LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

# TĪMEKĻA VIETNES PRIEKŠGALSISTĒMAS ĀTRDARBĪBAS UZLABOŠANA MĒROGOJAMĪBAS NODROŠINĀŠANAI

KVALIFIKĀCIJAS DARBS

Autors: **Jēkabs Jānis Kalniņš**

Studenta apliecības Nr.: Jk16070

Darba vadītājs: profesors, Dr. dat. Uldis Straujums

RĪGA 2018

## Anotācija

Kvalifikācijas darbā – “Tīmekļa vietnes priekšgalsistēmas ātrdarbības uzlabošana mērogojamības nodrošināšanai” ir izstrādāts interneta veikals. Priekš izstrādes tika izvēlēta spējas programmatūras izstrādes metode. Interneta veikals nodrošina informāciju krievu valodā. Interneta veikalam ir iespējams piekļūt no jebkuras viedierīces, kurai ir savienojums ar internetu. Informācijas menedžēšana notiek ar administrācijas palīdzību, kura atrodas uz šīs pašas vietnes.


Atslēgvārdi: spējas programmatūras izstrāde, interneta veikals, MVC, pirkumi.

## Abstract

## Web site front-end response speed improvement for scalability support

In the qualification work an ecommerce site is developed. For development Agile programming method was used. The ecommerce site provides information in Russian. The ecommerce site is available from any smart device which has a connection to the internet. Information management is available from the admin, which is on the same domain.
Keywords: Agile programming method, ecommerce, MVC, purchases

# Satura rādītājs

## Apzīmējumu saraksts

Satvars – Framework.

JS – JavaScript programmēšanas valoda.

jQuery – JavaScript valodas bibliotēka. Ļauj rakstīt JavaSctipt kodu vieglākā pierakstā.

Bootstrap – viena no populārākajām CSS stilu bibliotēkām. Ļoti noderīga responsīvajam dizainam.

Opencart – interneta veikala platforma.

CodeIgniter – PHP valodas satvars, kurā ir iestrādāta pamatfunkcionalitāte programmatūras veidošanai.

Minimizācija – programmas koda saīsināšana, kas nodrošina koda ātrāku ielādi lapā.

MatthiasMullie/Minify – failu samazināšanas programmatūra, kas darbojas ar PHP.

Controller – komponente, kas apstrādā informāciju, kuru iegūst no View un Model.

View – komponente, kurā atrodas HTML un PHP kods. PHP tiek izmantots dinamiskai datu izvadei.

Model – komponente, no kuras tiek veikti pieprasījumi uz datubāzi.

NPM – pakotnes pārvaldnieks, kas ir balstīts uz JavaScript.

Gulp – NPM bibliotēka, kas ļauj izmantot SCSS stilu pierakstu, minimizē failus un ļauj reāllaikā apskatīt veiktās izmaiņas pārlūkā.

SCSS – CSS pieraksts, kas ļauj pārskatāmāk rakstīt stilus.

SimpleXMLReader – uz PHP rakstīta klase, kas ļauj nolasīt XML failu struktūru un atgriezt informāciju pa vienam ierakstam.

Callback – atzvanīšanas metode. Metode, kura izpildās pēc kādas funkcijas darbības.

ISBN – internacionālā standarta grāmatu numurs.

Lazyload – bilžu pakāpeniska ielāde pēc visa HTML ielādes. Izmanto, lai netiktu pieprasīta uzreiz visa informācija. Parasti ielādē bildes tikai tad, kad tās ir nonākušas skatlaukā.

LazySizes – uz JavaScript bāzēts spraudnis, kas spēj ieladēt bildes ar lazyload metodi.

Swiper – uz JavaScript bāzēts spraudnis, kas piešķir lapas elementiem slaidrādes efektu.

Gulp – programmatūra, kas automatizē darbus. Balstīts uz JavaScript.

# Ievads

Cilvēki arvien biežāk izmanto interneta veikalus [14]. Tāpēc iespēja kļūt par lielu interneta veikalu, kas ienes peļņu, paliek arvien lielāka. Daudzi uzņēmumi pārdod savu produkciju internetā vietās kā Facebook [15], bet tas nav tik ērti lielākiem piegādātājiem. Tāpēc pasūtītājs vēlas izveidot interneta veikalu, kas nodrošinātu iespēju lieliem piegādātājiem pārdot savu produkciju.

Interneta veikala uzdevums ir nodrošināt produktu piedāvāšanu klientiem krievu valodā.

Programmatūras izstrāde notika PHP, JavaScript valodās. Interneta veikls tika veidots uz Opencart platformas. Programmatūras izstrādē tika izmantota spējas programmatūras izstrādes metodika.

Dokuments strukturāli ir sadalīts septiņās daļās: funkcionālās prasības, lietotājstāsti, programmatūras projektējuma apraksts, projekta organizācija, konfigurācijas pārvalde, kvalitātes nodrošināšana un darbietilpības novērtējums. Pielikumā ir programmatūras koda fragments failu samazināšanai un apvienošanai ar divām dažādām metodēm.

# 1. Funkcionālās prasības

Projekta izstrādes sākumā ir svarīgi izvēlēties labu satvaru uz kura balstīt interneta veikalu. Šim projektam ir izmantots Opencart, kurš ir veidots uz CodeIgniter satvara. Opencart aizmugursistēmai tiek izmantots PHP un priekšgalsistēmai tiek izmantotas Bootstrap, kas ir CSS bibliotēka un jQuery, kas ir JavaScript bibliotēka.

