TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Kristjan Tõeväli 185993IADB

Kaasaegse remondi käsiraamatu rakenduse loomine NSVL-aegse autoremondi raamatu põhjal

Bakalaureusetöö

Juhendaja: Meelis Antoi

Magistrikraad

Tallinn 2023

Autorideklaratsioon

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

Autor: Kristjan Tõeväli

14.03.2023

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua NSVL-aegsete auto remondiraamatute põhjal kaasaegne remondi manuaali veebirakenduse prototüüp. Prototüübi nõuete määramiseks ja inspiratsiooni ammutamiseks analüüsitakse kahte tänapäevast remondi manuaali rakendust FordEtis ja ElsaWin. Antud rakendus on avatud lähtekoodiga ning julgustab ka teisi, kes tunnevad et antud teema neid huvitab, rakendusse panustama.

Töö eesmärk on arendada teenusepool ja kliendipool eraldatud, et vajadusel oleks võimalus tulevikus ühte või teist välja vahetada. Lisaks käsitletakse kasutatavaid tehnoloogiaid ning räägitakse rakenduse arhitektuurist.

Arenduse käigus luuakse veebirakenduse prototüüp, kus on võimalik sisestada otsitava masina andmed. Vastavalt andmetele kuvatakse kasutajale remondiks, hoolduseks ning keretöödeks vajalikud juhendid või sobilik elektriskeem vastavalt masina andmetele. Kasutaja saab liikuda viidete abil ühelt juhendilt teisele, kui hetkel vaadeldav töö eeldab mõne teise komponendi eemaldamist. Arendusprotsess on jagatud mitmesse peatükki, kus käsitletakse teenusepoolset ja kliendipoolset lahendust.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 55 leheküljel, 6 peatükki, 15 joonist, 1 tabelit.

Abstract

Creation of a Modern Repair Manual Application Based on the USSR-era Car Repair Book

The aim of this bachelor's thesis is to create a prototype of a web application for a modern repair manual based on Soviet-era car repair books. To determine the requirements and gather inspiration for the prototype, two contemporary repair manual applications, FordEtis and ElsaWin, will be analyzed. The given application is open source and encourages others interested in the topic to contribute as well.

The objective of the thesis is to develop a separate service side and client side, allowing the possibility of exchanging one or the other, if necessary, in the future. Additionally, the technologies used will be discussed, and the architecture of the application will be addressed.

During the development process, a prototype of the web application will be created, enabling the input of data for the desired machine. Based on the data, the user will be presented with the necessary instructions for repairs, maintenance, and bodywork, or a suitable electrical diagram corresponding to the machine's data. Users will be able to navigate from one guide to another using references, particularly when the current task requires the removal of another component. The development process is divided into several chapters, addressing the solution from the service side and the client side.

The thesis is written in Estonian and consists of 55 pages, including 6 chapters, 15 figures, and 1 table.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendusliides
Backend	Teenusepoolne keskkond
BSON	*Binary JSON*, binaarvorm lihtsate või keerukate andmestruktuuride esitamiseks.
CMS	*Content Management System*, sisuhaldustarkvara
CSS	*Cascading Style Sheets*, on küljendamisel kasutatav märgistuskeel
DRY	*Don't Repeat Yourself,* disainimuster, mis otsetõlkes tähendaks „Ära korda ennast"
DTO	*Data Transfer Object*, andmeedastusobjekt
ElsaWin	DVD'l olev rakendus, mis on pakub rakendust, mille läbi on võimalik otsida erinevaid remondi- ning hooldustööde juhiseid Škoda automargi sõidukite jaoks.
FordEtis	DVD'l olev rakendus, mis on pakub rakendust, mille läbi on võimalik otsida erinevaid remondi- ning hooldustööde juhiseid Ford automargi sõidukite jaoks.
HTML	*Hyper Text Markup Language*, veebilehe märgendamise keel
IDE	*Integrated Development Environment*, rakendus, mis on arendajatele koodi kirjutamiseks
JSON	*Javascript Object Notation*, Javascript'il põhinev andmevahetusvorming
LTS	*Long-term support*, tähenda, et toodet toetatakse uuendustega pikemat perioodi
MVC	*Model-View-Controller*, on tarkvara arhitektuurimuster, mis jagab tarkvararakenduse kolmeks omavahel seotud osaks
NoSQL	Mitterelatsiooniline alternatiiv klassikalisele relatsioonilisele andmebaasile.
NuGet	Paketihaldur, mida kasutatakse peamiselt. NET'i raamistikku kasutades kirjutatud tarkvara pakkimiseks ja levitamiseks.
ORM	*Object-Relational Mapping*, on põhimõte, mille eesmärk on objekt-orienteeritud koodi kaardistada ümber teise andmevormi
RenderFragment	Kasutatakse ASP.NET Core Blazor rakendustes ühe kasutajaliidese osa kirjeldamiseks.

REST	*Representational State Transfer*, veebiteenusega suhtlemise liidese arhitektuur
SOAP	*Simple Object Access Protocol*, veebiteenusega suhtlemise liidese protokoll
SPA	*Single Page Application*, üheleherakendus
STS	*Short-term support*, tähenda, et toodet toetatakse uuendustega lühikest perioodi
SQL	*Structured Query Language*, Struktuurpäringukeel
UI	*User Interface*, kasutajaliides
UX	*User Experience*, kasutajakogemus
Workshop manual	Töökoja käsiraamat
XML	*Extensible Markup Language*, dokumentide märgendamiskeel, mis on nii masin- kui ka inimestele loetav

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Täna päeval on hobimehaanikutel ligipääs erinevatele moodsatele remondi manuaali rakendustele nagu näiteks FordEtis ja ElsaWin. Nendest on kerge vaevaga võimalik leida tööde juhendeid koos selgitavate joonistega. Lisaks sellele on lihtne erinevate teemade vahel navigeerimine viite linkide abil.

Sellisele mugavusele pole aga ligipääsu neil hobimehaanikutel, kes tegelevad nõukogude aegsete masinatega. Kõik tolle aegsed remondi manuaalid on paber kandjal. Selline lahendus on arhailine ja kohmakas kasutada, kuna varuosade ja lisa informatsiooni leidmiseks pöördutakse ikkagi arvutisse või nutitelefoni, et otsida neid internetist.

Käesolev töö analüüsib hetkel turul olevaid remondi manuaali rakendusi, vaadates nende pakutavaid võimalusi, ülesehitust ja kasutaja sõbralikkust. Eelnevalt nimetatud probleemi lahenduseks luuakse NSVL-aegsest remondi raamatust ja tänapäevastest remondi manuaali rakendustest inspireerituna NSVL tehnika remondi manuaali veebirakendus, kus on võimalik otsida läbi rakenduse vaja minevaid varuosi.

Lõputöö autor on ise ka hobimehaanik, kellele meeldib tegeleda NSVL-aegsete masinate remondiga. Samuti tunneb töö autor mitmeid hobimehaanikuid, kes on samuti tundnud, et NSVL-aegsetele masinatele võiks eksisteerida selline rakendus, mis meenutaks tänapäevaseid remondi manuaali rakendusi. Sellest võib järeldada, et probleemi lahendamiseks tuleks luua veebirakendus, mis suudaks pakkuda võrdväärset kasutaja kogemust võrreldes tänapäevaste remondi manuaali rakendustega.

1.1 Metoodika

Käesoleva lõputöö jooksul selgitatakse esiteks lahti hetkel eksisteeriv probleem. Kirjeldatakse sarnaseid olemas olevaid lahendusi ja tuuakse välja nende puudused. Selle tulemusena pakutakse välja sobiv IT lahendus, mis jääks lõputöö skoopi. Kuid ei välistaks võimalusi tulevikus antud tööd edasi arendamast.

Lahenduse analüüsi osas tehakse kindlaks sobivad tehnoloogiad, mida kasutada. Põhjendatakse nende valikuid nii teenuse poolel kui ka kliendi poolel. Samuti ka arendusvahendite valikut.

Prototüübi valmimist on kirjeldatud mitmes osas, kus kirjeldatakse veebirakenduse serveripoolse, kliendipoolse lahenduse valmimist ning veebiämbliku arendust ja kasutajaliidese disaini.

2 Ülevaade probleemist

Workshop manual ehk töökoja käsiraamat on laialt levinud nimetus rakendustele, mille eesmärgiks on pakkuda erinevate auto mudelite jaoks juhendeid, kuidas mingit detaili eemaldada, parandada, testida ja paigaldada [1]. Erinevatel auto tootmis ettevõttetel on enda loodud rakendused, kust saab remondi töödeks vajalikku informatsiooni nende toodetud masinate jaoks.

Workshop manual nime all võidakse mõista kahte erinevat tüüpi manuale [1].

- Tehase hooldus manuaalid (*Factory Service Manuals)*, mis on loodud auto tootjate endi poolt. Need on üldiselt väga tehnilised sisaldades informatsiooni erinevate vedelike tasemete, pingutus momentide, diagnostika jooniste, remondi läbi viimise kohta, hooldus intervallide jne [1].

- Järelturu auto remondi manuaalid *(Aftermarket Car Repair Manuals)*, mis on mõeldud rohkem auto tavakasutajale ja on kirjutatud lihtsamas keeles andes nippe ja õpetusi. Nõuanded on suunatud lihtsamale auto hoolduse ja korras hoiule [1].

Tulenevalt lääne maailma ettevõtete järjepidevast tegutsemisest, on teatud osa nende vanemaid auto mudeleid kantud ka nende *Workshop manual* rakendustesse. NSVL'i kokku kukkumisega kaasnes ka selle autotööstuse langus. Sellest tulenevalt vähenesid rahalised võimalused luua selliseid rakendusi nende toodetud masinatele.

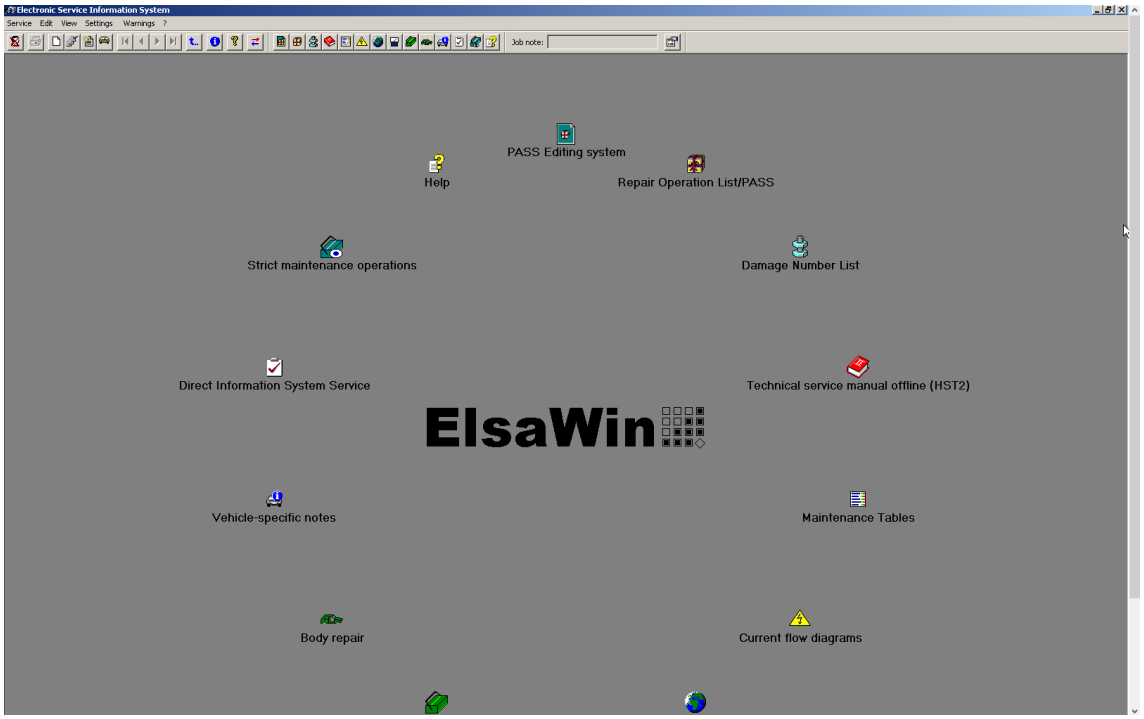
Probleem seisneb eelkõige selles, et inimesed kes on huvitatud NSVL-aegsete masinate remondist, siis neile ei eksisteeri mitte ühtegi rakendust hetkel, mis täidaks sarnast ülesannet nagu näiteks FordEtis. Sellest tulenevalt oleks vaja luua rakendus kuhu oleks võimalik hakata lisama NSVL-aegsete masinate remondi juhiseid. Antud lahendus annaks võimaluse koguda kokku ja talletada paberkandjal olevad remondi raamatud ühtekohta digitaalselt ning võimaldaks paljudele hobimehaanikutele lihtsamat ligipääsu autoremondiks vajaminevale informatsioonile.

2.1 Eksisteerivate lahenduste analüüs

NSVL-aegsete masinate remondiks ei eksisteeri hetkel samasuguseid *Workshop manual'e* nagu on neid erinevatele väljas pool NSVL'i toodetud auto markidele. See tõttu tuli analüüsida ja võrrelda hetkel eksisteerivaid mitte NSVL'i masinatele mõeldud remondi manuaale. Selleks on hetkel võimalik võrrelda ElsaWini ja FordEtist, kuna töö autoril on töö kirjutamise hetkel neile ligipääs.

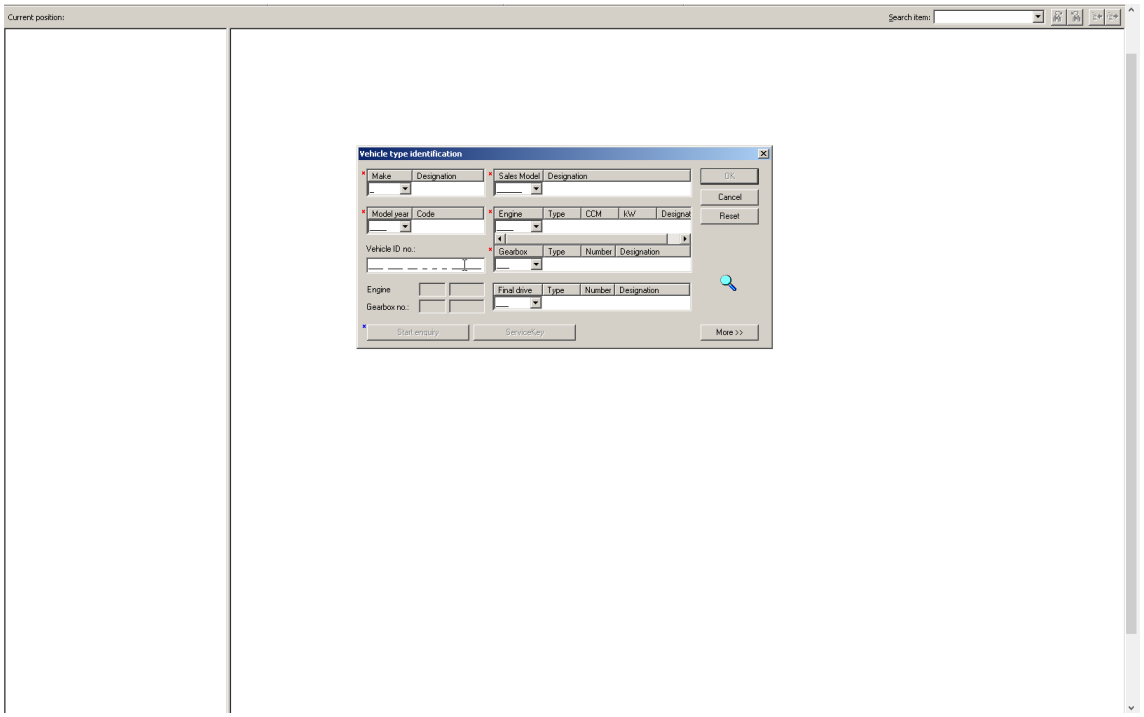
2.1.1 ElsaWin

ElsaWin on DVD põhine *Workshop manual*, mis toetab enamus Volkswagen AG masinaid, kuni aastani 2015 (Joonis 1). Praeguseks on Volkswagen AG läinud üle uuele rakendusele mida on võimalik tellimuse baasil rentida ja selle nimi on erWin.



