LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

AUTOMĀTISKA TĪMEKĻA VIETŅU STILA ĢENERĒŠANA

MAĢISTRA DARBS

Autors: **Jānis Ozoliņš**
Stud. apl. nr.: jo11022
Darba vadītājs: Dr.dat. Uldis Bojārs

RĪGA 2017

ANOTĀCIJA

Individuālas tīmekļa vietnes izstrāde ir sarežģīts process, kam nepieciešams daudz resursu, laika un specifisku zināšanu, taču internetā pieejamie rīki ne vienmēr spēj nodrošināt unikālu, vajadzībām atbilstošu risinājumu. Lai samazinātu nepieciešamos resursus un izstrādes laiku, tiek piedāvāta sistēma, kura tīmekļa vietnes stilu ģenerē, par pamatu izmantojot šīs vietnes saturu.

Darba mērķis ir izstrādāt tīmekļa vietnes stila ģenerēšanas prototipu, izpētot literatūru un esošos risinājumus, lai atrastu pieeju un rīkus, uz kuriem balstīt risinājuma un prototipa izstrādi.

Darba rezultātā tika veikts literatūras apskats, esošo risinājumu izpēte un izstrādāts risinājums tīmekļa vietnes stila un izkārtojuma ģenerēšanai. Balstoties uz rasto risinājumu, tika izstrādāts sistēmas prototips, kas veic minētās funkcijas.

ATSLĒGVĀRDI: tīmekļa vietne, stils, izkārtojums, ģenerēšana, satura analīze

ABSTRACT

Automatic website style generation

Development of an individual website is a complex process that requires resources, time and specific knowledge; available tools are not always able to provide unique and customized solutions. To reduce the necessary resources, a system concept that performs website style generation on the basis of website content is proposed.

The aim of the Thesis is to develop a solution and prototype of website style generation through a literature review and a research of existing solutions in order to assess an approach and tools to use in the development process of the solution and prototype.

As a result of this research, literature review and existing solutions were analyzed, and a solution for website style generation was developed. Based on the proposed solution, a prototype system that provides previously mentioned functions was developed.

KEYWORDS: website, style, layout, generation, content analysis

AUTOREFERĀTS

Darba rezultāts ir sistēma, kura ģenerē tīmekļa vietnes stilu un izkārtojumu, izmantojot tās saturu. Sistēmas pamatā ir moduļu bibliotēka, kas sastāv no neatkarīgiem moduļiem, kurus savstarpēji kombinējot tiek iegūts unikāls tīmekļa vietnes izkārtojums un stils. Sistēma izmanto satura analīzi, lai nodrošinātu pēc iespējas labāku tīmekļa vietnes stila un izkārtojuma ģenerēšanas rezultātu.

Darba ietvaros tika veikta literatūras izpēte, pēc kuras apkopotas svarīgākās pieejas, tehnoloģijas un rīki, kuri izmantoti sistēmas prototipa izstrādē. Darbs ir vairāk praktisks, tika pielietoti jaunākie rīki un pieejas, par ko zinātniskajā literatūrā ir mazāk informācijas. Lielākā daļa darba zināšanu bāzes iegūta nozares specifikas un pieejamo rīku dokumentācijā.

Darbā detalizēti aprakstītas pamata tehnoloģijas un pieejas, kuras izmantotas risinājuma un tā prototipa izveidē. Darbā veikts līdzīgu sistēmu apraksts, lai veidotu priekšstatu par pieejamo risinājumu iespējām. Darba gaitā izstrādātais risinājums ir aprakstīts 3 dažādos līmeņos. Vispirms risinājums aprakstīts jēdzieniskā un teorētiskā sistēmas uzbūves līmenī. Pēc tam raksturotas atsevišķas sistēmas komponentes, uzsverot izstrādi; visbeidzot parādīts risinājuma prototips, tā funkcijas un izmantotās tehnoloģijas.