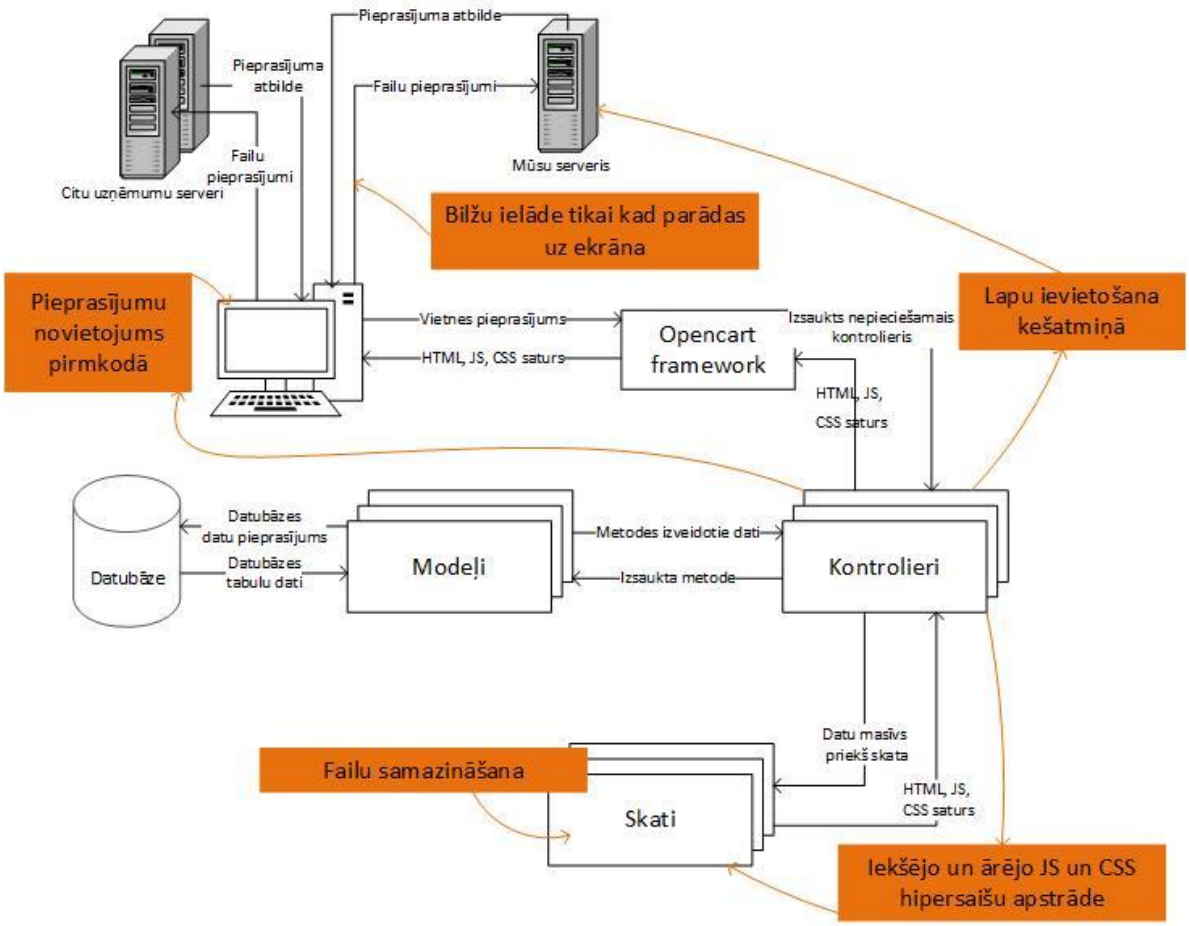
Lai nodrošinātu pasūtītājam apmierinošu darbību interneta veikalā ar 700 000 produktiem ir jāievieš bilžu izmēra samazināšana, stilu un JavaScript minimizācija**.** Pats Opencart nav ļoti optimizēts un ātrs, skatoties pēc Google PageSpeed Insights, tāpēc ir vērīgāk jāskatās kā optimizēt lapu. Arī ir jānodrošina, ka visas lapas funkcijas strādās uz visiem modernajiem pārlūkiem un Internet Explorer [1].

Lai samazinātu lapas ielādi tika izmantota NPM bibliotēka Gulp, kas samazina failu izmēros, tos sakompresējot [2]. Papildus tika izmantota MatthiasMullie/Minify bibliotēka, kas apvieno JS un CSS failus, kas tiek lapā ielādēti kā hipersaite. Arī HTML ir iespējams sakompresēt un tas neietekmēs negatīvi klienta pieredzi, jo vienīgas, kur ir iespējams pamanīt izmaiņu, ir apskatot lapas pirmskodu. Tur tas ir grūti pārlasāms.

Sākuma lapai interneta veikalam ir jāielādējas ātri, jo tā ir pirmā lapa, ko klients pamana [3]. Produkti, kas tiek ielādēti no dažādām kategorijām dinamiski, nedrīkst ietekmēt ātrdarbību.

Opencart darbojas uz MVC principa [4]. Tā ir programmatūras struktūra, kura palīdz nodalīt programmatūras darbību uz trīs veidiem. Model ir programmatūras saziņas centrs ar datubāzi. Šeit informācija tiek sagatavota nepieciešamajā struktūrā un nosūtīta uz kontrolieri, kas ir izsaukusi kādu no Modeļa funkcijām. Controller ir programmatūras galvenā daļa, kas pēc tam dotajiem parametriem, kas parasti tiek padoti ar GET vai POST parametriem, saveido pilnīgi visus datus, kas būs nepieciešami View, kas šo informāciju attēlos uz ekrāna. Vienas lapas ielādē nostrādā vairāki kontrolieri, kur katrs kontrolieris veic savu specifisko uzdevumu.

Izanalizējot lietotājstāstus attēlā 1.1 ir iespējams apskatīt kādi optimizācijas darbi lapai ir nepieciešami, lai sasniegtu ātrāku lapas darbību.




*1.1 att.* **Priekšgala optimizācijas darbi**

# 2. Lietotājstāsti

Lai izveidotu šo programmatūru tika izvirzīta spējas izstrādes metode [5], jo tā paredz izstrādi ar iteratīviem soļiem, kura katra soļa sākumā tiek precizēts soļa plāns. Katra soļa sākumā tiek izveidoti akcepttesti, kuri novērtē funkcionalitātes darbību.

Katra lietotājstāsta ilgums ir līdz 4 dienām, kur katrs lietotājstāsts ir viens uzdevums no programmatūras pasūtītāja. Tikai pēc veiksmīgi pabeigta akcepttesta un pasūtītāja novērtējuma tiek uzsākta darbība pie nākamā lietotājstāsta.

Tā kā projekts nebija tikai lapas priešgalsistēmas ātrdarbības uzlabošana, autoram bija jāveic arī citi lapas uzturēšanas un funkcionalitātes papildināšanas darbi.

Lietotājstāsti tapuši sarunās ar pasūtītāju:

1. Iterācija - produktu datu apstrāde.

*2.1. tabula*

**1. Iterācija**

| Lietotājstāsts | Sarežģītības novērtējums |
|---|---|
| Interneta veikalā ir nepieciešami produkti no piegādātāja. | 7 |
| Vēl viena internetveikala XML produktu imports lapā no nekvalitatīviem datiem. | 3 |
| Ievietot visus nepieciešamos kuponus datubāzē. | 2 |
| Xml lapas izveidošana priekš salidzini.lv un kurpirkt.lv preču salīdzināšanas portāliem. | 4 |