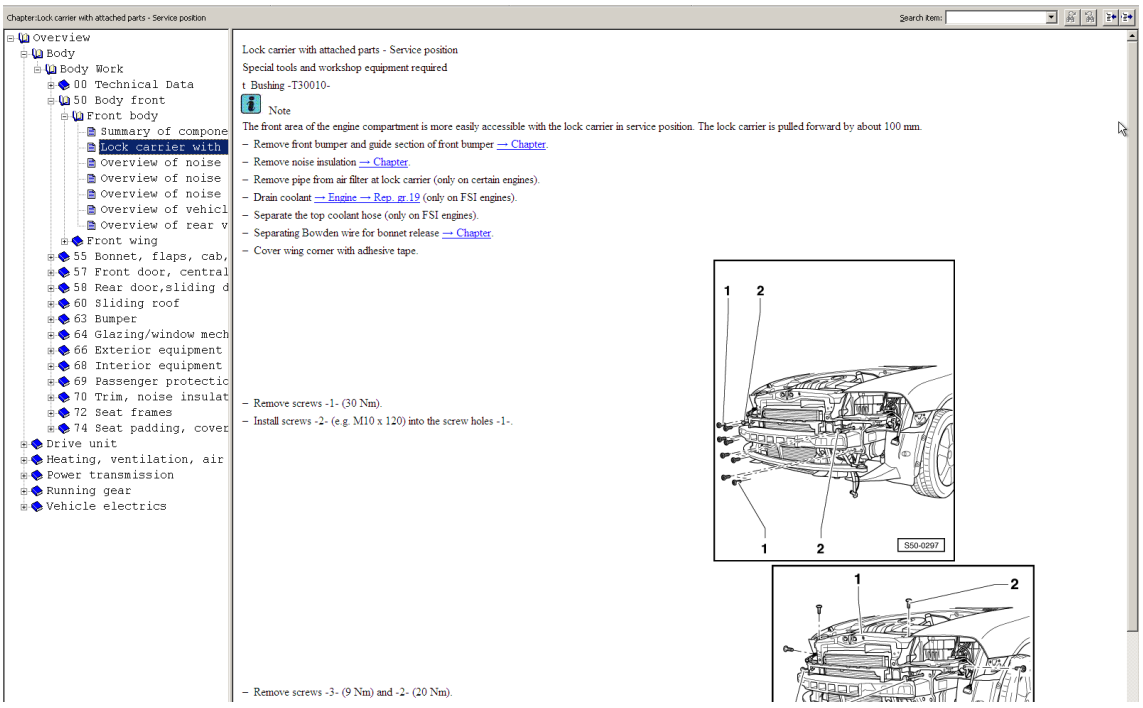
Joonis 1. ElsaWin esileht

ElsaWin on Windowsil töötav töölaua rakendus. Tänapäeva standardite järgi on UI (User interface) kujundus üsnagi iganenud ja eelmise millenniumi lõppu jäänud veebi lehekülgi meenutav. Kuigi informatsiooni otsimine on üsnagi lihtne. Esmalt sisestab kasutaja otsitava masina andmed (Joonis 2).



Joonis 2. Masina andmete sisestus

Teemad on jagatud eraldi menüüdeks ja alammenüüdeks. Kui kasutaja soovib informatsiooni elektriskeemi kohta sisestab ta otsitava auto kohta andmed ja vastavalt sellele pakub rakendus talle vasteid (Joonis 3).



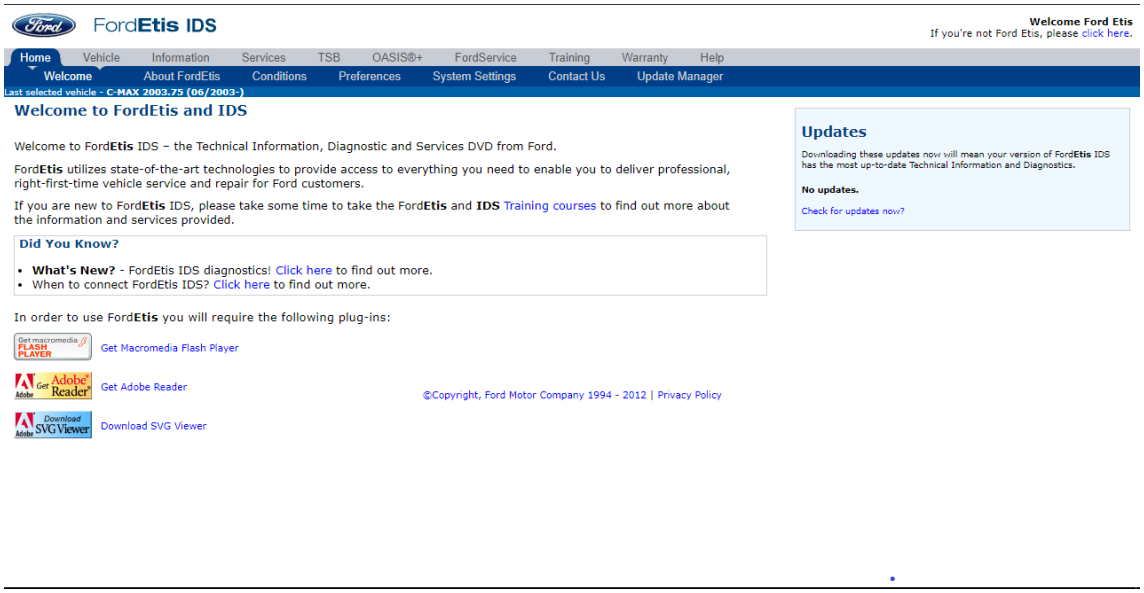
Joonis 3. Menüü valiku riba ja konkreetne juhend

Kõige suurem puudus ElsaWin'i puhul seisneks selle vananenud kasutajaliidese lahenduses. Üldine UX'i (User experience) lahendus on üsnagi lihtne ja otse kohene. Kõigepealt sisestatakse enda otsitava masina andmed. Sealt edasi saadakse valida alustuseks suurem üldisem grupeeriv kategooria ning järjest tase tasemelt liikuda täpsemalt otsitava diagrammi või remondi juhendi poole.

Positiivseks näiteks võib välja tuua remondi juhendite menüüs ikoonide muutused, mis väljendavad kas tegemist on kogumiga või leheküljega. Lisaks on erinevus avatud kogumiku ja mitte avatud kogumiku ikoonis.

2.1.2 FordEtis

FordEtis tehase *Workshop manual* (Joonis 4) rakendus, mis on loodud euroopa turule toodetavate Ford marki sõidukite hoolduseks ja remondiks. See on samuti DVD põhine *Workshop manual*, mida enam edasi ei kasutata. Ford on läinud samuti üle tellimuse alusel sarnase teenuse rentimisele.

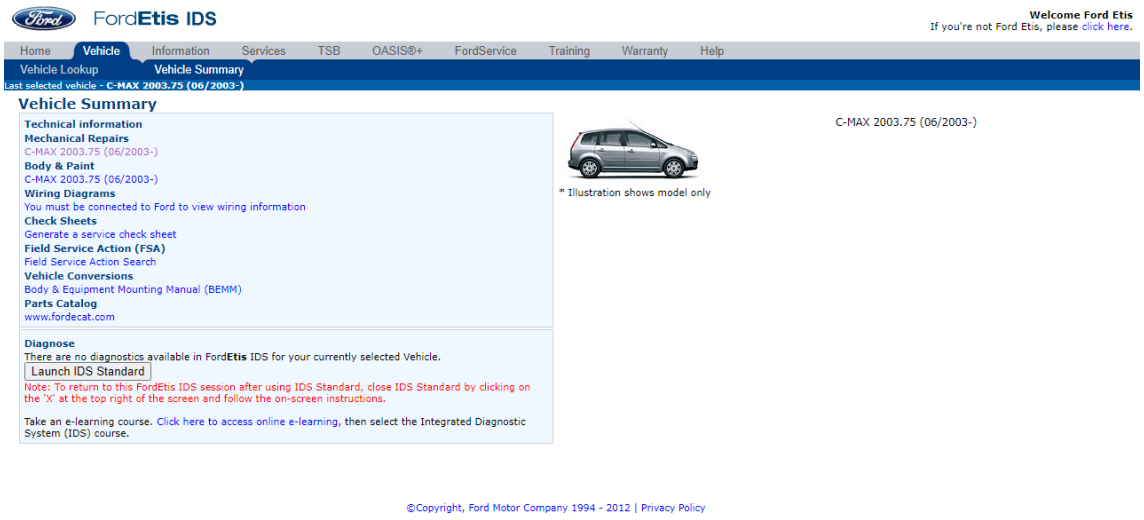


Joonis 4. FordEtis IDS esileht

FordEtis on Windowsil töötav töölaua rakendus. Kasutaja kogemus on üsnagi hea omades selget eristust valimata ja valitud paneelide vahel. Kasutaja valib auto andmed, et leida juhendeid (Joonis 5).

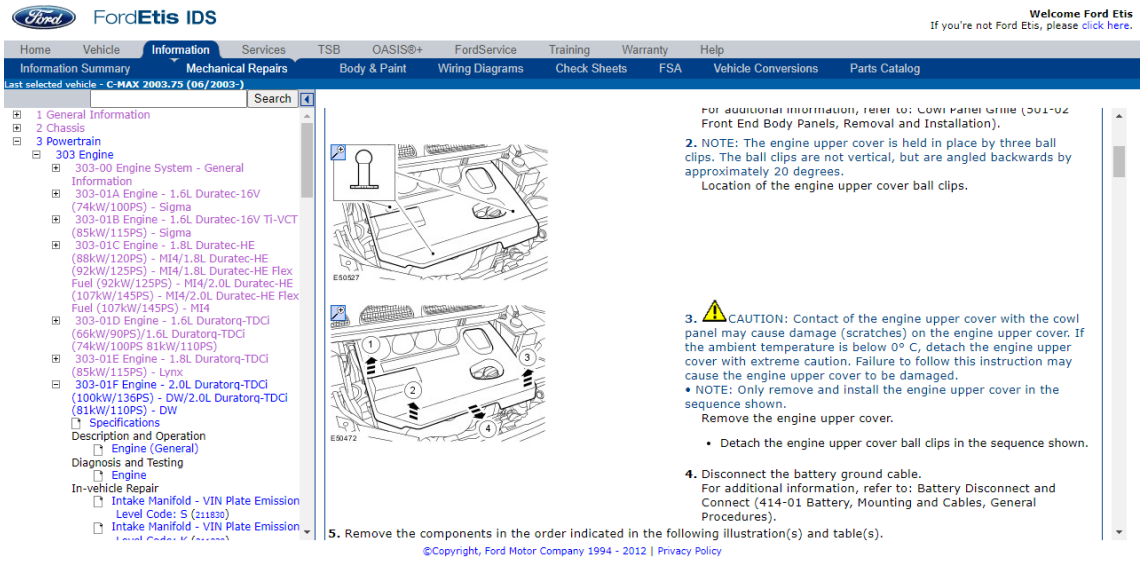
Joonis 5. Soovitava sõiduki otsing

Järgnevalt näidatakse kasutajale erinevaid teemasid, mille järgi saab kasutaja otsustada, kuhu täpsemalt ta edasi soovib suunduda (Joonis 6).



Joonis 6. Kategooriad infoga valitud sõiduki kohta.

Pärast soovitus teema valimist kuvatakse menüü, mida mööda navigeerides saab otsida sobiva juhendi (Joonis 7).



Joonis 7. Valitud masina remondi juhis.

Sarnaselt ElsaWin'iga on vastava masina kohta menüü, kust on võimalik navigeerida otsitava juhendi juurde. Osade juhendite juures on viited teistele juhenditele, mis käsitlevad teatud probleemi täpsemalt või on antud tööga seotud.

Kasutades FordEtis't ilmneb probleem, et valides auto mudeli ei saa täpsustada antud masina mootorit ega käigukasti tüüpi. Sellest tulenevalt on juhendi otsingu menüü

sisutihe, kuna valikus on kõik mootorid ja käigukasti tüübid, mis sellele mudelile paigaldada võimalik on. See muudab konkreetse mootori või käigukasti otsingu näpuga järjeajamiseks, et soovitud teemat ülesse leida. Autori arvates on parem lähenemine ElsaWin'is, kus täpsustatakse alguses mark, mootori ja käigukasti tüüp. Nii kuvatakse juhendite menüüs ainult valitud masina kohta juhendeid.

FordEtis'e kasutajaliidese lahenduste poolest võib positiivseks välja tuua juhendi menüü, kus on erinevalt märgitud kategooriad, gruppide pealkirjad ja konkreetsed juhendid. Teiseks on üldmenüü, kust saab valida erinevate lehekülgede vahel. Vastavalt valitud leheküljele kuvatakse sinisele alammenüüribale valitud lehekülje alamleheküljed.

2.2 Loodava lahenduse skoop

Kuna ei eksisteeri ühtegi sellist rakendust, mis oleks spetsiifiliselt mõeldud NSVL'is toodetud masinatele, siis tuli võtta analüüsi aluseks väljas pool NSVL'i toodetud auto markidele loodud *Workshop manual* rakendused.

Kasutaja mugavust arvestades oleks kõige mõistlikum luua selline rakendus veebirakendusena, omades sarnast võimekust nagu Windowsi töölaua rakendustel Elsawin ja FordEtis. Nii ei olene kasutaja riistvarast, kas ta saab kasutada rakendust või mitte. Samuti võimaldab see kasutajatel pääseda veebirakendusele ligi ka nutitelefoniga. Windowsi baasil töötava töölaua rakenduse lahendus sunniks kasutaja installeerima rakenduse enda arvutisse. Lisaks poleks võimalik seda rakendust kasutada ühegi teise seadmega, mis ei kasuta operatsiooni süsteemiks Windowsi. See tõttu on valitud lahenduseks luua töökoja käsiraamatu rakendus veebirakendusena.

Antud rakenduse loomise eesmärgiks ei ole tulu teenimine vaid pakkuda hobimehaanikutele, kes tegelevad NSVL-aegsete masinatega võimalust kasutada rakendust, mis oma olemuselt sarnaneb teiste töökoja remondi manuaali rakendustega. Pakkudes neist inspireeritud funktsioone, kuid paremat kasutus mugavust.

3 Loodava remondi manuaali veebirakenduse analüüs

Antud peatükis käiakse läbi erinevad punktid, kus kirjeldatakse veebirakenduse arenduseks võimalikke sobivaid tehnoloogiaid. Selle tulemusena tehakse otsus, milliseid tehnoloogiaid kasutades veebirakendust looma hakatakse.