Aptuveni puse darba apjoma ir risinājuma koncepta izstrāde, kuras gaitā tika pētītas līdzīgas sistēmas un rīki, ko izmantot, veidojot prototipu. Otra puse darba tika ieguldīta praktiskajā daļā, kurā tika veikta risinājuma prototipa izstrāde. Darba izklāstā atspoguļots, kā prototips tika attīstīts, mainīts un darba gaitā uzlabots, lai izpētītu konceptu un tehnoloģijas reālos dzīves apstākļos.

Darba mērķis ir sasniegts, tika atbildēts uz visiem pētījuma jautājumiem, kā arī atrisināta pētījumā izvirzītā problēma. Darba ietvaros tika izstrādāts tehnisks prototips, kurš nav pilnībā pabeigts un gatavs reālai lietošanai, tāpēc to ir grūti salīdzināt ar līdzīgām sistēmām. Prototips darba rakstīšanas brīdī netiek izmantots praksē, bet autoram ir vēlme to pilnveidot un lietot ikdienas vajadzībām.

Darbs ir noformēts, vadoties pēc LU DF *Maģistra darba izstrādes un aizstāvēšanas metodiskajiem norādījumiem* un atbilst tajā atrodamajām norādēm un vadlīnijām. Darba teksta izstrādē izmantotas Microsoft Word sniegtās iespējas un pareizrakstības rīki, kā arī veikta darba teksta vairākkārtēja caurskatīšana un atrasto kļūdu labošana..

Visi darbā izmantoti resursi un atsauces ir norādītas literatūras sarakstā, un attiecīgās atsauces ir ievietotas darba pamattekstā. Visi darbā izmantotie attēli un koda paraugi ir autora paša sastādīti un radīti. Darbā izmantotās shēmas un ilustrācijas radītas ar Microsoft Visio programmatūru.

SATURS

APZĪMĒJUMU SARAKSTS

Apzīmējums	Apraksts
3D (*Three-dimensional space*)	Trīsdimensiju telpa
ADI (*Artificial Design Intelligence*)	Mākslīgais dizaina intelekts
AI (*Artificial Intelligence*)	Mākslīgais intelekts
API (*Application Programming Interface*)	Lietotņu programmēšanas saskarne
CCS (*Cassowary Constraint Solver*)	Cassowary ierobežojumu risinātājs
CCSS (*Constraint Cascading Style Sheets*)	Ierobežojumu kaskādes stila lapas
Clickbait	Saturs kura mērķis ir piesaistīt lietotāja uzmanību
CMS (*Content Management System*)	Satura vadības sistēma
CORS (*Cross-Origin Resource Sharing*)	Krusteniskās izcelsmes resursu koplietošana
CSS (*Cascading Style Sheets*)	Kaskādes stilu lapas
Čatbots (*chatbot*)	Programmatūra ar kuru var veikt saraksti
DOM (*Document Object Model*)	Dokumenta objektu modelis
GSS (*Grid Style Sheets*)	Režģa stilu lapas
HTML (*HyperText Markup Language*)	Hiperteksta iezīmēšanas valoda
JS (*JavaScript*)	JavaScript programmēšanas valoda
JSON (*JavaScript Object Notation*)	JavaScript objektu notācija
Miksini (*Mixins*)	Klase, kura satur metodes, kuras var izmantot citas klases nemantojot vecāka klasi
MVC (*Model View Controller*)	Koda arhitektūras tips
NER (*Named Entity Recognizer*)	Nosaukto entītiju atpazinējs
NoSQL	Bezrelāciju datu bāzu pārvaldības sistēma
OAuth2	Atvērts autorizācijas standarts
OCR (*Optical Character Recognition*),	Rakstzīmju optiskā pazīšana
Polyfill	Kods, kurš nodrošina specifisku tīmekļa pārlūka funkcionalitāti

PHP	PHP programmēšanas valoda
RAKE (*Rapid Automatic Keyword Extraction*)	Ātrā automātisko atslēgvārdu izgūšana
RC (*Right Click*)	Labais peles klikšķis
REST (*REpresentational State Transfer*)	Reprezentācijas stāvokļu pārvietošana
SEO (*Search Engine Optimization*)	Meklēšanas dzinēju optimizācija
UI (*User Interface*)	Lietotāja saskarne
URL (*Uniform Resource Locator*)	Vienotais resursu vietrādis
VFL (*Visual Format Language*)	Vizuālā formāta valoda
W3C (*World Wide Web Consortium*)	Organizācija