2. Iterācija - groza darbību uzlabošana.

*2.2. tabula*

**2. Iterācija**

| Lietotājstāsts | Sarežģītības novērtējums |
|---|---|
| Dažādām ģeolokācijām ir nepieciešama savādāka cena sūtīšanai. | 4 |
| Ja groza summa ir lielāka par noteiktu summu, ir nepieciešama bezmaksas piegāde. Vērtībai ir jābūt rediģējamai no administrēšanas puses. | 4 |
| Pasūtījuma pēdējā posmā ir jāparāda veiksmīga pirkuma lapa, kurā ir jāparāda sūtījuma kopsavilkums un, ja ir klāt arī bezmaksas produkts, tas ir vizuāli jāattēlo. | 5 |
| Jāvar produktus pasūtīt kā juridiskai personai. | 2 |
| Dažādiem atmaksas veidiem ir nepieciešami dažādi e-pasta paziņojumi. | 1 |
| Izveidot, lai modulim nebūtu noklusētās vērtības sūtīšanai uz pakomātu. | 3 |

3. Iterācija - pēc pasūtījuma informācijas menedžēšana.

*2.3. tabula*

**3. Iterācija**

| Lietotājstāsts | Sarežģītības novērtējums |
|---|---|
| Pēc e-pasta atlasīt visu klienta pasūtījumu vēsturi. | 1 |
| Kad nomainās sūtījuma statuss ir jānosūta klientam ziņa. | 7 |
| Uztaisīt pasūtījuma stāvokļa reprezentāciju. | 3 |

4. Iterācija - produktu stratēģiska izvietošana lapā.

*2.4. tabula*

**4. Iterācija**

| Lietotājstāsts | Sarežģītības novērtējums |
|---|---|
| Produkta atvērumā vajag būt līdzīgiem produktiem. | 4 |
| Sākumlapā nepieciešami 3 nejauši izvēlēti produkti no divām kategorijām. | 3 |
| Nepieciešams, lai viegli varētu pievienot dažādām kategorijām slīdņu skatus. | 3 |
| Pirktāko produktu kategorijā jābūt iespējai nopozicionēt kādu produktu augstākā pozīcijā. | 7 |

| Lietotājstāsts | Sarežģītības novērtējums |
|---|---|
| Sākumlapā blakus slīdnim ir nepieciešams dienas preces bloks. Precēm jābūt norādāmām administrēšanas pusē. Jāvar norādīt uzreiz vairākas preces. | 7 |
| Kategoriju izvēlnē ir jāparādās visām apakškategorijām, jāmaina esošā moduļa funkcionalitāte. | |

5. Iterācija - lapas dizaina maiņas

*2.5. tabula*

**5. Iterācija**

| Lietotājstāsts | Sarežģītības novērtējums |
|---|---|
| Atlaidēm vajag nomainīt dizainu, lai tās izceļas. | 2 |
| Lapā ir nepieciešamas reklāmjoslas, kuras var attēlot katrā lapā. | 2 |
| Viedierīcēm ir jābūt galvenei ar kompaktāku informāciju. | 3 |

6. Iterācija - priekšgalsistēmas ātrdarbības uzlabošana

*2.6. tabula*

**6. Iterācija**

| Lietotājstāsts | Sarežģītības novērtējums |
|---|---|
| Bildes slīdņos nevajag uzreiz visām ielādēties. Tikai tad, kad parādās uz ekrāna. | 4 |
| Ārējo un iekšējo hipersaišu (CSS, JS) failu savienošana un minimizācija. | 5 |
| View failu minimizācija. | 3 |
| Sākumlapas visas informācijas saglabāšana kešatmiņā. | 4 |
| Pārējo lapu saglabāšana kešatmiņā. | 3 |

# 3. Programmatūras projektējuma apraksts

## 3.1. Ievads

Viens no iemesliem, kāpēc tika izvēlēta spējas programmēšanas metode ir tas, ka pasūtītājs nezināja kāda funkcionalitāte viņam būs nepieciešama. Šī iemesla dēļ, sākot projektu, netika izstrādāts programmatūras projektējuma apraksts.

Tā kā darbs netiek rakstīts par visu sistēmu, bet tikai tās daļām, katrs no iterācijas soļiem ir aprakstīts atsevišķi ar savu uzbūvi un funkcijām. Funkcijas šīm iterācijām ir kopējas, jo atrodas tajā pašā programmatūras daļā. Daudz jaunu klašu netika izveidotas, jo Opencart sniedz visas nepieciešamās bāzes klases un bija nepieciešams papildināt funkcionalitāti vai to optimizēt. Vienīgās kopīgās funkcijas ir no Opencart.

Darbā ir dokumentēta tikai funkcionalitāte, kuru ir ieviesis autors.

## 3.2. Produktu datu apstrāde

Lai importētu un eksportētu datus tika izmantots XML formāts. Gan produktu piegādātāju produktu fails bija šādā formātā, gan salidzini.lv [10] un kurpirkt.lv [11] salīdzināšanas portāliem ir nepieciešami XML faili, lai viņi varētu iekļaut mūsu veikala produktus.

### 3.2.1. Produktu ievietošana datu bāzē

Produktu ievietošanai tika izveidotas divi klases pagarinājumi. Viens no tiem ir Opencart Controller paplašinājums, kas atļauj izmantot noklusēto funkcionalitāti. Otrs ir SimpleXMLReader klases paplašinājums, kas atļauj nolasīt XML failu un tā struktūru. Ļoti svarīga SimpleXMLReader klases īpašība ir tāda, ka tā atgriež tikai nepieciešamo informāciju datu masīvā.

XML fails, kurš tiek apstrādāts, tika lejupielādēts iepriekš, jo tā izmērs ir 860 MB. Lai nodrošinātu pilnīgu XML importu, bija nepieciešams palaist funkcionalitāti no komandrindas, jo, to darot no interneta pārlūka, autors saskarās ar pieprasījuma noilgumiem.

Otra interneta veikala produktu imports netiks aprakstīts, jo tika izmantotas šīs pašās metodes, bet tikai pamainīta *parsephp* metode, lai dati tiktu no jaunā XML tiktu apstrādāti pareizi.

#### 3.2.1.1. Produktu ievietošanai izmantotās metodes

Šīs funkcionalitāte, galvenokārt, balstās uz Model metodēm, kuras ievieto informāciju datubāzē. Kategorijas, kurās dati tiks importēti, tika noteikti pirms funkcionalitātes izveides.

***Metožu dokumentācija***

- (void)registerCallback

  Iestata XML koka daļu, kuru vajag nolasīt no faila.

  Parametri:

  - (String) path – ceļš uz XML koka daļu;
  - (Array) callback – norāda callback metodes nosaukumu.

- (void) parsephp

Sagatavo produktu datus pēc datubāzes uzbūves, un tiek veikta nepieciešamā datu manipulācija.

Parametri:

  - data_xml – visi dati, kas tika atlasīti no SimpleXML Reader darbības.
- (int) addProduct

Datubāzē tiek pievienota produkta informācija. Tiek atgriezts rediģētā produkta identifikators mūsu datubāzē.

Parametri:

  - (int) id – produkta unikāls identifikators, kurš tiek iegūts no preču piegādātāja XML;
  - (array) data – dati, kuri tika iegūti no parsephp.