3.1 Nõuete määramine

Nõuete paika panemisel on arvestatud, et probleemi käsitletakse hetkel Eesti lõikes. Veebirakenduse prototüübi kasutajale kuvatavaks keeleks on Eesti keel.

Nõuete määramiseks on tehtud kasutajalood, kus põhi ja ainu rolliks praeguses skoobis on hobimehaanik/mehaanik (edaspidi kasutaja).

3.1.1 Mittefunktsionaalsed nõuded

Kasutaja-põhised mittefunktsionaalsed nõuded:

- Kasutajana soovin aru saada rakenduse sisust, kui oskan rääkida eesti keelt.
- Kasutajana soovin kasutada rakendust ükskõik millise laialt levinud brauseriga.

3.1.2 Funktsionaalsed nõuded

Kasutaja-põhised funktsionaalsed nõuded:

- Kasutajana soovin saada sisestada masina andmeid, et leida antud masina jaoks juhiseid.
- Kasutajana soovin, et erinevad teemad oleks jagatud selgelt aru saadavatesse kategooriatesse
- Kasutajana soovin, et remondi juhiste juures oleks illustreerivad joonised.
- Kasutajana soovin omavahel seotud juhendite vahel liikuda viite linkide abil.
- Kasutajana soovin, et minu poolt sisestatud masina andmeid mäletatakse seni kuni brauseri aken on avatud.

- Kasutajana soovin, et remondi juhise juures oleks võimalik vaadata, millised e-poed müüvad planeeritava remondi töö jaoks varuosi.

3.2 Tehnoloogiate valik

Veebirakenduse arendamiseks on tehnoloogiate valik üpris suur [2]. Antud lõputöö raames võetakse võrdlusesse laialt levinumad tehnoloogiad ja pinud. Selles töös ei kasutata ega vaadelda CMS (Content Management System), kuna sellised lahendused suunavad lahendust kindlas suunas ning ei anna lahendusele piisavat paindlikust. CMS on loodud lihtsamate standard lahenduste jaoks nagu e-poed, foorumid, blogid jne [3].

Kaasaegse veebirakenduse arendusel on tähtis eraldada kood oma ette loogika blokkideks, kus iga blokk vastutab kindla ülesande eest. See annab lahendusele paindlikkuse ja taaskasutus võimaluse [4]. Selline lähenemine võimaldab teenust ja klient lahendusi eraldi seisvalt arendada, et nad ei oleks üksteisest sõltuvad. Kui on vaja välja vahetada teenuse või kliendipool ei pea hakkama terve arendusega otsast peale.

Tulenevalt eelnevalt valitud lahendusest hoida teenusepoole ja kliendipool eraldi seisvatena on lisaks vaja valida lahendus, kuidas API (Application Programming Interface) nende vahel välja nägema hakkab. REST (Representational State Transfer) on hulk arhitektuurilisi piiranguid, mis määravad päringute käsitlemis korra [5]. Teiseks lahenduseks oleks SOAP (Simple Object Access Protocol), mis lahendab REST'iga sama probleemi aga erineval viisil [6]. REST'i eeliseks on suurem valik andme edastus formaate [5]. Arhitektuurilised piirangud on valikulised ehk neid saab võtta kasutusele siis, kui vajadus tekkib ning lisaks on kiirem ja võimaldab rakenduse kohandatavust vastavalt vajadustele [5]. SOAP'i eeliseks on kindel ja protokolli juhitud andmevahetus, mistõttu on SOAP kasutusel peamiselt veel kohtades, kus tegeletakse veebis toimuvate tehingutega [6].

Järgnevalt tuuakse välja mõningad kriteeriumid, mille alusel hinnatakse teenusepoolseid ja kliendipoolseid tehnoloogiaid. Samuti programmeerimiskeeli ja raamistikke.

- Kogemus – tehnoloogia valimisel on oluline lõputöö autori kogemus tehnoloogiatega. Kuigi töö autor soovib oma teadmisi laiendada tuleb arvestada antud töö raames piiratud ajaga ning uues tehnoloogia õppimine ajamahukas ülesanne.

- Dokumenteeritus – valiku tegemisel ühe või teise tehnoloogia kasuks on mõistlikum pöörduda tehnoloogiate poole, mis on hästi dokumenteeritud ja omavad head tuge, et probleemile lahenduse leidmine oleks lihtne.

- Paindlikus – kasutusele võetavad tehnoloogiad peaksid olema piisavalt paindlikud, et neid saaks veebirakenduse prototüübi arenduse ajal omavahel ühilduma panna.

3.2.1 Teenusepoolse programmeerimiskeele valik

Programmeerimiskeele valikust sõltub, millist raamistiku hakatakse kasutama. Tuntumateks teenusepoolsete programmeerimiskeelte hulka kuuluvad:

- Java – üldkasutatav objekt orienteeritud programmeerimiskeel, mida kasutatakse laialdaselt mitmetel platvormidel. JVM ehk Java Virtual Machine on interpretaator, mis teisendab Java bit koodi masin loetavaks koodiks [7].

- Python – on objekt orienteeritud, aspektipõhine programmeerimiskeel, mis on kasutusel keeruliste matemaatiliste probleemide lahendamisel, andmeanalüüsis ja tehisintellekti arendamisel. Aspektipõhisus tähendab, et programmi tööajal määratakse suure hulga viidetega õige tüübi definitsioon [8].

- C# - põhiliselt .NET platvormil kasutatav universaalne objekt orienteeritud programmeerimiskeel, mis kuulub C programmeerimiskeelte perekonda jagades nendega mitmeid sarnaseid karakteristikuid. On üks populaarsemaid programmeerimiskeeli ning omab suurt ja tugevat kogukonda [9]. Sellest tulenevalt omab head dokumentatsiooni .

- JavaScript – on kergekaaluline programmeerimiskeel, mida kasutatakse põhiliselt veebi arenduses, et luua dünaamilisi lahendusi veebi lehekülgede, rakenduste ja mängude jaoks [10].

- Ruby – objekt orienteeritud üldkasutatav programmeerimiskeel, mille fookuses on lihtsus ja produktiivsus. Ruby on kasutusel mitmetel aladel nagu arvutiteaduses, veebiarenduses ja ka andmete analüüsis [11].

- PHP – on laialdaselt kasutatud vabavaraline skriptimis ja programmeerimiskeel, mida on võimalik kombineerida HTML'iga. Kuna tegemist on serveri poolse keelega, siis serveri poolel genereeritud kood saadetakse kliendi poolele HTML kujul [12].

Kasutades võrdluspunktideks programmeerimiskeele õppimiskeerukust ja lõputöö autori kogemust nendega on koostatud tabel (Tabel 1) nende võrdlemiseks. Kuna tegemist on tuntumate teenusepoolsete programmeerimiskeeltega on kõigi võrreldavate keelte dokumenteeritus tase hea.

Tabel 1. Programmeerimiskeelte võrdlus.

Programmeerimiskeel	Autori kogemus	Hinnatav õppimiskeerukus
Java	Hea	Keskmine [13] [14]
Python	Rahuldav	Madal [13] [14]
C#	Väga hea	Keskmine [9]
JavaScript	Rahuldav	Madal [13] [14]
Ruby	Puudub	Madal [11]
PHP	Kehv	Madal [14]

Võttes arvesse, et lõputöö kirjutamiseks on piiratud aeg on mõistlik valida programmeerimiskeel, millega töö autoril on kõige rohkem kogemust. Antud keeled,

milles omab töö autor kõige rohkem kogemust on C# ja Java. Nagu eelevalt öeldud on lõputöö kirjutamiseks piiratud aeg ja ükski teine keel ei tõuse märgatavalt teistest esile, et põhjendada aja kulu õppimaks uut keelt.

C# üheks suureks eeliseks on .NET maailma tugi, millest räägitakse täpsemalt järgnevas peatükis. Lisaks on NuGet pakettide haldur, mis võimaldab erinevate lisa pakettide kasutamist arenduses. Paljud paketid on tasuta kasutatavad ning vabavaralised.

3.2.2 Teenusepoolsed raamistikud

Teenusepoolsete tehnoloogiate seas on mitmeid valikuid. Võttes vaatluse alla enim kasutatavad raamistikud, et nende seast valida välja sobiv käesolevas töös püstitatud nõuete saavutamiseks.

- Express.js – minimalistliku lähenemisega raamistik, mida on lihtne kasutada ja omab head jõudlust. Raamistik pakub ka mitmeid funktsioone ja tööriistu, et lihtsustada arendustööd [15].

- Laravel – avatud lähtekoodiga raamistik, mis on kirjutatud PHP's. Üheks hinnatumaks funktsiooniks on sisse ehitatud autentimis süsteem, mille abil on hõlpsasti võimalik luua kasutaja registreerimis ja sisselogimis vorme. Samuti on Laravel'is võimalik kasutada ORM'i (Object Relational Mapper) nimega Eloquent, mis vabastab arendaja kohustusest ise kirjutada komplekksseid SQL (Structured Query Language) päringuid [15].

- .NET – on avatud lähtekoodiga raamistik, mis toetab C#, F# ja Visual Basic programmeerimiskeeli [16]. Antud raamistik omab väga head dokumentatsiooni ja võimaldab väga kiiret veebirakenduste arendust [15].

- Spring Boot – avatud lähtekoodiga Java baasil loodud raamistik. Võimaldab arendajal kasutada erinevaid tööriistu sisseehitatud, et lihtsustada ja kiirendada rakenduse arendust [15].

- Django – Python'i jaoks loodud avatud lähtekoodiga raamistik. Raamistiku eesmärgiks on taas kasutatavus, lihtsasti ühendatavus, kiire arendus ning DRY

(Don’t Repeat Yourself). Lisaks omab erinevaid tööriistu nii autentimiseks kui ka ORM’i [15].

- Flask – samuti Python’i baasil loodud mikro raamistik. Minimalistlik lähenemine annab kasutajale võimaluse ise valida, milliseid lisa teeke kasutada rakenduse arendamiseks [15].

- Ruby on Rails – on avatud lähtekoodiga raamistik kirjutatud Ruby programmeerimiskeeles. See raamistik on tuntud oma lähenemise poolest ehk konventsioon üle konfiguratsiooni. Antud lähenemine võimaldab minimaalse konfigureerimisega luua töötav rakendus. Võimaldab samuti lisada erinevaid teeke lisa funktsionaalsuse lisamiseks [15].

- Next.js – on React’i baasil loodud raamistik, mida kasutatakse, et luua funktsionaalselt rikkalikke rakendusi. [15].

Arvestades eelnevas peatükis valitud programmeerimiskeelega on tark valida kasutatavaks raamistikuks .NET. Esiteks on see mõeldud C# keele kasutamiseks ja teiseks on lõputöö autoril selle raamistikuga kõige rohkem kogemust võrreldes teiste raamistikega. .NET omab väga head dokumenteeritusse taset ja pakub laialdast tööriistade valikut, et arendada moderne ja kiire veebirakendus.

3.2.3 Kliendipoolne tehnoloogia

Kliendipoolse tehnoloogia valimisel tuleb arvesse võtta, et suurem osa raamistikke kasutab JavaScript’i või selle peale ehitatud TypeScript’i. Seetõttu tuleb raamistiku valimisel arvestada, ka selle raamistiku programmeerimiskeelega. Välja on toodud mõned tuntumad raamistikud ja paketid.

- React – on avatud lähtekoodiga JavaScript’i pakett. React võimaldab kasutada komponentide põhilist lähenemist, kus arendaja saab luua neid taaskasutatavana [17].

- Vue – on samuti avatud lähtekoodiga raamistik, mis kasutab sarnaselt React’ile komponentide põhilist lähenemist. Lisaks on see deklaratiivne, mis tähendab, et HTML muutub vastavalt taustal arvutatud JavaScript’i tulemusest [18].

- Bootstrap – on kasutajaliidese arendamise raamistik, mis on avatud lähtekoodiga. Kombineerides HTML, CSS ja JavaScript'i on võimalik luua rikkalikke ja kasutaja sõbralikke veebirakendusi [19].

- MudBlazor – on samuti sarnaselt Bootstrap'ile mõeldud aitamaks arendajat luua visuaalselt ilusaid ja funktsionaalselt rikkalikke veebirakendusi. MudBalzor on mõeldud kasutamiseks Blazor'il põhinevatel rakendustel [20].

- Blazor – on Microsoft poolt loodud avatud lähtekoodiga raamistik, mis töötab .NET raamistikul kasutades C# programmeerimiskeelt. Blazor'i eripäraks on võimalus luua veebirakendus ilma ühtegi rida JavaScripti kirjutamatta. Samuti võimaldab Blazor luua veebirakendust, mis töötab nii kilendipoolel, kui ka serveripoolel [21].

Kõik välja toodud lahendused on avatud lähtekoodiga lahendused. Neist kõik sobivad kiireks arenduseks ja omavad head dokumenteeritus taset. Võrreldes teistega seisab Blazor välja, kuna on ainukene raamistik võrdluses, kelle eesmärgiks on pakkuda kliendipoolset lahendust ilma JavaScript'i kirjutamata.

Töö autor omab kõige rohkem kogemust Blazor'i raamistikuga. Nagu eelnevas peatükis osutus valituks .NET raamistik sobib Blazor sellega hästi. Blazor omab samuti head dokumenteeritus taset. Blazor'i kasutamise suureks eeliseks on võimalus jätta JavaScript'i kasutamine ära. Muidugi on see alati võimalik. Arvestades lõputöö eesmärki on kasulik võtta kasutusele ka MudBlazor ja BootStrap. Kasutades mõlemat on võimalik kombineerida parimad võimalused mõlemast.

3.2.4 Andmebaasi valik

Andmebaasid jagunevad mitmetesse kategooriatesse milleks on relatsioonilised, objekt orienteeritud, dokument orienteeritud, võti-väärtus ja puustruktuuriga andmebaasid [22]. Kuna antud lõputöös luuakse veebirakendust, siis võetakse valikusse relatsioonilised ja dokument-orienteeritud andmebaasid. Samuti on võrdluseks võetud andmebaasid avatud lähekoodiga.

Relatsioonilised andmebaasides hoiustatakse andmeid eelnevalt defineeritud tabelites, mis on omavahel seotud defineeritud viidetega [23]. Võrdluseks on välja toodud tuntumad relatsioonilised andmebaasid:

- PostgreSQL – on avatud lähtekoodiga andmebaas, mille kasutamine on tasuta. Avatud lähtekoodiga andmebaasi kohta pakub väga suurt funktsionaalsust nagu näiteks toetab JSON tüüpi info vastuvõtmist. See annab võimaluse luua andmebaasi võrgustikke ka NoSQL tüüpi andmebaasidega. Üheks negatiivseks aspektiks on PostgreSQL'i suund saavutada maksimaalne ühilduvus, mille arvelt kannatab operatsioonide kiirus [24].

- Sqlite – Mälus töötav relatsiooniline andmebaas, mis omab head lugemis ja kirjutamis kiirust [25]. Samuti on lihtsa süntaksiga ja kerge õppida, kuid andmebaasi maksimaalne suurus on limiteeritud 2 GB (Giga Byte) peale [26].

- MySQL – võimekuse poolest sarnasel tasemel nagu teised suuremad andmebaasid. Kui rakendus mille jaoks seda andmebaasi kasutatakse luuakse kinnise lähtekoodiga kommertslikul eesmärgil tuleb osta litsents MySQL andmebaasi kasutamiseks [27].