IEVADS

Pielāgotas tīmekļa vietnes izstrāde ir process, kam vajadzīgi daudzi resursi: laiks, līdzekļi, specifiskas zināšanas un speciālistu iesaiste. Tīmekļa vietnes izstrādes mērķis ir gala sistēma, kura atbilst klienta vēlmēm, apmierina tā biznesa prasības un sniedz pievienoto vērtību klienta idejai vai produktam. Ir atrodami vairāki risinājumi, kuri piedāvā ātri un vienkārši izveidot tīmekļa vietni, bet šīs iespējas ir standartizētas, tātad grūti pielāgojamas, līdzīgas citām tīmekļa vietnēm un ne vienmēr spēj apmierināt visas klienta prasības. Tāpēc nereti nākas izvēlēties starp kvalitatīvu, bet dārgu vai lētu, ātru, bet grūti pielāgojamu un ierobežotu risinājumu.

Iespēja, kā samazināt patērēto laiku un izmaksas, tajā pašā laikā saglabājot kvalitāti un vietnes unikalitāti, ir risināt problēmu ar automatizācijas palīdzību. Automatizācija jau sen nav jauns jēdziens, tā ir ienākusi teju katrā dzīves aspektā un jomā, jo tā ir efektīvs veids, kā paveikt apjomīgu darbu, saīsinot patērēto laiku un samazinot izmaksas. Arī tīmekļa vietņu izstrādē automatizācija nav jaunums, automātiska testēšana un satura veidošana jau labu laiku tiek izmantota praksē, taču visa izstrādes procesa nodošana automatizētas sistēmas rokās ir salīdzinoši jauna ideja.

Viens no problēmas risinājumiem ir veikt tīmekļa vietnes ģenerēšanu automātiski, vadoties pēc vietnes satura. Tīmekļa vietnes stila un izkārtojuma ģenerēšana nodrošina efektīvu vietnes izstrādi, maksimāli ietaupot tās izveidei nepieciešamos resursus. Par pamatu lietojot elementu bibliotēku, kuras elementu savstarpējās kombinācijās iespējams iegūt arvien jaunu vietnes stilu, kā arī izmantojot satura analīzes iespējas, ir iespējams radīt sistēmu, kura ir gan viegli lietojama, gan nodrošina ātru un unikālas tīmekļa vietnes izstrādi.

Darba galvenais mērķis ir izpētīt un piedāvāt sistēmas konceptu, kas spēj ģenerēt tīmekļa vietnes stilu un izkārtojumu, balstoties uz tās saturu. Lai pārbaudītu koncepta realizācijas iespējas, tiek izvirzīts sekundārs mērķis – sistēmas tehniskā prototipa izstrāde.

Darbs sadalīts divās daļās. Pirmā daļa ir teorētisks apraksts, kurā ietverta literatūras analīze, esošo risinājumu izpēte un pieejamo tehnoloģiju apskats. Otrā daļa ir praktisks pētījums, kura mērķis ir piedāvāt sistēmas konceptu un tās prototipu.

Teorētiskās daļas mērķi ir:

- Literatūras izpēte, lai uzlabotu zināšanas par tīmekļa vietnes dizaina ģenerēšanu, pamata tehnoloģijām un pieejamajiem rīkiem.
- Esošo risinājumu izpēte, to piedāvāto iespēju un problēmu izvērtēšana ar mērķi gūt zināšanas piedāvātās sistēmas izveidei.
- Piedāvāt risinājumu tīmekļa vietnes stila ģenerēšanai, kurš vismaz daļēji rastu atbildes esošajām problēmām.

Praktiskās daļas mērķi ir:

- Izveidot tīmekļa vietnes stila ģenerēšanas sistēmas un tas sastāvdaļu aprakstu.
- Izstrādāt piedāvātās sistēmas tehnisko prototipu, lai pārbaudītu risinājuma realizēšanas iespējas reālos dzīves apstākļos.
- Aprakstīt gūtos rezultātus un secinājumus.