- (int) editProduct

Rediģē datubāzē jau ievietota produkta informāciju. Maina tikai to, kura ir norunāta, ka nevar mainīties. Tiek atgriezts rediģētā produkta identifikators mūsu datubāzē.

Parametri:

  - (int) id – produkta unikāls identifikators, kurš tiek iegūts no preču piegādātāja XML;
  - (array) data – dati, kuri tika iegūti no parsephp.

#### 3.2.1.2. Produktu XML struktūra

Šādi izskatās viena produkta struktūra.

```
<offer id=" ... " type=" ... " available="... ">
      <url> ... </url>
      <price> ... </price>
      <categoryId> ... </categoryId>
      <categoryId> ... </categoryId>
      <picture> ... </picture>
      <author> ... </author>
      <name> ... </name>
      <publisher> ... </publisher>
      <series> ... </series>
      <year> ... </year>
      <ISBN> ... </ISBN>
      <volume> ... </volume>
      <description> ... </description>
      <md5> ... </md5>
      <cover> ... </cover>
      <pages> ... </pages>
</offer>
```

***Mainīgo dokumentācija***

Ne visi tagi ir katram produktam. Grāmatām ir papildus informācija.

**offer** – tags, kas ietver informāciju par produktu;

**id** – offer taga atribūts, kas ir unikāls produkta identifikators;

**type** – offer taga atribūts, kas ir produkta tips;

**available** – offer taga atribūts, kas nosaka vai produkts ir pieejams;

**url** – tags, kur atrodas hipersaite uz produktu piegādātāja lapā;

**price** – tags, kur atrodas produkta cena;

**categoryId** – tags, kurā atrodas produkta kategorijas id;

**picture** – tags, kur atrodas hipersaite uz produkta bildi;

**author -** tags, kur glabājas informācija par autoru (ja tā ir grāmata);

**name** – tags, kur glabājas produkta nosaukums;

**publisher** – tags, kur glabājas informācija par izdevēju (grāmatām);

**series** – tags, kur glabājas informācija par produkta sēriju (grāmatām);

**year** – tags, kur glabājas produkta izdevuma gads (grāmatām);

**ISBN** – tags, kurā glabājas ISBN kods (grāmatām);

**volume** – tags, kurā ir izdevuma numurs (grāmatām);

**description** – tags, kurā ir ievietots produkta apraksts;

**md5** – tags, kurā atrodas produkta jaucējkods, pēc kura var pārbaudīt vai produkta informācija ir mainījusies;

**cover** – tags, kurā ir vāka veids (grāmatām);

**pages** – tags, kurā ir lapas pušu skaits (grāmatām).

### 3.2.2. XML izveidošana no datubāzē esošajiem produktiem

Lai izveidotu XML failu, nebija nepieciešams izmantot jau kādu gatavu klasi, kas apstrādā datus, jo XML ir parasts teksta fails. To var salīdzināt ar kāda produkta View izveidi. Tika izveidots Controller klases pagarinājums. Bija nepieciešams, lai, kad kāds no šiem portāliem atver hipersaiti, automātiski tiktu lejupielādēts XML fails.

Lai atzīmētu kurus produktus vajag eksportēt, administrācijas pusē tika izveidots divas izvēles rūtiņas, kuru vērtības tika saglabātas datubāzē. Lai nebūtu jāmaina datubāzes struktūra, tika izmantota tabulas lauks, kurā tiek ievietota informācija JSON formātā.

#### 3.2.2.1. XML izveidošanai izmantotās funkcijas

- (void) index

Satur visu funkcionalitātes loģiku, visas metodes tiek izsauktas no šīs funkcijas.

- (void) getProductsSalidzini

Izpilda datubāzes pieprasījumu, kur dabū informāciju par produktu.

- (array) getProduct

Atgriež visu informāciju par produktu. Šajā gadījumā tika izmantots tikai, lai noteiktu dziļāko kategoriju kurā atrodas produkts.

Parametri:

- (int) product_id – produkta identifikātors.

- (string) getProductUrlFull

Tiek iegūta hipersaite uz produktu no kādas no kategorijām.

Parametri:

- (int) product_id – produkta identifikātors;
- $deepest_category_id – dziļākās kategorijas identifikators.

#### 3.2.2.2. Izveidotā XML struktūra

Šādi izskatās viena produkta struktūra priekš eksporta.

```
<item>
      <name> ... </name>
      <name_ru> ... </name_ru>
      <link> ... </link>
      <price> ... </price>
      <image> ... </image>
      <category_full> ... </category_full>
      <category_link> ... </category_link>
      <manufacturer> ... </manufacturer>
      <in_stock> ... </in_stock>
      <model> ... </model>
      <in_stock> ... </in_stock>
      <product_id> ... </product_id>
      <delivery_cost_riga> ... </delivery_cost_riga>
      <delivery_latvija> ... </delivery_latvija>
</item>
```

***Mainīgo dokumentācija***

Šie mainīgie tika izmantoti atbilstoši salīdzināšanas portālu prasībām. [10][11]

**item** – tags, kas ietver informāciju par produktu;

**name** – tags, kur atrodas produkta nosaukums;

**name_ru** – tags, kur atrodas produkta nosaukums krievu valodā;

**price**– tags, kur atrodas produkta cena;

**image** – tags, kur atrodas produkta bildes hipersaite;

**category_full** – tags, kur atrodas kategorijas nosaukums, kurā atrodas produkts;

**category_link** – tags, kur atrodas kategorijas hipersaite;

**manufacturer** – tags, kurā atrodas ražotāja nosaukums;

**in_stock** – tags, kurā glabājas, cik vienību ir palicis noliktavā;

**product** – tags, kurā atrodas produkta identifikators;

**delivery_cost_riga** – tags, kurā atrodas cena piegādei Rīgā;

**delivery_latvija** – tags, kurā atrodas cena piegādei Latvijā.

## 3.3. Groza funkcionalitātes uzlabošana

Jau esošā groza funkcionalitāte nodrošināja apmierinošu funkcionalitāti, tāpēc nebija nepieciešams veikt lielas izmaiņas. Groza process darbojas, katru reizi izveidojot jaunu pasūtījumu, kad nonāk pasūtījuma apstiprināšanas lapā.

### 3.3.1. Produkta pievienošanu ar nulles cenu

Interneta veikalā netiek uzturēti produkti, kuru cena ir nulle. Tāpēc nav iespējas izveidot jaunu produktu. Tas arī ir ļoti nepraktiski, jo ir nepieciešamība pievienot produktu, kas jau atrodas veikalā. Rastos datu dublēšana. Jaunas metodes netika pievienotas, bet tikai papildināta esošā *addOrder* Model metode.