Dokument orienteeritud andmebaaside puhul hoiustatakse andmeid JSON[1], BSON[2] või XML[3] formaadis . Sellised andmebaasid annavad paindlikuse arenduse käigus, kui andmete formaadid pole kindlaks määratud. Antud lähenemine muudaks liigutatava info palju lähedasemaks objektidele, mida kasutatakse rakenduse töös [22]. Tuntumateks dokument orienteeritud andmebaasideks on:

[1] *Javascript Object Notation*, Javascript'il põhinev andmevahetusvorming

[2] *Binary JSON*, binaarvorm lihtsate või keerukate andmestruktuuride esitamiseks.

[3] *Extensible Markup Language*, dokumentide märgendamiskeel, mis on nii masin- kui ka inimestele loetav

- MongoDB – on skaleeritav ja väga paindlik, kuna info talletamiseks pole vaja defineerida malle, mille järgi neid hoitakse. Samuti on süntaks lihtsam õppida, kui SQL süntaksit. MongoDB miinusteks on piiratud dokumenti suurus milleks on 16 MB (Mega Byte) ja ühe dokumendi sügavuseks on maksimaalset 100 kihti. Lisaks puudub lihtne viis tegeleda duubeldunud infoga, mis omakorda suurendab mälu mahu kasutamist [28].

- Couchbase – omab sarnaseid aspekte võrreldes MongoDB'ga, kuid konfiguratsiooni võimalused on paremad. Samuti info päringud on kiiremad ja lihtsamini loodavad tänu N1QL'ile. Negatiivseks punktiks on vähesem programmeerimiskeelte tugi võrreldes MongoDB'ga [29].

3.2.5 IDE ehk integreeritud arenduskeskkonna valik

Integreeritud arenduskeskkonna (IDE) valikul vaadeldakse IDE'sid, mis toetavad C# programmeerimiskeelt ning eelnevates peatükkides tehtud valikutest. Tuntumateks ja suurima kasutajas konnaga IDE'deks on Microsofti Visual Studio ja Visual Studio Code (VS Code) ning JetBrains'i Rider [30].

Võrreldes neid kolme IDE't, siis Rider on ainukene kes ei paku tasuta *Community* verisooni [30]. Tasuta ligipääsu Rider'ile saavad õpilased ja õppejõud tõestades enda õpinguid või tööd õppeasutuses [31]. Nii Visual Studio kui ka Rider on mõlemad võimekad IDE'd, mis on mõeldud professionaalse arendaja tööriistadeks. VS Code on avatud lähtekoodiga ja tasuta [30]. VS Code'i üheks miinuseks võib pidada selle abitööriistade puuduse esmasel paigaldamisel.

Koodihalduseks vaadeldakse ainult tasuta võimalustega, kuid levinuid koodihaldus keskkondi.

GitHub on Microsofti poolt omatud koodihaldus platvorm. Pakub ainult tasuta võimalust avalikele repositooriumitele [32]. On üks populaarsemaid koodihaldus platvorme sisaldades ca 69 miljonit projekti [32]. Tänu suurele populaarsusele on GitHubi kasutades lihtne leida probleemidele lahendusi.

GitLab sai alguse 2011. aastal. See on avatud lähtekoodiga ja pakub suurel hulgal võimalusi tasuta arenduseks [32]. Pakub tasuta pakette nii avatud, kui ka suletud repositooriumite jaoks [32].

BitBucket on Atlassiani poolt omatud veebipõhine repositoorium [33]. Pakub samuti tasulisi pakette, kui ka tasuta paketti [33]. BitBucketi üheks eeliseks on Jira, mis on Atlassiani toode. See on üks paremaid arendustöö organiseerimiseks loodud tööriistu [33].

3.2.6 Veebiämbliku tehnoloogia valik

Tulenevalt antud lõputöö nõudest on vaja valida sobiv programmeerimiskeel, mille baasil luua antud rakenduse jaoks *Web Crawler* (veebiämblik). Kuna lõputöö autoril ei ole laialdast kogemust veebiämblike loomisega peab võrdlusesse võetava programmeerimiskeele jaoks olema piisavalt materjale ja näiteid. Järgnevalt on välja toodud kasutatumad programmeerimiskeeled selle jaoks.

- Python – on kõige tuntum programmeerimiskeel, mida kasutatakse veebiämblike loomiseks. Python'il on suur hulk toetavaid teeke, mis on mõeldud veebiämblike ehitamiseks. Lisaks sellele on Python'il lihtne süntaks [32].

- Ruby – üheks eeliseks Ruby puhul on Nokogiri, mida tihti nimetatakse lihtsamaks tööriistaks, kui need mis Python'is võimalik kasutada on. Ruby omab ka suurepärast testimis raamistikku, mille kaudu on lihtne testida ka veebiämblikke [32].

- JavaScript – Node.js'i abil on võimalik ehitada veebiämlikke, mille eeliseks eelnimetatutega on võimekus hankida dünaamiliselt muutuvatelt veebi lehtedelt. Node.js'i baasil tehtud veebiämblikud sobivad lihtsamateks töödeks ja ei ole sobilikud suure hulga andmete töötlemiseks [32].

- C++ - on üldkasutatav programmeerimiskeel ja seetõttu on sellega võimalik luua veebiämblik. Kuigi veebiämblikud selles programmeerimiskeeles pole eriti levinud. Üheks põhi probleemiks on võimetus tegeleda dünaamiliselt loodavate veebi lehtedega. Sobib lihtsamate veebiämblike jaoks [32].

- Java – on üks suurima kasutajaskonnaga programmeerimiskeeli. Java'l on häid teeke, mis on loodud veebiämblike loomiseks. Mõned kasutatumad on JSoup, HTMLUnit ja Jaunt. Kuigi jääb oma võimekuse poolest alla eelnimetud Python'i ja Ruby's loodavate veebiämblike võimalustele [32].

Python omab laia valikut teekidest ja raamistikest, mille abil on võimalik arendada veebiämblikku. Üheks populaaremaks teegiks, et eraldada kokku korjatud andmetest kasulikku informatsiooni on BeautifulSoup. Lisaks on mõistlik võtta kasutusele raamistik, mis annab tööriistad kiiremaks veebiämbliku arenduseks. Selleks raamistikuks oleks Scrapy. Scrapy on raamistik, mis on mõeldud veebilehtede ronimiseks ja neilt andmete kogumiseks, mida saab kasutada suure hulgal kasulikeks eesmärkideks nagu informatsiooni kogumine, selle töötlemine või arhiveerimine [33].

Python'is loodud veebämbliku kasutamiseks C# koodis on mõistlik kasutusele võtta IronPython. IronPython võimaldab .NET keeltes lihtsat ligipääsu Python'ile [34]. Läbi selle on lihtne kasutada Python'is kirjutatud skripte.

Antud võrdlusest võib järeldada, et sobivamateks kandidaatideks oleks Python ja Ruby. Mõlemad omavad häid teeke, mille abil on võimalik luua keerukaid veebiämblike. Lõputöö autoril ei ole varasemat kokkupuudet Ruby programmeerimiskeelega. Python'i baasil on lõputöö autor loonud algelisi veebiämblike varem, mistõttu töö autori arvates on aja kulu mõistes kasulikum kasutada Python'it.

3.3 Veebirakenduse disain

Antud veebirakenduse põhi ülesandeks on remondi manuaali digitaalne esitamine. Sellest tulenevalt peab olema kasutaja kogemus väga hea. Kasutaja jaoks peavad olema erinevad valikud ja teemade kategoriseerimine lihtsasti arusaadav.

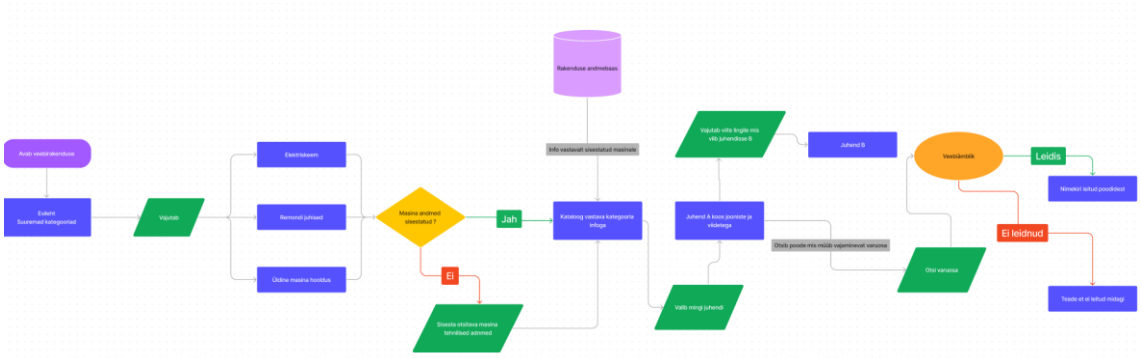
Kindlasti peab juhtima kasutaja tähelepanu lisa funktsionaalsusele selle olemas olul. Kasutaja peab arusaama, kas sinna tuleb vajutada, lohistada või piisab lihtsalt hiire peale viimisest [35]. See tähendab näiteks kõikide navigeerimis linkide selgelt eristamist ülejäänud tekstist lehel, et kasutaja mõistaks selle asja funktsiooni.

Selleks, et paremini analüüsida loodava veebirakenduse disaini on see jagatud järgnevateks alampeatükkideks:

- Kasutajakogemuse disain
- *Mood board*'i ehk meeleolupaneeli loomine
- Planeeritav rakenduse disain Figma baasil

3.3.1 Kasutajakogemuse disain

Vastavalt kasutajalugudele on võimalik kokku panna kasutaja teekond veebirakenduses (Joonis 8).



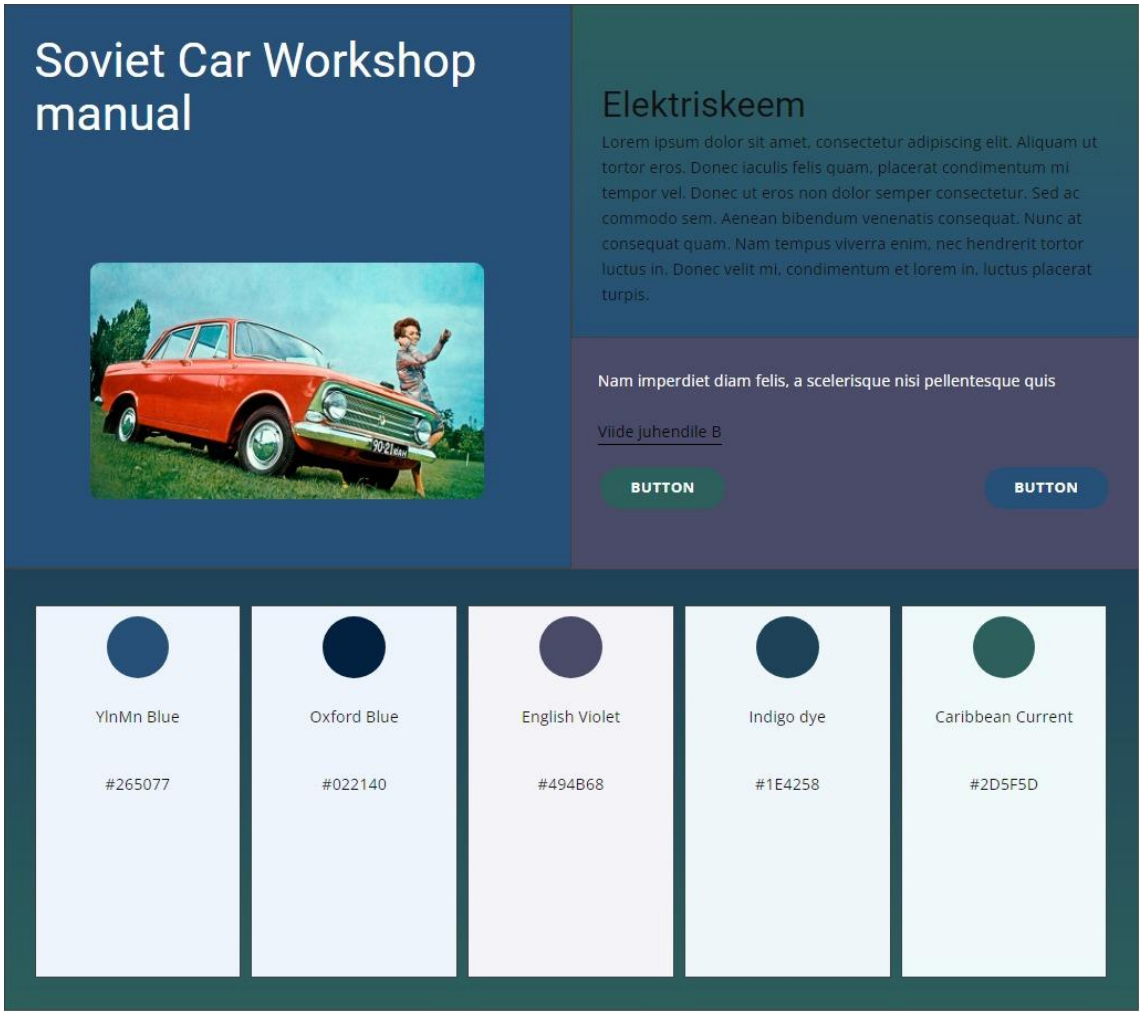
Joonis 8. Kasutajakogemuse diagramm

Joonisel on kujutatud kasutaja teekonda läbi rakenduse. Esmasel lehel kuvatakse kasutajale üldiseid kategooriaid. Pärast kategooria valikut kontrollitakse, kas antud kasutaja on juba sisestanud antud masina andmed mille kohta infot otsitakse. Kui mitte, siis kasutaja sisestab selle. Kasutajal on alati võimalus muuta masina andmeid uute päringute jaoks. Pärast seda kuvatakse kasutajale masina põhist infot. Olenevalt kategooria valikust on kasutajal võimalus otsida veebiämbliku abil poode, mis pakuvad otsitavaid varuosi. Lisaks on kasutajal võimalik liikuda kiirelt omavahel seostuvate tööde juhendite vahel viidete abil.

3.3.2 Meeleolupaneel

Meeleolupaneeli loomine annab parema ülevaate ja ettekujutuse loodava rakenduse disainist. Antud meeleolupaneel (Joonis 9) on loodud nicepage [36] abil. Selle abil on

lihtsam panna paika värvi kombinatsioon, kirjatüübid, nupud ning muud elemendid. Värvi kombinatsioonid on loodud coolors [37] veebilehe abil.



