mujallt.
- Ei, rakenduse funktsioone leiaksin ka mujalt

Kuidas hindate rakenduse kasutamise kogemust?

Kopeeri

10 vastust



- 1 - ei kasutaks rakendust teist korda
- 2 - rakenduse kasutajakogemus ei vasta ootustele
- 3 - kasutajakogemus on talutav
- 4 - kasutajakogemus on piisavalt hea
- 5 - kasutajakogemus on väga hea

Kas soovitaksite seda rakendust teistele?

Kopeeri

10 vastust

- Jah
- Ei

Lisa 3 – Mudelite testimise näidised

```
from django.test import TestCase
from convert.models import Provider, Currency, CurrencyRate
from django.core.files.uploadedfile import SimpleUploadedFile


class ProviderModelTestCase(TestCase):

    def setUp(self):
        self.provider = Provider.objects.create(
            name='Test Provider',
            website='http://www.example.com',
            last_update='2022-01-01 12:00:00',
            logo=SimpleUploadedFile("logo.jpg",
                                    b"file_content",
                                    content_type="image/jpeg"),
            transaction_time=3,
            is_neutral=True
        )

    def test_string_representation(self):
        self.assertEqual(str(self.provider), 'Test Provider')

    def test_logo_upload(self):
        self.assertIsNotNone(self.provider.logo)

    def test_transaction_time_type(self):
        self.assertIsInstance(self.provider.transaction_time, int)

    def test_is_neutral_type(self):
        self.assertIsInstance(self.provider.is_neutral, bool)

    def test_website_max_length(self):
        max_length = self.provider._meta.get_field('website').max_length
        self.assertEquals(max_length, 500)

    def ...


class CurrencyModelTestCase(TestCase):

    def ...

class CurrencyRateModelTest(TestCase):

    def ...
```

Lisa 4 – Vaadete testimise näited

```
from django.test import TestCase
from django.urls import reverse

from decimal import Decimal
from datetime import datetime

from convert.models import Provider, Currency, CurrencyRate


class CurrencyRateListViewTest(TestCase):
    def setUp(self):
        # Create currencies
        self.usd = Currency.objects.create(name="US Dollar", code="USD")
        self.eur = Currency.objects.create(name="Euro", code="EUR")
        self.pen = Currency.objects.create(name="Peruvian Sol", code="PEN")

        # Create providers
        self.provider1 = Provider.objects.create(
                                    name="Provider 1",
                                    is_neutral=False,
                                    last_update='2022-01-01 12:00:00',
                                    transaction_time=3)
        self.provider2 = Provider.objects.create(
                                    name="Provider 2",
                                    is_neutral=False,
                                    last_update='2022-01-01 12:00:00',
                                    transaction_time=3, )
        self.provider3 = Provider.objects.create(
                                    name="Provider 3",
                                    is_neutral=True,
                                    last_update='2022-01-01 12:00:00',
                                    transaction_time=3, )

        # Create currency rates
        self.rate1 = CurrencyRate.objects.create(
            provider=self.provider1,
            currency_from=self.usd,
            currency_to=self.pen,
            exchange_rate=Decimal('3.5'),
            date_time=datetime(2022, 5, 3, 10, 0, 0)
        )
        self.rate2 = CurrencyRate.objects.create(
            provider=self.provider1,
            currency_from=self.usd,
            currency_to=self.pen,
            exchange_rate=Decimal('3.6'),
            date_time=datetime(2023, 5, 4, 10, 0, 0)
        )
        self.rate3 = CurrencyRate.objects.create(
            provider=self.provider2,
            currency_from=self.usd,
            currency_to=self.pen,
            exchange_rate=Decimal('3.7'),
            date_time=datetime(2023, 5, 4, 10, 0, 0)
        )
```

```
        self.rate4 = CurrencyRate.objects.create(
            provider=self.provider2,
            currency_from=self.usd,
            currency_to=self.pen,
            exchange_rate=Decimal('3.8'),
            date_time=datetime(2023, 5, 5, 10, 0, 0)
        )
        self.rate5 = CurrencyRate.objects.create(
            provider=self.provider3,
            currency_from=self.usd,
            currency_to=self.pen,
            exchange_rate=Decimal('3.9'),
            date_time=datetime(2023, 5, 5, 10, 0, 0)
        )

    def test_view_url_exists_at_desired_location(self):
        url = '/convert/USD-PEN/'
        response = self.client.get(url)
        self.assertEqual(response.status_code, 200)

    def test_view_url_accessible_by_name(self):
        url = reverse('convert:currency_rates', args=['USD', 'PEN'])
        response = self.client.get(url)
        self.assertEqual(response.status_code, 200)

    def test_view_uses_correct_template(self):
        url = reverse('convert:currency_rates', args=['USD', 'PEN'])
        response = self.client.get(url)
        self.assertTemplateUsed(response, 'convert/currency_rate_list.html')

    def test_view_displays_latest_rates(self):
        url = reverse('convert:currency_rates', args=['USD', 'PEN'])
        response = self.client.get(url)
        self.assertContains(response, 'Provider 1')
        self.assertContains(response, 'Provider 2')
        self.assertContains(response, 'Provider 3')
        self.assertContains(response, '3.8000')
        self.assertNotContains(response, '3.7000')
        self.assertNotContains(response, '3.5000')
```