No datubāzes tika iegūtas vērtības, kuras no administrācijas puses tika saglabātas konfigurācijā (iestatījumu tabula datubāzē). No iegūtajiem datiem ar *getProduct* metodi tika iegūts produkts, un tikai nepieciešamā informācija ievietota produkta struktūrā.

```
$data['products'][] = [
    'product_id' => $free_product['product_id'],
    'name' => $free_product['name'],
    'model' => $free_product['model'],
    'option' => [],
    'download' => [],
    'quantity' => 1,
    'subtract' => 0,
    'price' => 0,
    'total' => 0,
    'tax' => 0,
    'reward' => 0,
    'tax_class_id' => 0,
    'cart_id' => 0,
    'key' => 0,
];
```

Ir atstātas daudz atslēgas ar tukšām vērtībām, lai nodrošinātu datu struktūras saglabāšanu.

## 3.4. Priekšgalsistēmas ātrdarbības uzlabošana

Apakšnodaļā tiek aprakstīts kādas darbības veica autors, lai uzlabotu lapas ielādes ātrumu. Izveidotajā interneta veikalā radās liela problēma - vietne ielādējās ļoti lēni un tas atturētu klientus no lapas izmantošanas [12].

Lapas ātrdarbības uzlabošanai ļoti palīdzēja Google PageSpeed Insights [13]. Tur bija iespēja apskatīt, kāda bija lapas ātrdarbība un kāda tā ir pēc kādas optimizācijas ieviešanas. Kā arī iespēja redzēt kādus darbus vēl nepieciešams veikt, lai uzlabotu ātrdarbību.

### 3.4.1. Bilžu ielādē ar lazyload palīdzību

Bilžu attēlošanai sākuma lapā tiek izmatots "lazySizes" spraudnis. Lapas ielādes laikā tiek norādītas bildes uzreiz redzamajiem produktiem. Pārējos slīdņos tiek norādīts ceļš uz bildi, kuru būs nepieciešams ievietot. Katrai bildei ir jāpieliek klāt klase "lazyload". Tā nav paredzēta bildes stiliem, bet JavaSript, lai varētu ielādēt bildi pēc nepieciešamības. Pēc ielādes klase tiek nomainīta uz klasi lazyloaded. Produkti ir ievietoti Swiper ietvarā, kas tos attēlo kā slīdņus.

#### 3.4.1.1. Slīdņu uzbūve

```
<div class="swiper-container">
  <div class="swiper-wrapper">
   <div class="product-layout col-xs-12 swiper-slide">
    <div class="product-thumb transition">
      <div class="image"><a href=" ... " class="img-responsive
      lazyload ltest"/></a></div>
       <div class="caption">
         <p class="sm center">
          <a href=" ... "> ... </a>
        </p>
         <p class="price"> ... </p>
      </div>
    </div>
   </div>
  </div>
</div>
```

## 3.5. Failu apvienošana un samazināšana

Galvenie faili, kas tiek ielādēti mājaslapā ir JavaScript un CSS. To novietojums lapā un laiks, kad šie faili tiek ielādēti, ir būtisks, jo tas ietekmē lietotāja pieredzi ar lapu. Autors, papildus turpmāk minētajām funkcionalitātēm, veica failu lejupielāžu pozīciju maiņu, kas priekš lapas lejupielādes sākuma atstāja tikai pašus nepieciešamākos stilus un JavaScript kodu.

### 3.5.1. Servera failu samazināšana

Darba failu samazināšanai tika izmantots Gulp. Darbs tiek dalīts divas daļās: izstrādes vide, gala produkta kods. Gulp atļauj veidot izmaiņas darba vidē automātiski tos nosūtīt uz gala produkta vidi - jau samazinātu un apvienotu vienā failā. Gulp arī atļauj strādāt ar SCSS failiem, kurus tas automātiski pārveido uz parastu CSS. Gulp fails sastāv no uzdevumiem, kuri tiek izsaukti no servera komandrindas.

Papildus ir iespējams arī samazināt HTML failu izmērus, arī samazinot tajos esošus JavaScript un CSS kodus. Tas ir paredzēts parastiem HTML tipa failiem, tāpēc, lai varētu izmantot to failos ar PHP kodu (.tpl), ir jāizmanto atsevišķs mainīgais, lai PHP kods tiktu ignorēts.

#### 3.5.1.1. Uzdevuma uzbūve .tpl failiem

Vairāk gulp faila kodu var apskatīt 1. pielikumā

```
 return gulp.src(['template/**/*.tpl'])
        .pipe(htmlmin({
             collapseWhitespace: true,
             minifyJS: true,
             minifyCSS: true,
             removeComments: true,
             ignoreCustomFragments: [ /<%[\s\S]*?%>/,
/<\?[=|php]?[\s\S]*?\?>/ ],
          }))
          .pipe(gulp.dest('../' + templateDirName + '/template/'))
        .pipe(browserSync.stream());
```

#### 3.5.1.2. Mainīgo dokumentācija

**collapseWhitespace** – vai izdzēst liekās atstarpes kodā;

**minifyJS** – vai samazināt JS;

**minifyCSS** – vai samazināt CSS;

**removeComments** – vai izņemt visus komentārus. Tie gala produktam nav nepieciešami;

**ignoreCustomFragments** – regex, kur tiek aprakstīts, kurš kods tiks ignorēts samazināšanas procesā.

## 3.5.2. Pieprasīto CSS un JavaScript failu samazināšana un apvienošana

Veidojot programmatūru, daļa no funkcionalitātes tiek iegūta no ārējiem resursiem. Tāpat ir faili, kas neatrodas failu struktūrā, kur darbojas Gulp. Šādos gadījumos arī ir nepieciešams, lai faili būtu samazināti un apvienoti, lai tiktu veikti mazāk pieprasījumi uz serveri pēc failiem.

### 3.5.2.1. Metodes dokumentācija

- (viod) check_and_minimize (sīkāk skatīt 2. pielikumā)

Funkcijas izsaukumi atrodas gan galvenē, gan kājenē. Izmanto opencart metodes “getStyle” un “getScript”.

Parametri:

- &data – datu masīvs, kur atrodas visa informācija, ko Controller ir izveidojis. Tiek padots ar referenci, lai varētu viegli modificēt esošo informāciju;
- type – nosaka vai tiks apstrādātas JavaScript hipersaites vai CSS hipersaites.