Joonis 9. Soviet Car Workshop manual veebirakenduse meeleolupaneel

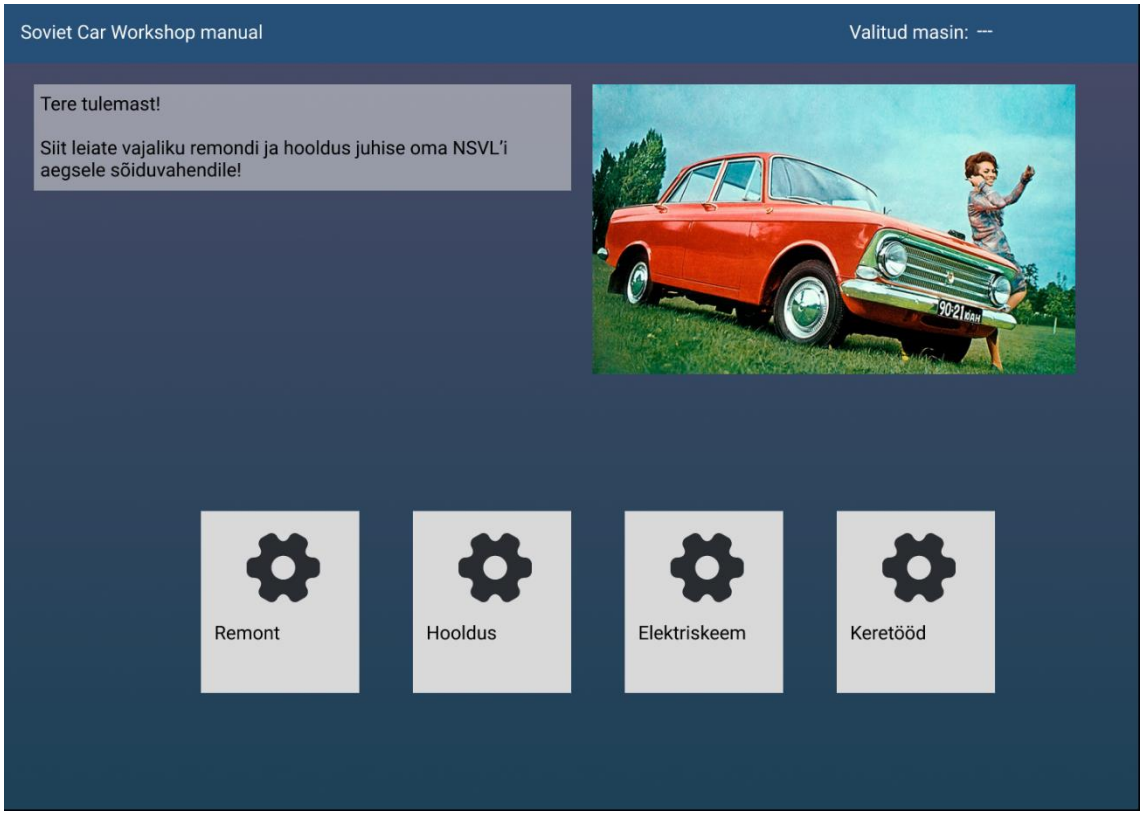
Loodud paneeli puhul sai valitud värvide kombinatsioon, mis oleks võimalikult rahulik, kuid samas silma hakkav. Kuna tegemist on veebirakendusega mille peamiseks ülesandeks on remondi manuaalide info edastamine peavad kirjatüübid olema tagasi hoidlikud, et neid oleks lihtne lugeda.

3.3.3 Kasutajakogemuse disain Figma baasil

Figma on brauseri põhine, tasuta versiooni pakkuv UI disainimis ja graafikate loomis rakendus. Selle abil on võimalik luua rakenduste disaini prototüüpe, sotsiaalmeedia postituste disainimine ja kõike muud nende vahele jäävat [38].

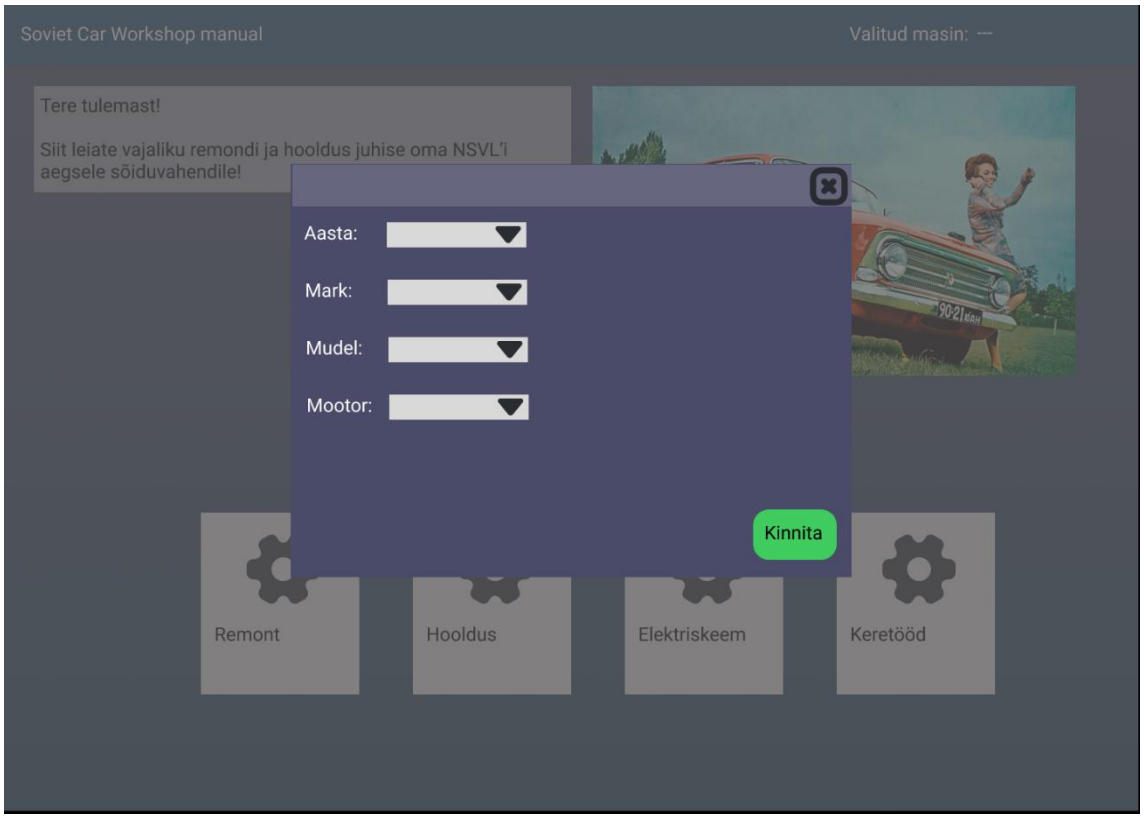
Veebirakenduse visuaalse disaini prototüübi loomine aitab hoida hilisemas arendustöös aega kokku. Prototüübi loomine võimaldab juba varakult panna paika arenduseks funktsionaalsed nõuded ning kuidas elemendid paiknema võiksid hakata.

Esileht kujutab endast tervitus ja infopaneeli ning pakub valikuks neli võimalikku teemat, kuhu edasi liikuda (Joonis 10).



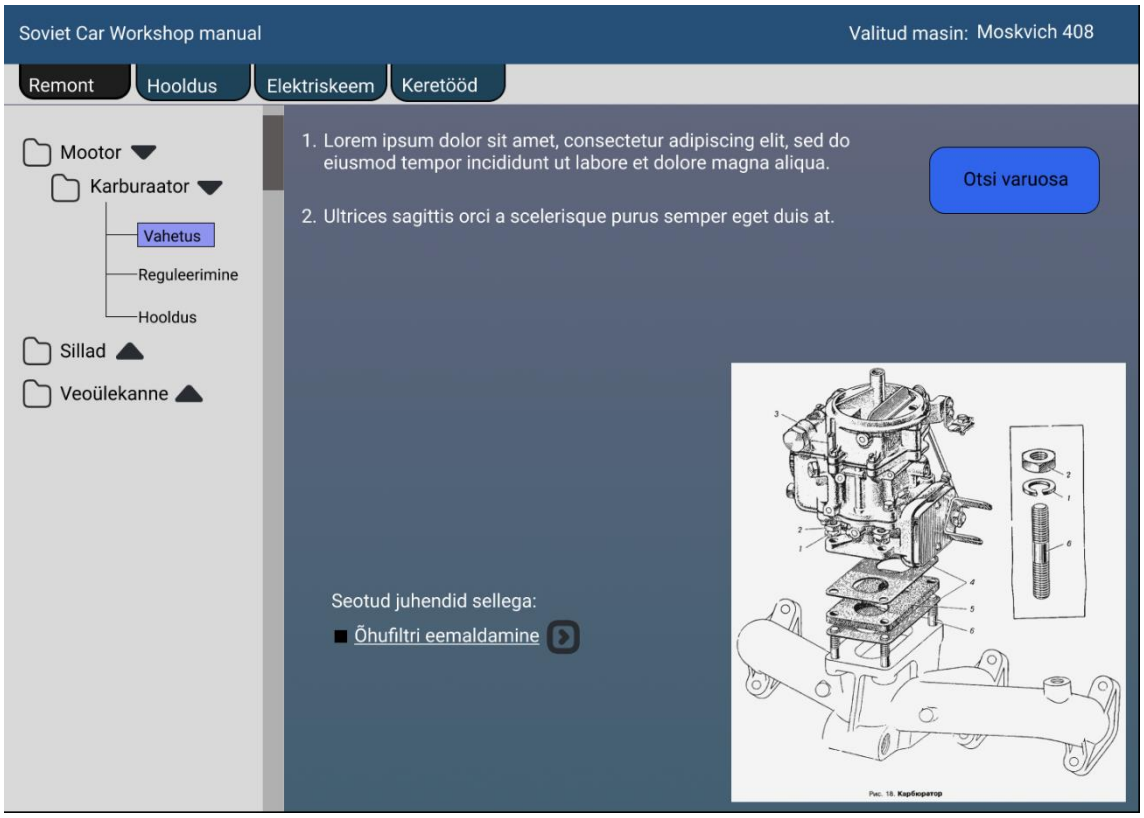
Joonis 10. Veebirakenduse kodulehe disaini plaan

Järgnevalt vajutades ühele neist valikutest kuvatakse aken. Selles aknas valitakse otsitava masina andmed ja kinnitatakse enda valik (Joonis 11). Selline aken ei ilmu kasutajale, kui ta on korra juba sisestanud enda masina andmed. Enda masina valikut saab kasutaja alati eemaldada akna paremal ülevas nurgas asuvast nupust.



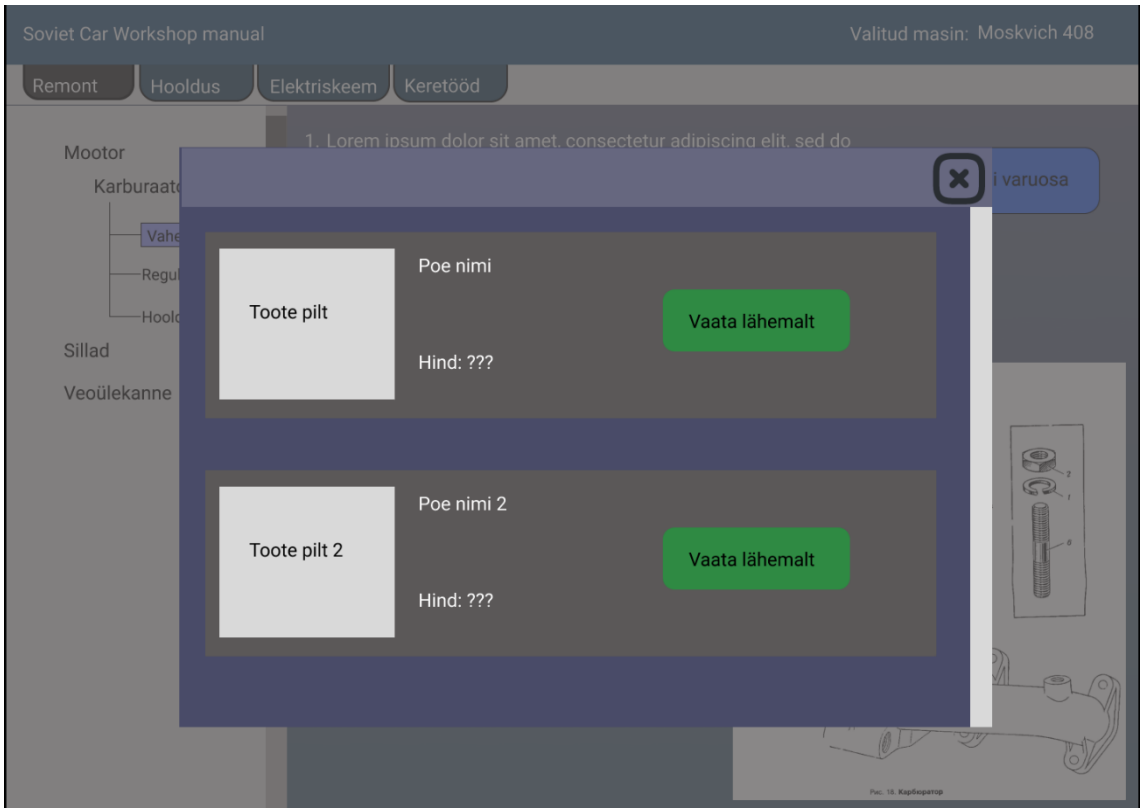
Joonis 11. Masina andmete sisestamine konkreetse kategooria avamiseks

Pärast masina andmete sisestamist kuvatakse kasutajale menüü, mida mööda navigeerides on võimalik jõuda otsitava juhendini (Joonis 12). Juhendi juures on lisaks juhistele kuvatud ka navigatsiooni link või lingid, mis viivad antud juhendiga seotud juhendite juurde. Samuti on juhendite juures ka nupp, mille abil saab otsida veebipoode, kus müüakse käesoleva töö jaoks vaja minevat varuosa.



Joonis 12. Konkreetse remondi juhendi vaade

Kui kasutaja soovib vaadata, millised e-poed hetkel pakuvad juhendis kujutatud remondi töö jaoks varuosi, siis kuvatakse kasutajale selline aken. Selles aknas näidatakse toote pilti, hinda, poe nime ja viidet sellele tootele antud poes (Joonis 13).



Joonis 13. Varuosade otsingu tulemus

3.4 Analüüsi kokkuvõte

Antud peatükis kirjeldati kasutajate kohta käivaid mittefunktsionaalseid ja funktsionaalseid nõudeid.

Analüüsi käigus osutus valituks lähenemine, et loodav rakendus peab eksisteerima veebiteenuse kujul, mis kasutab REST API põhimõtteid. Sellest tulenevalt eksisteeriks iseseisvalt kliendiliides, mis suhtleb läbi API taga rakendusega. Antud lähenemine võimaldab ühte või teist poolt tulevikus välja vahetada ilma teise poole tööd segamata.

Teenusepoolseks programmeerimiskeeleks osutus C#, kuna lõputöö autoril on selle keelega kõige rohkem kogemust, C# omab väga head dokumenteeritus taset ja korralikke arenduskeskkondi. Raamistikuks teenusepoole valiti C# jaoks .NET, mis on väga võimekas raamistik ja sobib antud lõputöö veebirakenduse arendamiseks. Kliendipoolseks lahenduseks valiti Blazor, mis võimaldab luua kliendipoolseid

lahendusi ilma ühtegi rida Javascripti kirjutamata. Koodi kirjutamiseks kasutatakse JetBrains Rider'i ja andmebaasi valikuks on avatud lähtekoodiga PostgreSQL. Koodihaldus keskkonnaks on GitLab, kuna pakub tasuta paketis kõige rohkem võimalusi ning avatud lähtekoodiga on võimalik teistel huvilistel ka arenduses kaasa aidata.

4 Remondi manuaali veebirakenduse prototüübi arendus

Antud veebirakenduse prototüübi lahendus jagati kolme peatükki, milleks on veebiteenuse lahendus, kliendirakenduse lahendus, veebiämbliku arendus. Nendes peatükkides on lähemalt kirjeldatud prototüübi arendusega seotud lahendusi.

4.1 Veebiteenuse lahendus

Veebirakenduse *backend* ehk teenus vastutab äriloogika ning andmetele ligipääsu eest. See on ehitatud .NET raamistiku peale ning kasutab andmebaasi, kuhu on salvestatud klientrakenduse jaoks vajalik äriloogika informatsioon. Järgnevates peatükkides on täpsemalt kirjeldatud veebiteenuse arenduse tehnoloogiaid ja etappe.