Lisa 5 – Veebikaabitsa testide näited

```
from bs4 import BeautifulSoup
from django.core.files.uploadedfile import SimpleUploadedFile
from convert.Spiders import Scrapers
from django.test import TestCase
from decimal import Decimal
from unittest.mock import patch
from convert.models import Provider, Currency, CurrencyRate

class TestScrapers(TestCase):
    def setUp(self):
        self.provider_tucambista = Provider.objects.create(
                                        name='Tucambista',
                                        website='http://www.tucambista.com',
                                        last_update='2022-01-01 12:00:00',
                                        logo=SimpleUploadedFile(
                                                "logo.jpg",
                                                b"file_content",
                                                content_type="image/jpeg"),
                                        transaction_time=3,
                                        is_neutral=False)
        self.currency_usd = Currency.objects.create(name='US Dollar',
                                                    code='USD')
        self.currency_pen = Currency.objects.create(name='Peruvian Sol',
                                                    code='PEN')

    @patch.object(Scrapers, 'get_soup_selenium')
    def test_scrape_tucambista(self, mock_get_soup_selenium):
        # Set up mock data
        mock_soup = '''<html><body>
                                <div class="flex flex-row items-center">
                                        <span>3.25</span></div>
                                <div class="flex flex-row items-center">
                        <               span>3.3769</span></div>
                    </body></html>'''
        mock_get_soup_selenium.return_value = BeautifulSoup(mock_soup,
                                                            'html.parser')
        # Run scraper method
        scrapers = Scrapers()
        scrapers.scrape_tucambista()
        print(CurrencyRate.objects.all)

        # Check that data was saved to database
        self.assertTrue(CurrencyRate.objects.filter(
                                provider=self.provider_tucambista,
                                currency_from=self.currency_usd,
                                currency_to=self.currency_pen,
                                exchange_rate='3.25').exists())

        self.assertTrue(CurrencyRate.objects.filter(
                                provider=self.provider_tucambista,
                                currency_from=self.currency_usd,
                                currency_to=self.currency_pen,
                                exchange_rate=
                                        str(
                                        round(1 / Decimal('3.3769'),
                                                4)))
                                        .exists()
```

```
@patch('requests.get')
def test_get_soup(self, mock_get):
    mock_response = mock_get.return_value
    mock_response.content = b'<html><head></head><body></body></html>'

    soup = Scrapers.get_soup(self.provider_tucambista)

    mock_get.assert_called_once_with(self.provider_tucambista.website)
    self.assertIsNotNone(soup)

def test_save_to_db(self):
    exchange_rate = Decimal('3.7525')
    Scrapers.save_to_db(self.provider_tucambista,
                        self.currency_usd,
                        self.currency_pen,
                        exchange_rate)
    rate = CurrencyRate.objects.get(provider=self.provider_tucambista,
                                    currency_from=self.currency_usd,
                                    currency_to=self.currency_pen,
                                    exchange_rate=exchange_rate)
    self.assertEqual(rate.provider, self.provider_tucambista)
    self.assertEqual(rate.currency_from, self.currency_usd)
    self.assertEqual(rate.currency_to, self.currency_pen)
    self.assertEqual(rate.exchange_rate, exchange_rate)
```