### 3.5.2.2. Mainīgo dokumentācija

**$internal_links, $external_links masīvs** – izmantoti, lai atdalītu ārējos pieprasījumus no failiem, kuri atrodas uz servera;

**$positions masīvs** – divas sadefinētas vērtības. Izmantotas, lai nodrošinātu koda nedublicēšanos;

**${'minimizer_' . $position}** – tiek piešķirta JavaScript vai CSS klases instance;

**${$position . '_hash'}** – tiek glabāts failu masīva hipersaišu jaucējkods;

**${$position . '_path'}** – nosaka faila nosaukumu.

## 3.5.3. Lapas saglabāšana kešatmiņā

Sākumlapā ir ievietoti daudz produkti, un visi tiek ielādēti veicot garus pieprasījumus datubāzei, pagarinot lapas ielādi līdz pat 70 sekundēm. Ar lapas sastāva ievietošanu kešatmiņā var panākt, ka vairs netiek veikti iepriekšējie pieprasījumi uz datubāzi, bet ielasīti no

failsistēmas. Lai varētu failus saglabāt kešatmiņā, tiek izmatota autora izveidota “Cache” klase, kurā ir iespēja manipulēt kešatmiņu.

#### 3.5.3.1. Metožu dokumentācija

- (String) get

Tiek iegūta kešatmiņas faila informācija

Parametri:

- key – kešatmiņas faila identifikators.

- (Void) set

Nepieciešamā informācija tiek saglabāta ar noteiktu derīguma laiku, papildus līdz 3 minūtēm, lai nebūtu tik daudz situāciju, kad ir nepieciešams apstrādāt visu informāciju reizē.

Parametri:

- key – kešatmiņas faila identifikators;
- value – vērtība, kuru ir nepieciešams ierakstīt failā., piemēram, ar visu informāciju savadīts home view HTML.

# 4. Projekta organizācija

Projekta ietvaros tika izmantota spējas programmatūras izstrādes metode [5]. Šīs metodes svarīgākā īpašība ir tas, ka prasības tiek izvirzītas regulāri, un esošās prasības ir iespējams izlabot vai pilnveidot. Šī metode ir piemērota šādam projektam, jo programmatūras izstrāde tiek veikta ātrāk un problēmu risināšana ir vieglāka.

Minimālās sistēmas prasības tika formulētas projekta sākumā pēc klienta vēlmēm.

Organizēšanai tika izmantots "Redmind", kas ir projektu vadības rīks un kurā tika aprakstīti tekošie un ieplānotie projekta uzdevumi [6]. Tas arī kalpoja kā projekta dokumentēšanas rīks, jo tajā tika aprakstīta iepriekšējo uzdevumu gaita.

Pēc katra uzdevuma izpildes tika veikta produkta testēšana ar akcepttestu palīdzību. Tas bija īpaši svarīgi, jo, pēc katra uzdevuma paveikšanas, tas tika publicēts.

Autors darba plānošanā konsultējas ar uzņēmuma galveno programmētāju, kurš konsultēja par sistēmas uzbūvi un optimālāku metožu izvēli. Darba programmatūras programmēšanā piedalījās divi programmētāji – autors un projekta vadītājs.

Autors izstrādāja šādas programmatūras daļas: PHP, JS programmēšanu, interneta veikala funkcionalitātes testēšanu un dokumentēšanu.

# 5. Konfigurācijas pārvalde

Priekš versiju kontroles tika izmantota Git pārvaldības sistēma [7]. Git direktorijs ir pilnvērtīgs repozitorijs ar pilnu versiju vēsturi, kas ir neatkarīgs no tīkla piekļuves vai centrālā servera. Šī sistēma tika galvenokārt izmantota versiju kontrolei un pārceltu izmaiņas no darba vides uz galveno lapu. Tā kā abi programmētāji strādāja uz viena servera, nebija problēma ar to, ka kāda programmētāja darba kods tiktu izmanīts pietiekami, lai būtu konflikti. Izmantojot Git, arī bija iespēja izmantot "zarus"(Branch), kuros ir iespēja veikt lielākas programmatūras izmaiņas, nesabojājot pamatprogrammas funkcionalitāti.

Pēc uzdevuma paveikšanas un akcepttestu izpildes, izmaiņas tika saglabātas Git repozitorijā, kas ļauj viegli sekot līdz programmatūras izmaiņām. Programmatūras izmaiņas tika saglabātas vismaz vienu reizi dienā.

# 6. Kvalitātes nodrošināšana

Lai tiktu nodrošināts augsts programmatūras kvalitātes līmenis, pēc katra uzdevuma paveikšanas tika veikti akcepttesti. Kā arī pēc katras programmatūras labošanas vai papildināšanas testi tika atkārtoti, lai nodrošinātu ka programmatūra turpina strādāt korekti. Kad projekta izstrāde bija pabeigta, tika veikta programmas veiktspējas tests. Testā tika izmantots Google PageSpeed Insights, kurš veic lapas pieprasījumus un novērtē lapas ielādes ātrumu un lapas optimizāciju [8].

Google PageSpeed Insights parāda lietotājiem redzamākos rādītājus:

- FCP, jeb First Contentful Paint, kas nozīmē - cik ilgā laikā lietotājam parādās pirmā vizuālā informācija, kas nav balta lapa.
- DCL, jeb DOM Content Loaded, kas nozīmē - cik ilgā laikā tiek ielādēts HTML un tas tiek noparsēts.

Programmatūras kļūdu gadījumos informācija tiek rakstīta teksta failā, kurš ir uz servera. Tajos tiek aprakstīts, kurā lapā ir bijusi kļūda un kas kļūdu izraisīja.

Lai nodrošinātu programmatūras uzturamību, efektivitāti un pielāgojamību tika izmantoti programmatūras labās prakses ieteikumi [9]. Viens no tiem ir metožu nosaukumus nosaukt vārdos, kas ir sevis izskaidrojoši, piemēram,
*public function getProduct($product_id)*, kur var redzēt, ka funkcijai tikai tiek padots produkta identifikātors un atgriezta produkta informācija.

# 7. Darbietilpības novērtējums

Programmatūras izstrādei balstoties uz spējas izstrādes metodēm, tika iegūts darbietilpības novērtējums, ņemot vērā autora pieredzi veicot kādu no uzdevumiem. Izstrādei veltītais laiks tiek noteikts ar sarežģītības punktiem. Viens punkts ir salīdzināms ar kādu laika vienību. Katram uzdevumam ir savs punktu skaits, kas norāda dažādas uzdevumu darbietilpības.

Viens sarežģītības punkts šajā projektā ir pielīdzināms vienai programmēšanas dienai. No tā tika izvērtēta katra iterācija. Iegūtais iterāciju dienu skaits pēc aprēķina tika vēl sareizināts ar 0.6, lai sniegtu reālistiskāku skaitli. Reizinājums ir tāpēc, ka bija darbi, kas neaizņēma visu dienu, bet autors pārskatāmības dēļ ievietoja apaļus skaitļus.

*7.1.1 tabula*

**Veltāmais dienu skaits iterācijai pirms izstrādes**

| Iterācija | Dienu skaits |
|---|---|
| 1. | 13 |
| 2. | 11 |
| 3. | 6 |
| 4. | 12 |
| 5. | 3 |
| 6. | 14 |

Projekts tika izstrādāts 2 dienas pēc plānotajām 59 dienām. Programmatūras izstrādes procesā ļoti svarīgi bija, lai viss strādātu korekti, kas nozīmē, ka bija nepieciešams vairāk laiks, lai veiktu testus.