4.1.1 .NET rakenduse arhitektuur

.NET raamistiku tööriistad võimaldavad luua kiirelt ja vähese vaevaga nii MVC (Model-View Controller) ning API kontrollerid. Selle eelduseks on domeeniobjektide loomine, mille baasil võimaldab .NET aspnet-codegenerator kiirelt ehitada erineva otstarbega faile [39]. Konfiguratsiooni ja teotavate teenuste määramine on samuti mugavalt juhitav Program.cs ja appsettings.json failide.

Projekt on struktureeritud silmas pidades taaskasutamise eesmärki, kus baasfunktsionaalsus on eraldi seisvates projektides ja nende nimed algavad Base prefiksiga. Antud rakenduse on jagatud järgnevalt nimetatud projektide kogumitesse. Iga kihi vahele on loodud DTO - Data Transfer Objects ehk andmeedastusobjektid.

- Base – baasfunktsionaalsus, mida saab erinevate projektide vahel jagada.
- Domain – domeeniobjekte kirjeldavad klassid
- DAL – Data Access Layer ehk andmete pärimis kiht mille läbi käib suhtlus andmebaasi ja ülejäänud rakenduse vahel.
- BLL – Business Logic Layer ehk äriloogika kiht. Siin asuvad kõik rakenduse äriloogikaga seotud klassid ja tegevused.

- WebWsApp – Siin asuvad REST API kontrollerid ja API konfiguratsiooni failid.

Nagu eelnevalt mainitud saab vähese vaevaga luua REST API kontrollereid kasutades .NET'i sisse ehitatud koodi generaatorit. Kuna kliendirakenduses puudub vajadus andmete kustutamiseks, uuendamiseks ja uute lisamiseks, siis on ainukesed API päringud, mida teha saab, GET päringud.

4.1.2 Andmebaasi kasutamine rakenduses

Analüüsi tulemusena selgus, et antud projekti arenduses kasutatakse PostgreSQL andmebaasi. Selleks et kasutada .NET Entity Framework'i PostgreSQL andmebaasiga on vaja NuGet paketti haldurist alla laadida pakett, mis võimaldab PostgreSQL andmebaasi kasutusele võtta. Entity Framework'is on objekt-relatsiooniline kaardistamine, mis võimaldab .NET'i arendamiseks kasutavatel arendajatel töötada ilma ise kirjutamata suurt osa tavaliselt andmete pärimiseks vaja minevat koodi [40].

Arendusprotsess on andmebaasiga mugavam, kui kasutada Dockerit. See lubab lihtsasti panna püsti rakenduses defineeritud andmete põhjal andmebaasi. Lisaks võimaldab see ka teistel arendajatel, kes soovivad projekti arendusega ka alustada vähese vaevaga luua enda kohaliku andmebaasi.

Arenduse käigus toimub andmebaasis paratamatult muudatusi. Andmebaasi muutuste ehk migratsioonide loomine ja rakendamine on lihtsasti tehtav läbi .NET enda. Migratsioonide rakendamist saab rakenduse käivitamis konfiguratsiooni faili kirja panna. Kõik uued migratsiooni rakendatakse järgmisel rakenduse käivitusel. Vastavad migratsiooni failid genereeritakse kasuta *Migrations* ning projekti kus asub rakenduse andmebaasi konteksti kirjeldav klass.

4.1.3 Turvalisus

.NET tervik lahendust ja tööriistu, et lisada loodavale rakendusele ligipääsu piirav autentimis ja autoriseerimise kiht. Autentimine tähendab, et kontrollitakse kas kasutaja on see, kes ta väidab ennast olevat ning autoriseerimise all mõistetakse seda, kas kasutajal on teatud tegevused lubatud [41].

Tervik lahendus .NET'ist turvalisuse tagamiseks on rolli põhilist autoriseerimist. Selle kasutusele võtt loodavasse rakendusse on väga lihtne. Kui luuakse uut käivitatavat projekti Rideris, siis märgitakse enne projekti ehitamist, et projekt ehitatakse rolli põhise autoriseerimisega. Antud prototüübi lahenduses võetakse rolli põhine autoriseerimine kasutusele, et piirata ligipääsu teenusepoolse rakenduse administreerimis lahendusele.

4.2 Kliendirakenduse lahendus

Kliendirakendus on arendatud kasutades ASP.NET core Blazorit. Blazor on loodud võimaluseks arendajatele, kes ei soovi Javascript'i kasutada, et luua veebirakendusi. Blazor pakub samuti valikut, kas lasta serveri poolel teha kogu UI komponentide ehitamine või kasutades *Webassembly't* viia UI komponentide ehitamine ning jooksutamine brauserisse. Antud rakenduse arenduses on valitud *Webassembly* kasuks, sest see vähendab serveri poolset töömahtu kasutades ära võimalust jätta UI komponentide jooksutamise brauseri hooleks [42].

Kuna klientrakenduses ei ole ettenähtud kasutajal konto loomist ega sisse logimist pole klientrakenduses loodud lahendusi, mis tegeleks turvalisusega. Turvalisus lahendusi pole arendatud, kuna läbi klientrakenduse ei ole võimalik taga rakendusse midagi saata, sest taga rakenduse API ei toeta POST, UPDATE ega DELETE päringuid.

4.2.1 Bootsrap'i ja MudBlazor'i kasutus

Bootstrap on loodud Twitteri poolt ja esimene avalik versioon nägi ilmavalgust 2011. aastal. Bootstrap raamistiku eesmärk on suunata arendajaid kasutama uuemaid CSS (Cascading Style Sheets) atribuute, vähendama sõltuvusi ning tagama selle läbi sarnase kasutajakogemuse [43]. Bootstrapi lisamiseks projekti tuleb vastav Bootstrapi versioon alla laadida läbi NuGet paketti halduri. Bootstrap sisaldab mitmeid CSS stiilifaile ja Javascript faile.

MudBlazor on Blazor lehekülgede arendamiseks mõeldud komponentide teek. MudBlazor'i pakutavad komponendi on üldkasutatavad ja muudavad arenduse kiiremaks ning mugavamaks [44]. MudBlazor'i kasutamiseks projektis tuleb see kõige pealt allalaadida soovitud projekti juurde. Lisaks komponentidele pakub MudBlazor ka

tööriista, millega on lihtne soovi korral disainida enda rakenduse välimust. Panna komponentide kujunduse paika, muutes nende taustade värve, lisades varje jne.

Valides Bootstrap'i raamistiku ja MusBlazor'i komponentide teegi saab kasutada juba valmis komponente ja neile soovikorral lisada Bootstrap'iga uusi stiile. MudBlazor lisab enda komponentide stiliseerimiseks ka CSS faili. Sellest võib tekkida Bootstrap'i ja MudBlazor'i stiilide vahel konflikt. Antud rakenduses valiti hetkel lähenemine lahendada konfliktid lisades töö autori enda loodud stiliseering, mis kirjutab üle vaike stiliseeringu. Miks selline lahendus viis osutus valituks oli põhjus, et stiili konflikte on väga vähe. Lisaks hoiab antud lähenemine aega kokku, kuna keerukamate komponentide loomine on ajamahukas. Vastasel juhul, kui lülitada välja MudBlazor'i komponentide vaike stiliseering peaks looma stiliseeringu kõigile komponentidele ning see oleks aega nõudev tegevus. Autori arvates on kiirem ja lihtsam lahendada üksikutes kohtades konflikt manuaalselt, kui hakata välja lülitama terve teegi stiliseeringut.

4.2.2 Blazor – *Single Page Application* (SPA)

Single Page Application ehk lühendina SPA on veebirakenduse lahendus, mis kasutab ainult ühte HTML lehekülge. Lehekülje info uuendamise ja muutmise eest vastutab Javascript ning brauser ei pea tegema teenuse poole uut pöördumist [45].

SPA on praegusel hetkel populaarseim lähenemine. Antud lähenemine pakub kasutajale sujuvamat veebilehe kasutamist, suuremat kiirust ja paremat reageerimis aega [46]. Kuna leheküljel andmete uuendamine ei vaja terve lehekülje uuesti laadimist, siis saab kasutaja jätkata lehekülje kasutamist ning sama ajal uuendatakse teatud komponendis kuvatav informatsioon.

Blazor'i baasil rakenduste arendamise keskne põhimõte on luua komponente. Komponentide paindlikus tähendab, et komponent võib olla terve lehekülg või lahter, mis kuulub lehekülje sisse, kuhu kasutaja saab sisestada andmeid [47]. Komponendid koosnevad C# ja HTML'ist koodist [48]. HTML'is defineeritakse, kuidas komponendi sisu kuvatakse UI's ning C# kood taustal tegeleb vajaliku info töötlemise, pärimise ja edastamisega HTML'i. Komponentide ehitamisel on mõistlik lähtuda loogikast, et loodud komponendid oleksid üldkasutatavad ja erinevate projektide vahel vajadusel

jagatavad. Selline lähenemine aitab vähendada koodi kordusi ja kiirendab tulevikus uute projektide arendust.

4.2.3 Blazor veebirakenduse struktuur

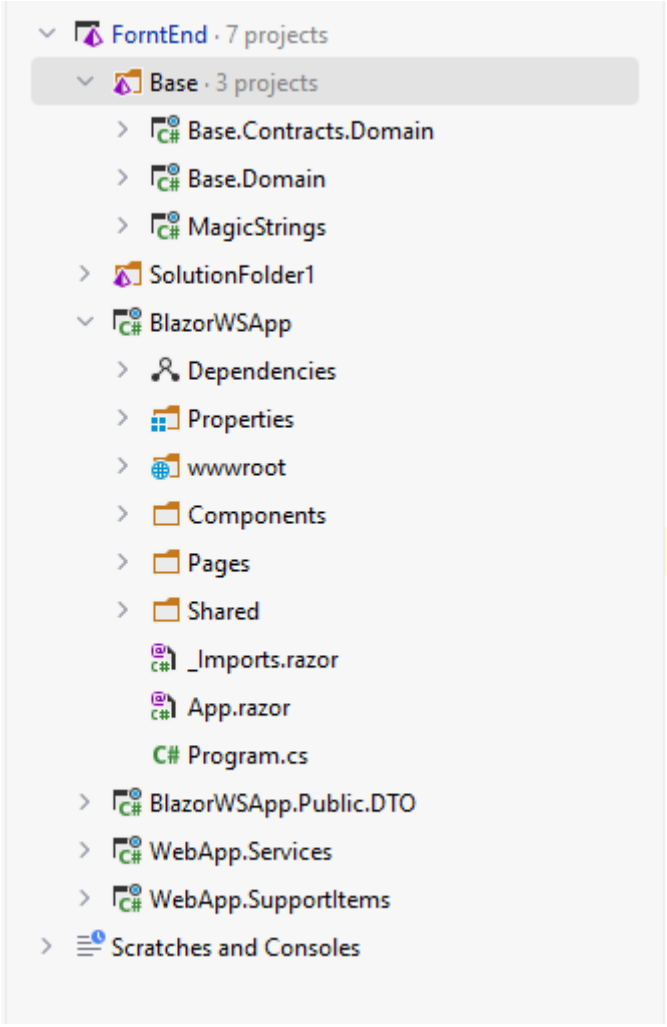
Blazori baasprojekt sisaldab faile nagu index.html, Program.cs, App.razor, launchSettings.json ja muid razor komponente nagu näiteks MainLayout.razor. See razor fail (Joonis 14) kirjeldab lehekülje põhi komponendid nagu päis ja artikli osa. Artikli ossa muutuja *Body* väärtustatakse *RenderFragment* tüüpi väärtusega, mis kujutab endast seda informatsiooni, mida antud leheküljel kuvada.

```
@inherits LayoutComponentBase
@inject NavigationManager MyNavigationManager
@using BlazorWSApp.Components
<MudThemeProvider/>
<MudDialogProvider/>
<MudSnackbarProvider/>

<div class="page" style="...">

    <main class="body-style" style="...">
        <header>
            <HeaderBar></HeaderBar>
        </header>
        <article class="content px-4 ">
            @Body
        </article>
    </main>
</div>
```

Joonis 14. MainLayout.razor fail

Rakenduse arenedes lisanduvad rakendusele juurde uusi komponente, klasse ja projekte (Joonis 15). Hetkel on komponentide kataloog asetatud käivitatava Blazori klientrakenduse projekti sisse. Põhjus miks komponendid pole eraldi seisvasse projekti viidud, mis oleks jagatav erinevate projektide vahel on see, et praegusel hetkel pole vajadust viia neid eraldi seisvasse projekti, kuna pole kasutajat neile. Nii viisi hoiab

projekti struktuuri ka lihtsamana, kuna antud rakenduse jaoks vajalikud komponendid on selle juures ja lihtsalt leitavad.



Joonis 15. Blazor WebAssembly rakenduse struktuur

4.3 Veebiämbliku arendus

Veebiämblikud võivad mööda veebi ringi roomamisega täita mitmeid ülesandeid. Levinumaks neist on veebilehtede indekseerimine, et neid oleks võimalik otsingumootoreid kasutades ülesse leida. Samuti võivad nad koguda informatsiooni, mida kuvatakse veebilehtedel ja selle informatsiooni põhjal luuakse uut sisu. Näiteks võib tuua erinevad toodete hindade võrdlus platvormid.

4.3.1 Veebiämbliku loomine Scrapy põhjal

Antud rakenduses hakkab loodav veebiämblik külastama ette antud nimekirja autovaruosade veebilehti, millelt kogutakse kokku andmeid müüdavate varuosade kohta. Kogutud info puhastatakse ja talletatakse andmebaasi. Sellisel lähenemisel ei pea kasutaja ootama pikalt pärast varuosade otsimis nupule vajutust, et talle vajalikku informatsiooni kuvatakse. Lisaks hoiab see ära asjatuid veebiämbliku ronimisi.

Scrapy erinevaid ämblike baas klasse, millest edasi arendada vajaliku keerukusega ämblikke. Antud rakenduses valiti baasiks *CrawlSpider*, mis on kõige levinum ämbliku tüüp veebis ronimiseks. Ämblikule on võimalik seada ette mitmeid reegleid nagu näiteks kinni pidamine veebilehtede robots.txt failist ja selles kirjeldatust. Määrata lubatud domeeni ning linkide kaudu maksimaalne liikumise sügavus. Ämbliku käivitamisel on võimalik seda ka välja sõelutud informatsiooni tagastus kuju nagu JSON, JSON lines, CSV ja XML [49]. Käesolevas rakenduses eelistatakse JSON'i.

5 Hinnang loodud veebirakendusele

Arendatud veebirakenduse prototüüpi võib hinnata saavutatud funktsionaalsuste baasil. Lisaks võib hinnangu aluseks võtta visuaalsele lahendusele, kasutatud tehnoloogiate uudsusele ja edasiarendatavusele. Tehnoloogiate uudsuse puhul võib märkida, et veebirakenduste arenduses kasutatavad tehnoloogiad arenevad kiiresti ja arendajale tuleb kasuks osata ja olla kursis uute võimaluste ning lahendustega. Blazor on uudne lahendus .NET maailmas klientrakenduste arendamiseks pakkudes arendajale võimalust luua klientrakendus ilma ühtegi rida Javascripti kirjutamata.

5.1 Rakenduse saavutatud kasutatavus

Prototüübi puhul said valmis kõik mittefunktsionaalsed ja enamus funktsionaalseid nõudeid. Klientrakenduse arendamine kulges üldjoontes sujuvalt, kuna töö autoril on eelnevalt kogemust Blazori kasutamisega ning lisaks sellele lihtsustab Blazori kasutamist vajadus tunda ainult C#. Samas jättes võimaluse arendajale lisada ja ise luua Javascripti teeke, mida kasutada Blazor rakenduses.

Üheks saavutamata funktsionaalsuseks jäi hetkel kasutajal võimalus vaadata remondi juhendi juurest võimalikke e-poode, kus müüakse remondi töö jaoks vaja minevat varuosa. Antud lahenduse saavutamisel osutus kõige suuremaks katsumuseks töö autori vähene kogemus veebämblike loomisel ja nende töö juhtimisel. Lisaks raskendas lahenduse valimist asjaolu, et osad veebilehed kasutavad sisu kuvamiseks Javascripti meetodeid ning sellised meetodid ei pruugi käivituda. Nende meetodite mitte käivitumise pärast jääb osa infost veebiämblikele nähtamatuks ja seetõttu on raske teatavate e-poodide tootevalikutest saada head ülevaadet.

Klientrakenduse teised funktsionaalsed nõuded on saavutatud ja prototüüp rakendus on kasutatav. Kasutaja saab valida nelja suurema teema vahel ning järgnevalt valida otsitava masina andmed. Pärast seda kuvatakse kasutajale menüü, kust kasutaja saab otsida talle vajaliku teema ning selle avada. Avades kuvatakse kogu informatsioon ning jooniste olemas olul ka joonised ning viited teistele juhistele.

Prototüüpi ei ole hetkel optimeeritud mobiili peal kasutamiseks. Blazor rakenduses on suudetud koos tööle panna Bootstrap ja MudBlazor, mis mõlemad pakuvad rakenduse ülest stiliseerimis faile. Sellest tulenemata on konfliktide hulk väike ja tekkinud konfliktid manuaalselt loodud CSS failidega parandatud.

5.2 Kasutatud tehnoloogiate uudsus

Prototüüp veebirakenduse arendamisel oli tehnoloogiate valimise põhimõte kasutada arenduses tuntumaid tehnoloogiaid ning arvestada autori kogemustega nendega. Blazor on klientrakenduse arendamiseks üks uudsemaid tehnoloogiad, mille esimene versioon lasti avalikkuse ette 2018. aastal. Teenusepoolse rakenduse arendamiseks kasutati .NET raamistikku versiooniga 7, mis on STS (Standard Term Support). Töö autor oleks eelistanud kasutada LTS (Long Term Support) versiooni .NET'ist, kuid .NET 8.0 on alles arenduses ning tuleb oodata selle avaldamist. Pärast .NET 8.0 avaldamist saab rakenduse kolida uuele versioonile üle.

Veebiämbliku arendamiseks kasutati Pythonit versiooniga 3.11.2, mis on töö kirjutamise hetkel kõige hilisem versioon. Arendatud veebiämbliku jaoks kasutati ka Scrapy raamistikku, mis on üks tuntuimaid veebist informatsiooni kokku kraapimis raamistikke.

Kokkuvõttes võib hinnata, et arenduses on kasutatud nii üpris uudseid lahendusi nagu Blazor ning juba tuntud ja ennast tõestanud .NET raamistik ja Pythoni põhine Scrapy raamistik. Kasutatud tehnoloogiate sobivust ja aktuaalsust võib hinnata kõrgelt.

5.3 Võimalused edasi arendamiseks

Tähtsaim prioriteet on edasi arendusel valmis saada hetkel pooleli olev veebiämbliku lahendus. Esimene versioon sellest korjaks oma andmeid ainult kindlaks määratud lehekülgedelt, et oleks võimalik kindlustada kasutajale tervikliku informatsiooni kuvamine rakenduses.

Järgnevalt on vaja luua teenusepoolsele rakendusele administraatori tööriistad, et oleks võimalik lisada, muuta ja kustutada remondi juhendeid masinate kohta. Samuti peaks

olema võimalik piltide lisamisel kindlaks määrata pildi paiknemise leheküljel. Salvestatud muudatustest peaks tagavaraks salvestama ka eelneva versiooni, kui ilmneb pärast uuenduse tegemist, et sisestatud info on valesti kirjutatud või esineb mõni viga. See lihtsustab muudatuste tagasi kerimist hetke, kus viga ei eksisteerinud. Muidugi tuleb määrata ära, kui mitu versiooni salvestatakse ja peale millist hakatakse vanemaid kustutama uuemate lisandumisel, et vältida üleliigset andmemahu kasvu andmebaasis.

Lisaks tuleks lisada juhendi otsimiseks võimalus otsida seda fraasi järgi. Kasutaja saaks kirjutada otsitava nime otsingu lahtrisse ning vastavalt selle kuvatakse menüüs ainult juhendeid, mille nimed sobituvad otsi fraasiga.

Peale eelnevalt nimetatud etappide saavutamist oleks võimalik rakendus sättida ülesse kasutajatele ligipääsuks. Kasutajaid, kes võiksid olla rakendusest huvitatud leiaks Facebookis olevatest venetehnika huviliste gruppidest, kellele pakkuda antud rakendust prooviks. Sellisel viisil oleks võimalik koguda ka kasutajate käest otsest tagasisidet, kuidas rakendus kasutajale tundub. Kasutaja käest oleks võimalik saada konstruktiivset kriitikat funktsionaalsuse või UI kohta, mis neile ei meeldi. Selle info põhjal on võimalik antud rakendust edasi arendada ja paremaks muuta.

6 Kokkuvõte

Antud lõputöö eesmärgiks oli luua NSVL-aegse remondi manuaali raamatu alusel kaasaegne remondi manuaali veebirakendus. Lõputöö skoobiks oli luua kategoriseeritud remondi manuaal veebikeskkonnas, millele lisaks on võimekus otsida vaja minevat varuosa internetist müügilt.

Töö analüüsi osas anti ülevaade kavandatavatest funktsionaalsustest, võrreldi erinevaid võimalikke kasutatavaid tehnoloogiaid arenduseks ning pandi paika veebirakenduse disain, mis aitas efektiivsemalt töötada veebirakenduse arendusega. Lõputöö veebirakenduse prototüübi arenduskäiku kirjeldav peatükk annab ülevaate nii teenusepoolse, kui ka kliendipoolse rakenduse arhitektuurist, teenusepoole turvalisusest ning loodud lahendusest. Kasutatavateks tehnoloogiateks valiti teenusepoolel C# programmeerimiskeel ja .NET raamistik koos PostgreSQL andmebaasiga. Kliendipoolse tehnoloogia valikus osutus sobivaks .NET raamistikule loodud Blazor, mis võimaldab arendajal luua klientrakendusi ilma ühtegi rida Javascripti kirjutamata. Lisaks sellele oli ülesandes püstitatud veebiämbliku lahendus, mille arendamiseks valiti programmeerimis keel Python. Pythoni laiast teekide ja raamistike valikust otsustati Scrapy ja BeautifulSoupi kasuks, et luua veebiämblikut.

Loodud veebirakenduse prototüübil on olemas enamus analüüsi käigus täpsustatud nõudeid. Kuigi veebiämbliku funktsionaalsus ei valminud töö arenduse käigus vastavalt analüüsis seatud nõuetele, kuna lõputöö koostamiseks on piiratud aeg ja antud lahendus kujunes keerulisemaks, kui töö autor oli seda alustades hinnanud. Sellest olenemata on ülejäänud funktsionaalsus veebirakenduse prototüübil olemas. Antud rakendust kavatsetakse edasi arendada ning peale veebämbliku edukalt tööle saamist pakkuda potentsiaalsetele kasutajatele proovida, et koguda kasutajate tagasisidet rakenduse kohta. Lisaks sellele on töös selgelt väljatoodud rakenduse edasiarenduse võimalused ning ükskõik kes võib rakenduse edasisse arenduskäiku julgelt panustada.

Kasutatud kirjandus

[1] J. William, „3aam,“ 5 september 2020. [Võrgumaterjal]. Available: https://3aam.com/what-is-a-car-workshop-manual/. [Kasutatud 01 aprill 2023].

[2] G. B ja V. Vlad, „Technology Stack for Web Application Development,“ RubyGarage, 04 jaanuar 2020. [Võrgumaterjal]. Available: https://rubygarage.org/blog/technology-stack-for-web-development#article_title_16. [Kasutatud 04 aprill 2023].

[3] Kinsta, „What Is a Content Management System (CMS)?,“ Kinsta, 20 märts 2023. [Võrgumaterjal]. Available: https://kinsta.com/knowledgebase/content-management-system/. [Kasutatud 04 aprill 2023].

[4] J. Gottfired, „A complete guide to modern web applications,“ 26 veebruar 2020. [Võrgumaterjal]. Available: https://medium.com/jeremy-gottfrieds-tech-blog/a-complete-guide-to-modern-web-applications-793ae71b57ad. [Kasutatud 04 aprill 2023].

[5] RedHat, „What is a REST API?: RedHat,“ RedHat, 08 mai 2020. [Võrgumaterjal]. Available: https://www.redhat.com/en/topics/api/what-is-a-rest-api. [Kasutatud 10 aprill 2023].

[6] A. S. Gillis, „SOAP (Simple Object Access Protocol): TechTarget,“ TechTarget, juuni 2022. [Võrgumaterjal]. Available: https://www.techtarget.com/searchapparchitecture/definition/SOAP-Simple-

Object-Access-Protocol. [Kasutatud 10 aprill 2023].

[7] J. Hartman, „What is Java? Definition, Meaning & Features of Java Platforms,“ GURU99, 04 märts 2023. [Võrgumaterjal]. Available: https://www.guru99.com/java-platform.html. [Kasutatud 04 aprill 2023].

[8] Teradata, „What is Python?,“ Teradata, [Võrgumaterjal]. Available: https://www.teradata.com/Glossary/What-is-Python. [Kasutatud 06 aprill 2023].

[9] altexsoft, „The Good and the Bad of C# Programming,“ altexsoft, 29 oktoober 2021. [Võrgumaterjal]. Available: https://www.altexsoft.com/blog/c-sharp-pros-and-cons/. [Kasutatud 10 aprill 2023].

[10] J. A, „What Is JavaScript? A Basic Introduction to JS for Beginners,“ Hostinger, 31 jaanuar 2023. [Võrgumaterjal]. Available: https://www.hostinger.com/tutorials/what-is-javascript. [Kasutatud 10 aprill 2023].

[11] A. Jalli, „What Is the Ruby Programming Language?,“ builtin, 01 november 2022. [Võrgumaterjal]. Available: https://builtin.com/software-engineering-perspectives/ruby-programming-language. [Kasutatud 10 aprill 2023].

[12] A. S, „What Is PHP? Learning All About the Scripting Language,“ Hostinger, 10 aprill 2023. [Võrgumaterjal]. Available: https://www.hostinger.com/tutorials/what-is-php/. [Kasutatud 10 aprill 2023].

[13] Polyseptic, „Programming Languages Ranked By Difficulty,“ medium, 07 jaanuar 2023. [Võrgumaterjal]. Available: https://medium.com/@polyseptic/programming-languages-ranked-by-difficulty-afa6d564955b. [Kasutatud 10 aprill 2023].

[14] Vishak, „List Of Top Programming Languages From Easy To Hard To Learn [Updated 2023],“ codehack, 05 jaanuar 2023. [Võrgumaterjal]. Available: https://codeandhack.com/easy-to-hard-to-learn-programming-languages/. [Kasutatud 10 aprill 2023].

[15] K. Ubah, „The 8 Most Popular Backend Frameworks for Web Development,“ MAKEUSEOF, 06 aprill 2023. [Võrgumaterjal]. Available: https://www.makeuseof.com/most-popular-backend-frameworks/. [Kasutatud 10 aprill 2023].

[16] Microsoft, „What is .NET?,“ Microsoft, [Võrgumaterjal]. Available: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet. [Kasutatud 10 aprill 2023].

[17] React, „React,“ Meta Open Source, [Võrgumaterjal]. Available: https://react.dev/. [Kasutatud 10 aprill 2023].

[18] Vue, „Introduction,“ Vue, [Võrgumaterjal]. Available: https://vuejs.org/guide/introduction.html. [Kasutatud 10 aprill 2023].

[19] Bootstrap, „Build fast, responsive sites with Bootstrap,“ Bootstrap, [Võrgumaterjal]. Available: https://getbootstrap.com/. [Kasutatud 10 aprill 2023].

[20] MudBlazor, „The Blazor Component Library You Always Wanted,“ MudBlazor, [Võrgumaterjal]. Available: https://mudblazor.com/. [Kasutatud 10 aprill 2023].

[21] Microsoft, „Build beautiful web apps with Blazor,“ Microsoft, [Võrgumaterjal]. Available: https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor. [Kasutatud 10 aprill 2023].

[22] MongoDB, „Types of Databases,“ MongoDB, [Võrgumaterjal]. Available: https://www.mongodb.com/databases/types. [Kasutatud 10 aprill 2023].

[23] S. Sam, „Types of databases,“ tutorialspoint, 18 juuni 2020. [Võrgumaterjal]. Available: https://www.tutorialspoint.com/Types-of-databases#. [Kasutatud 10 aprill 2023].

[24] R. Peterson, „What is PostgreSQL? Introduction, Advantages & Disadvantages,“ GURU99, 18 veebruar 2023. [Võrgumaterjal]. Available: https://www.guru99.com/introduction-postgresql.html. [Kasutatud 10 aprill 2023].

[25] SQLite, „About SQLite,“ SQLite, [Võrgumaterjal]. Available: https://www.sqlite.org/about.html. [Kasutatud 10 aprill 2023].

[26] javaTpoint, „SQLite Advantages and Disadvantages,“ javaTpoint, [Võrgumaterjal]. Available: https://www.javatpoint.com/sqlite-advantages-and-disadvantages. [Kasutatud 10 aprill 2023].

[27] J. Hurley, „The Pros and Cons of MySQL,“ SMARTFILE, 3 juuni 2021. [Võrgumaterjal]. Available: https://www.smartfile.com/blog/the-pros-and-cons-of-mysql/. [Kasutatud 10 aprill 2023].

[28] D. Bruce, „Understanding the pros and cons of MongoDB,“ KnowledgeNile, [Võrgumaterjal]. Available: https://www.knowledgenile.com/blogs/pros-and-cons-of-mongodb/. [Kasutatud 10 aprill 2023].

[29] IntelliPaat, „Couchbase vs MongoDB - Differences and Comparison Table,“ IntelliPaat, 07 aprill 2023. [Võrgumaterjal]. Available: https://intellipaat.com/blog/couchbase-vs-mongodb/. [Kasutatud 10 aprill 2023].

[30] M. Kumwenda, „The 7 Best IDEs and Text Editors for C# Developers,“ MAKE USE OF, 23 aprill 2022. [Võrgumaterjal]. Available: https://www.makeuseof.com/ides-text-editors-best-c-sharp-development/. [Kasutatud 10 aprill 2023].

[31] JetBrains, „Free Educational Licenses,“ JetBrains, [Võrgumaterjal]. Available: https://www.jetbrains.com/community/education/#students. [Kasutatud 10 aprill 2023].

[32] H. Raoult, „5 Preferred Programming Languages for Web Scraping,“ ODSC, 11 august 2022. [Võrgumaterjal]. Available: https://opendatascience.com/5-preferred-programming-languages-for-web-scraping/. [Kasutatud 12 aprill 2023].