*7.1.2 tabula*

**Veltāmais dienu skaits iterācijai pēc izstrādes**

| Iterācija | Dienu skaits |
|---|---|
| 1. | 12 |
| 2. | 11.4 |
| 3. | 6.6 |
| 4. | 15.6 |
| 5. | 4.2 |
| 6. | 11.4 |

## Rezultāti

Laika periodā, kad tika rakstīts kvalifikācijas darbs, tika izveidots interneta veikals atbilstoši klienta prasībām. Tika izvēlēta pareizā programmatūras izstrādes metode, jo tā deva iespēja komfortabli organizēt darbu un veikt nepieciešamās izmaiņas pēc sanāksmēm ar klientiem.

Izstrādājot programmatūru, autors guva dziļāku ieskatu spējas programmēšanas principos. Kā arī tika nostiprinātas PHP, JavaScript, CSS, konfigurācijas pārvaldības zināšanas.

## Secinājumi

Pēc interneta veikala izstrādes autors secina, ka būtiski ir uzlabojusies lapas ātrdarbība un pilnveidots izstrādes process. Izstrādes modelis, kurš tika izvēlēts programatūras izstrādei, būtiski uzlaboja izstrādes organizāciju, atļaujot viegli apkopot klientu vēlmes un pēc nepieciešamības veikt izmaiņas. Arī iespējamās domstarpības ar klientiem tika novērstas šīs metodes dēļ, jo tika nodrošināta regulāra tikšanās. Gala rezultāts sakrita ar klientu vēlmēm, un viņi bija apmierināti ar darba gaitu.

Viena no lielākajām problēmām, ar ko saskārās autors, bija ar Gulp HTML samazināšanas rīku. Problēma radās no tā, ka šis rīks izdzēš arī svarīgu informāciju kā  tagu. Ja kāds no šiem tagiem ir pazudis no lapas, sabrūk visa lapas uzbūve. Interneta pārlūki cenšas aizvērt neaizvērtos tagus, bet tie nevar zināt, kur tas ir nepieciešams. Tāpēc bija nepieciešams tagus, kuri tika izdzēsti, izvadīt ar PHP kodu, kas nodrošināja, ka HTML uzbūve būs korekta.

Autors secina, ka ir guvis lielu pieredzi interneta veikalu izstrādē, kas darbojas uz Opencart. Izstrādātais interneta veikals atbilst klientu vēlmēm un jau ir publicēts. Interneta vietņu platforma ir ļoti plaša un iespējām pilna, tāpēc, ja nākotnē būs nepieciešama papildus funkcionalitāte, to būs iespējams pievienot.

## Izmantotā literatūra un avoti

[1] “Microsoft” “CSS Compatibility in Internet Explorer.”
https://msdn.microsoft.com/en-us/library/hh781508(v=vs.85).aspx
[2] “Google” “Minify Resources (HTML, CSS, and JavaScript)”
https://developers.google.com/speed/docs/insights/MinifyResources
[3] “TheSEMPost” “Google: How Bounce Rate Increases With Longer Mobile Load Times”
http://www.thesempost.com/google-bounce-rate-increases-longer-load-times/
[4] “Internet Alchemy” “What Are The Benefits of MVC?”
http://blog.iandavis.com/2008/12/what-are-the-benefits-of-mvc/
[5] John Favaro, "Agile", in "Computing Handbook, Vol.1", 2014, pp.84-1
- 84-28.
[6] “Redmine”
https://www.redmine.org/
[7] “Atlassian” “What is Git”
https://www.atlassian.com/git/tutorials/what-is-git
[8] “Google” “PageSpeed Tools Insights Frequently Asked Questions”
https://developers.google.com/speed/docs/insights/faq#speedmetrics
[9] “envfatotuts+” “Top 15+ Best Practices for Writing Super Readable Code”
https://code.tutsplus.com/tutorials/top-15-best-practices-for-writing-super-readable-code--net-8118
[10] “Salidzini.lv”
https://www.salidzini.lv/shops_info.php
[11] “Kurpirkt”
http://www.kurpirkt.lv/veikaliem.php
[12] “Section.io” “How Page Load Time Affects Bounce Rate and Page Views”
https://www.section.io/blog/page-load-time-bounce-rate/
[13] “Google” “PageSpeed Insights”
https://developers.google.com/speed/pagespeed/insights/
[14] “Fortune” “Consumers Are Now Doing Most of Their Shopping Online”
http://fortune.com/2016/06/08/online-shopping-increases/
[15] “Facebook” “Easily sell to buyers nearby.”
https://www.facebook.com/marketplace/learn-more/selling