[33] Scrapy, „Scrapy at a glance,“ Scrapy, 03 veebruar 2023. [Võrgumaterjal]. Available: https://docs.scrapy.org/en/latest/intro/overview.html. [Kasutatud 10 mai 2023].

[34] IronPython, „Overview,“ IronPython, [Võrgumaterjal]. Available: https://ironpython.net. [Kasutatud 10 mai 2023].

[35] D. A. Norman, „The Design of Everyday Things: Revised and Expanded Edition: PDF DRIVE,“ 2013. [Võrgumaterjal]. Available: https://www.pdfdrive.com/the-design-of-everyday-things-revised-and-expanded-edition-e194281340.html. [Kasutatud 14 aprill 2023].

[36] nicepage, „Koduleht,“ nicepage, [Võrgumaterjal]. Available: https://nicepage.com/. [Kasutatud 14 aprill 2023].

[37] coolors, „Koduleht: coolors,“ coolors, [Võrgumaterjal]. Available:

https://coolors.co/. [Kasutatud 14 aprill 2023].

[38] themejunkie, „What is Figma? (And How to Use Figma for Beginners),“ themejunkie, [Võrgumaterjal]. Available: https://www.theme-junkie.com/what-is-figma/. [Kasutatud 14 aprill 2023].

[39] R. Anderson, „dotnet aspnet-codegenerator,“ Microsoft, 06 detsember 2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/tools/dotnet-aspnet-codegenerator?view=aspnetcore-7.0. [Kasutatud 23 aprill 2023].

[40] R. Anderson, S. Dean, N. Schonning, T. Dykstra, S. Addie, isaac2004 ja P. Andy, „Entity Framework,“ Microsoft, 22 juuli 2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/aspnet/entity-framework. [Kasutatud 23 aprill 2023].

[41] D. Pine, W. Genevieve, K. Sharkey, N. Schonning, B. Wagner, D. Coulter, T. Dykstra, N. Turn, M. Wenzel, M. Jones, L. Latham, T. Pratt ja yishengjin1413, „Key Security Concepts,“ Microsoft, 26 mai 2021. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/standard/security/key-security-concepts. [Kasutatud 09 mai 2023].

[42] D. Roth, J. Fritz, T. Southwick, S. Addie ja S. Smith, „Blazor e-book,“ 13 veebruar 2023. [Võrgumaterjal]. Available: https://dotnet.microsoft.com/en-us/download/e-book/blazor-for-web-forms-devs/pdf. [Kasutatud 09 mai 2023].

[43] Bootstrap, „About,“ Bootstrap, [Võrgumaterjal]. Available: https://getbootstrap.com/docs/4.1/about/overview/. [Kasutatud 09 mai 2023].

[44] MudBlazor, „What is MudBlazor?,“ MudBlazor, [Võrgumaterjal]. Available:

https://mudblazor.com/mud/introduction#first-step:-mudblazor-as-a-component-library-. [Kasutatud 09 mai 2023].

[45] J.-Y. Perrier, O. Ruikar, E. Weyl, H. Willee, N. Schonning ja wbamberg, „SPA (Single-page application),“ Mozilla, 22 veebruar 2023. [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Glossary/SPA. [Kasutatud 09 mai 2023].

[46] C. BasuMallick, „What Is a Single-Page Application? Architecture, Benefits, and Challenges,“ spiceworks, 18 oktoober 2022. [Võrgumaterjal]. Available: https://www.spiceworks.com/tech/devops/articles/what-is-single-page-application/#_001. [Kasutatud 09 mai 2023].

[47] P. Morris, „Components,“ Blazor University, [Võrgumaterjal]. Available: https://blazor-university.com/components/. [Kasutatud 09 mai 2023].

[48] L. Latham, R. Anderson, A. Sharma, S. Shariq, B. Haarsma, R. Mourato, A. Krishna, S. Addie, M. Mahouti ja mrlife, „ASP.NET Core Razor components,“ Microsoft, 09 mai 2023. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/aspnet/core/blazor/components/?view=aspnetcore-7.0. [Kasutatud 10 mai 2023].

[49] Volkswagen AG, „erWin,“ Volkswagen AG, [Võrgumaterjal]. Available: https://erwin.volkswagen.de/erwin/showInfoTour.do. [Kasutatud 02 aprill 2023].

[50] sales, „Elsawin vs ETKA vs ETOS: what's the difference?,“ OBDexpress, 18 oktoober 2017. [Võrgumaterjal]. Available: http://blog.obdexpress.co.uk/2017/10/18/elsawin-vs-etka-vs-etos/#. [Kasutatud 1 aprill 2023].

[51] P. Krill, „C# popularity surges in Tiobe programming language index,“ InfoWorld,

10 mai 2022. [Võrgumaterjal]. Available: https://www.infoworld.com/article/3660078/c-sharp-popularity-surges-in-tiobe-programming-language-index.html. [Kasutatud 10 aprill 2023].

[52] ByteHideBlog, „C# in 2023: The MOST POPULAR Programming Language?,“ ByteHideBlog, 03 veebruar 2022. [Võrgumaterjal]. Available: https://www.bytehide.com/blog/c-wants-to-become-the-most-popular-programming-language-in-2022. [Kasutatud 10 aprill 2023].

[53] D. A. Norman, „Emotional Design: Why We Love (Or Hate) Everyday Things: PDF DRIVE,“ 2005. [Võrgumaterjal]. Available: https://www.pdfdrive.com/emotional-design-why-we-love-or-hate-everyday-things-d158398465.html. [Kasutatud 14 aprill 2023].

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

Mina, Kristjan Tõeväli

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Kaasaegse remondi käsiraamatu rakenduse loomine NSVL-aegse autoremondi raamatu põhjal“, mille juhendaja on Meelis Antoi
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

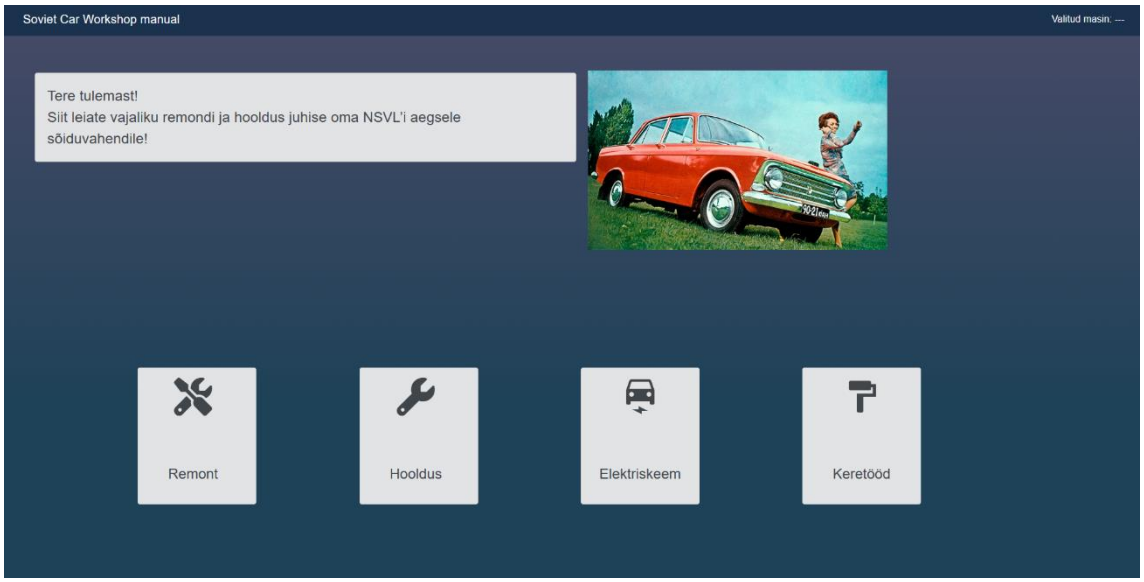
15.05.2023

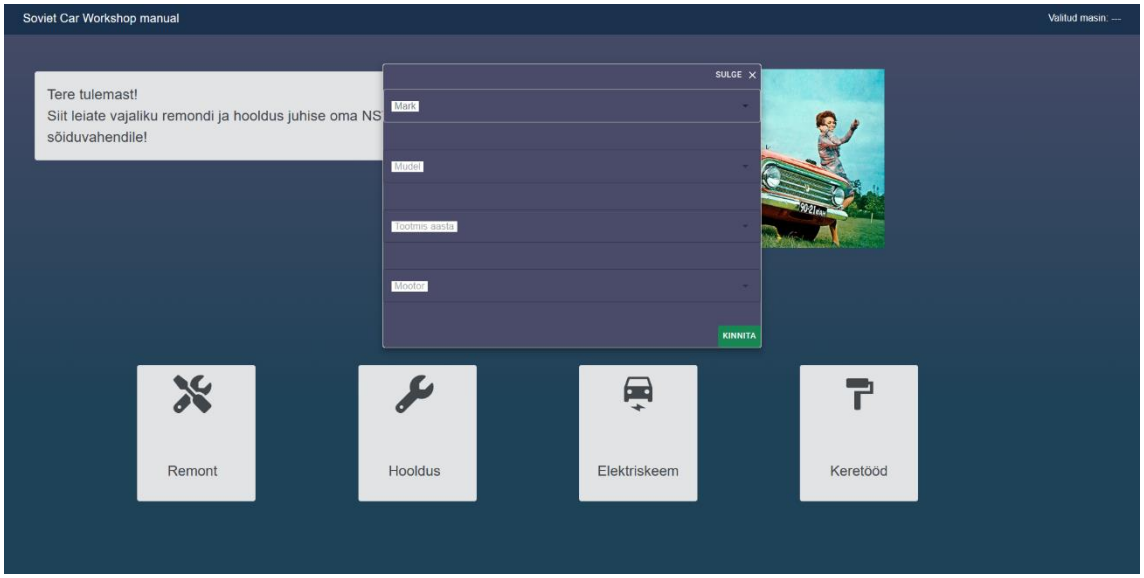
1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või 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.

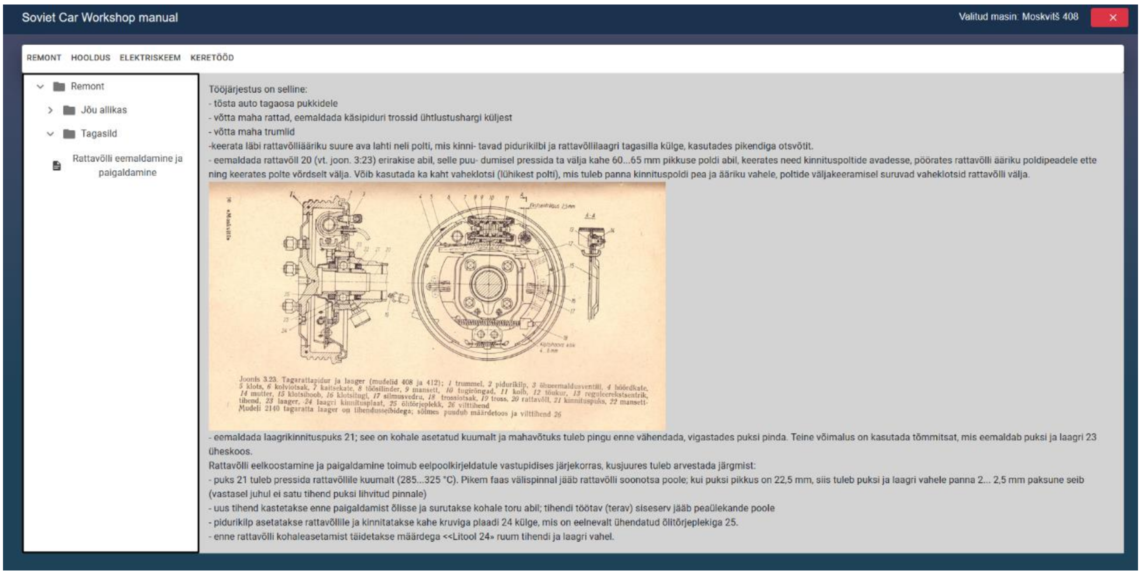
Lisa 2 – Veebirakenduse versioonihaldus

https://gitlab.com/l-put/CarWorkshopManual

Lisa 3 – Veebirakenduse prototüübi vaated






Soviet Car Workshop manual
Valitud masin: Moskvitš 408
REMONT
HOOLDUS
ELEKTRISKEEM
KERETÖÖD
Remont
Jõu allikas
Tagasild
Rattavõlli eemaldamine ja paigaldamine
Tööjärjestus on selline:
- tõsta auto tagaosa pukkidele
- võtta maha rattad, eemaldada käsipiduri trossid ühtlustushargi küljest
- võtta maha trumlid
-keerata läbi rattavõlliääriku suure ava lahti neli polti, mis kinni- tavad pidurikilbi ja rattavõllilaagri tagasilla külge, kasutades pikendiga otsvõtit.
- eemaldada rattavõll 20 (vt. joon. 3:23) erirakise abil, selle puu- dumisel pressida ta välja kahe 60...65 mm pikkuse poldi abil, keerates need kinnituspoltide avadesse, pöörates rattavõlli ääriku poldipeadele ette ning keerates polte võrdselt välja. Võib kasutada ka kaht vaheklotsi (lühikest polti), mis tuleb panna kinnituspoldi pea ja ääriku vahele, poltide väljakeeramisel suruvad vaheklotsid rattavõlli välja.
Joonis 3.23. Tagarattapidur ja laager (mudelid 408 ja 412); 1 trummel, 2 pidurikilp, 3 õhueemaldusventiil, 4 höördkate, 5 klots, 6 kolviotsak, 7 kaitsekate, 8 töösilinder, 9 mansett, 10 tugirõngad, 11 kolb, 12 tõukur, 13 reguleerekstsentrik, 14 mutter, 15 klotsihoob, 16 klotsitugi, 17 silmusvedru, 18 trossiotsak, 19 tross, 20 rattavõll, 21 kinnituspuks, 22 mansett-tihend, 23 laager, 24 laagri kinnitusplaat, 25 õlitõrjeplekk, 26 vilttihend
Mudeli 2140 tagaratta laager on tihendusseibidega; sõlmes puudub määrdetoos ja vilttihend 26
- eemaldada laagrikinnituspuks 21; see on kohale asetatud kuumalt ja mahavõtuks tuleb pingu enne vähendada, vigastades puksi pinda. Teine võimalus on kasutada tõmmitsat, mis eemaldab puksi ja laagri 23 üheskoos.
Rattavõlli eelkoostamine ja paigaldamine toimub eelpoolkirjeldatule vastupidises järjekorras, kusjuures tuleb arvestada järgmist:
- puks 21 tuleb pressida rattavõllile kuumalt (285...325 °C). Pikem faas välispinnal jääb rattavõlli soonotsa poole; kui puksi pikkus on 22,5 mm, siis tuleb puksi ja laagri vahele panna 2... 2,5 mm paksune seib (vastasel juhul ei satu tihend puksi lihvitud pinnale)
- uus tihend kastetakse enne paigaldamist õlisse ja surutakse kohale toru abil; tihendi töötav (terav) siseserv jääb peaülekande poole
- pidurikilp asetatakse rattavõllile ja kinnitatakse kahe kruviga plaadi 24 külge, mis on eelnevalt ühendatud õlitõrjeplekiga 25.
- enne rattavõlli kohaleasetamist täidetakse määrdega <<Litool 24» ruum tihendi ja laagri vahel.

Lisa 4 – Olemi-suhte diagramm