# Pielikumi

## 1. Pielikums. Gulp rīka kods

```
var gulp = require('gulp'),
    gutil = require('gulp-util'),
    env = require('./_env.json'), // ielasām environment mainīgos.
    sass = require('gulp-sass'),
    sourcemaps = require('gulp-sourcemaps'),
    rename = require("gulp-rename"),
    concat = require('gulp-concat'),
    uglify = require('gulp-uglify'),
    clean = require('gulp-clean'),
    buffer = require('gulp-buffer'),
    merge = require('gulp-merge'),
    cached = require('gulp-cached'),
    debug = require('gulp-debug'),
    htmlmin = require('gulp-htmlmin'),
    browserSync = require('browser-sync').create();

// Tēmas direktorija
var templateDirName = env.theme_folder;

// CSS
// CSS SRC
var cssSrcDir = 'scss/**/*.scss';
// CSS DEST
var cssDirName = 'styles';
var cssName = 'main.css';


// JS direktorija, kur atrodas stili un js faila nosaukums
var jsName = 'main.js'; // js Destination fails
var jsNamePlugins = 'plugins.js'; // js Destination fails
var jsDir = 'scripts'; // js Destination direktorija

// Faili, failu ceļi
var paths = {
    script_plugins: ['js/jquery.fancybox.js', 'js/modernizr.js',
'bower_components/slick-carousel/slick/slick.js'],
    script_main: ['scripts/**/*.js'],
    styles: [cssSrcDir]
};

bconfig = {
    notify: false, // In-line notifications (the blocks of text saying whether you
are connected to the BrowserSync server or not)
    open: true, // Set to false if you don't like the browser window opening
automatically
    port: 3000, // Port number for the live version of the site; default: 3000
    proxy: (env.proxy || 'http://localhost'),
    watchOptions: {
        debounceDelay: 100, // This introduces a small delay when watching for file
change events to avoid triggering too many reloads
        reloadDelay: 100
    }
}

gulp.task('serve', ['build'], function () {
    gutil.log('Palaidies serve!');
    browserSync.init(bconfig);
    gulp.watch('template/**/*.tpl', ['templates']);
    gulp.watch('stylesheet/**/*.css', ['styles']);
    gulp.watch(cssSrcDir, ['styles']);
    gulp.watch('src/sprites/*.png', ['sprites', 'styles']);
```

```
});

// KOMPILĒJAM STILUS
gulp.task('styles', function () {
    gulp.src('scss/**/*.scss')
        .pipe(sourcemaps.init()) // Process the original sources
        .pipe(sass({precision: 16}).on('error', sass.logError))
        .pipe(sourcemaps.write()) // Add the map to modified source.
        .pipe(rename(cssName)) // šis pārsauc failu tā, kā mums vajag
        .pipe(cached('styles'))
        .pipe(gulp.dest('../' + templateDirName + '/' + cssDirName))
        .pipe(browserSync.stream());


    gulp.src('stylesheet/**/*.css') // Kopējam templeitus
        .pipe(cached('stylesheet'))
        .pipe(gulp.dest('../' + templateDirName + '/stylesheet/'))
        .pipe(browserSync.stream());
});


// KOPĒJAM TEMPLEITUS
gulp.task('templates', function () {

    gulp.src('template/functions.php') // Kopējam functyions
        .pipe(cached('functions'))
        .pipe(gulp.dest('../' + templateDirName + '/template/'));

    gulp.src(['template/**/*.tpl'])
        .pipe(debug({title: 'unicorn:'}))
        .pipe(htmlmin({
            collapseWhitespace: true,
            minifyJS: true,
            minifyCSS: true,
            html5: false,
            removeComments: true,
            ignoreCustomFragments: [ /<%[\s\S]*?%>/, /<\?[=|php]?[\s\S]*?\?>/ ],
        }))
        .pipe(gulp.dest('../' + templateDirName + '/template/'))
        .pipe(browserSync.stream());


gulp.src(['../../../../system/storage/modification/catalog/view/theme/src/template/
**/*.tpl'])
        .pipe(htmlmin({
            collapseWhitespace: true,
            ignoreCustomFragments: [ /<%[\s\S]*?%>/, /<\?[=|php]?[\s\S]*?\?>/ ],
            minifyJS: true,
            minifyCSS: true
        }))

.pipe(gulp.dest('../../../../system/storage/modification/catalog/view/theme/' +
templateDirName + '/template/'))

.pipe(gulp.dest('../../../../system/storage/modification/catalog/view/theme/default
/template/'))
        .pipe(browserSync.stream());


    gulp.src(['../../../../extensions/**/src/**/*.tpl'])
        .pipe(htmlmin({
            collapseWhitespace: true,
            ignoreCustomFragments: [ /<%[\s\S]*?%>/, /<\?[=|php]?[\s\S]*?\?>/ ],
            minifyJS: true,
            minifyCSS: true
        }))
        .pipe( rename ( function(file){ file.dirname =
file.dirname.replace('/src/','/catalog/'); console.log ( file.dirname ) ;  }))
```

```
        .pipe(gulp.dest('../../../../extensions/'))
        .pipe(browserSync.stream());

    return gulp.src(['template/functions.php' ]) // Kopējam templeitus
        .pipe(gulp.dest('../' + templateDirName + '/template/'))
        .pipe(browserSync.stream());

});

// kopējam visu, ko vajag pārkopēt uz dest
gulp.task('copy', function () {
    gulp.src('images/**/**') // Kopējam Bildes
        .pipe(cached('images1'))
        .pipe(gulp.dest('../' + templateDirName + '/assets/images'));
    gulp.src('fonts/**/**') // Kopējam Fontus
        .pipe(cached('fonts1'))
        .pipe(gulp.dest('../' + templateDirName + '/assets/fonts'));

    gulp.src('stylesheet/fonts/*') // Kopējam Bootstrap fontus, dēļ Bootstrap
stiliem, fontiem jāatrodas šajā ceļā.
        .pipe(cached('fonts'))
        .pipe(gulp.dest('../' + templateDirName + '/stylesheet/fonts'));

    gulp.src('scripts/**/**')             // Kopējam Skriptus
        .pipe(cached('scripts'))
        .pipe(gulp.dest('../' + templateDirName + '/scripts/'));
});

gulp.task('build', ['styles', 'templates', 'copy']);
```

## 2. Pielikums. Failu apvienošanas un samazināšanas funkcija

- Nepieciešmās nosaukumvietas imports

```
use MatthiasMullie\Minify;
```

- Metodes izsaukšana

```
$this->check_and_minimize($data, 'Styles'); // Samazina JS faili
$this->check_and_minimize($data, 'Scripts'); // Samazina CSS failus
```

- Metodes kods

```
function check_and_minimize(&$data, $type){
       $extension =  $type == 'Scripts' ? 'js':'css';

       $internal_links = $external_links = [];
       foreach( $this->document->{"get$type"}('header') as $link) {
           $link_check = $type == 'Scripts'  ? $link:$link['href'];
           if(strpos( $link_check , "//") === false) {
               $internal_links[] = DIR_APPLICATION . "../" . $link_check;
           } else {
               $external_links[] = $link_check;
           }
       };

       $positions = ['internal','external'];

       foreach ($positions as $position){
       ${'minimizer_' . $position} =  $type == 'Scripts' ? new Minify\JS() : new
Minify\CSS();

           ${$position . '_hash'} = md5(json_encode(${$position . '_links'}));
           ${$position . '_path'} = "catalog/view/theme/minimised/" .
strtolower($type) . "/" . $type . "_" . $position . "-" . ${$position . '_hash'}
."."$extension";

           if(!file_exists(${$position . '_path'})){
               foreach (${$position . '_links'} as $link){
                   if($position == 'internal'){
                   ${'minimizer_' . $position}->add($link);
                   } else {
                   ${'minimizer_' . $position}->add(file_get_contents($link));
                   }
               }

               ${'minimizer_' . $position}->minify(${$position . '_path'});
           }
       $data[strtolower($type)][] = ${$position . '_path'};
       }
   }
```

Kvalifikācijas darbs „Tīmekļa vietnes priekšgalsistēmas ātrdarbības uzlabošana mērogojamības nodrošināšanai” izstrādāts Latvijas Universitātes Datorikas fakultātē.

Ar savu parakstu apliecinu, ka darbs izstrādāts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors:_____ Jēkabs Jānis Kalniņš

Rekomendēju darbu aizstāvēšanai

Darba vadītājs: Dr. dat. Uldis Straujums _____ 28.05.2018.

Recenzents: Dr.dat. Mārtiņš Zviedris

Darbs iesniegts 28.05.2018.

Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: _____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē

05.06.2018. prot. Nr. _____

Komisijas sekretārs(–e): _____