Darbā ir 7 nodaļas. Problēmas apraksts un ieskats tēmā sniedz nepieciešamās zināšanas par problēmas vidi un tās risināšanu. Esošo risinājumu izpēte sniedz ieskatu par tehnoloģijām un rīkiem. Pētījuma metodoloģijā tiek definēti pētījuma jautājumi, aprakstā izmantotā metodoloģija un uzdevumi, kuri ir svarīgi pētījuma izstrādē. Tālākajās nodaļās tiek aprakstīts piedāvātais risinājums un prototipa izstrāde. Rezultātu nodaļa attēlo darba laikā gūtos rezultātus. Pēdējā nodaļa satur rezultātu analīzi un idejas tālākam darbam tīmekļa vietņu ģenerēšanas jomā.

1. PROBLĒMAS APRAKSTS

Nodaļā sniegts pārskats par problēmām, ar kurām nākas saskarties tīmekļa vietnes izstrādes procesā. Tiek piedāvāta sistēma, kura risinātu šīs problēmas, kā arī aprakstīti šķēršļi, kas ierosinātās sistēmas realizēšanas gaitā jāpārvar. Nodaļa veidota, lai veicinātu lasītāja izpratni par darba izstrādes motivāciju un problēmām, ar kurām jāsaskaras, projektējot un īstenojot minēto sistēmu.

1.1. Problēmas ar tradicionālu tīmekļa vietnes izstrādes procesu

Internets kļuvis par pasaules vadošo mediju tā straujās attīstības dēļ. Tas sniedz jaunas iespējas, kā sasniegt cilvēkus jebkurā pasaules malā. Internets ir spēcīgs rīks, kuru izmantot saviem personīgajiem mērķiem, tādiem kā biznesa attīstība vai dzīves dokumentēšana. Diemžēl tīmekļa vietņu izstrāde ne vienmēr atbilst piedāvātajām iespējām un ātrajam attīstības līmenim. Katrai idejai var būt nepieciešama tīmekļa vietne vienā vai otrā formātā, bet ne vienmēr tās iegūšana ir viegls un patīkams process. Profesionālas tīmekļa vietnes izstrāde ir dārga, sarežģīta, aizņem daudz laika un bieži vien rezultāts var nebūt līdzekļu vērts. Pieejami arī dažādi gatavi risinājumi, kuri tāpat ne vienmēr spēj sniegt gaidāmos rezultātus. Internetā sastopamās sistēmas ātru tīmekļa vietņu izveidei bieži ir grūti pielāgojamas, īpaši neizceļas un nespēj konkurēt ar profesionāli izstrādātām tīmekļa vietnēm.

Katram, kurš nolēmis izveidot savu tīmekļa vietni (biznesam vai privātām vajadzībām) nākas saskarties ar problēmām, kuras tīmekļa vietnes izstrādes procesu padara dārgu, sarežģītu vai prasībām neatbilstošu. Veidojot tīmekļa vietni, nereti nākas saskarties ar šādām problēmām:

- Tīmekļa vietnes izstrāde prasa daudz resursu, tā ir dārga un/vai laikietilpīga.
- Iesācējam ir grūtības izveidot savu tīmekļa vietni, jo nepieciešamas atbilstošas zināšanas par vietnes izstrādi.
- Izveidot jaunu dizainu esošai tīmekļa vietnei ir sarežģīti un dārgi.
- Gatavie tiešsaistes risinājumi ne vienmēr spēj apmierināt visas vēlmes un parasti nav unikāli.

Problēmas ar kurām nākas saskarties veicot tīmekļa vietnes izstrādi var būt dažādas un atkarīgas no konkrētās situācijas. Darbā tiek apzinātas problēmas ar kurām autors ir saskāries darbojoties tīmekļa vietnes izstrādes nozarē.

1.2. Tīmekļa vietnes stila ģenerēšana

Autora piedāvātais risinājums ir sistēma, kura veic tīmekļa vietnes stila un izkārtojuma ģenerēšanu, balstoties uz tīmekļa vietnes saturu. Idejas pamatā ir doma tīmekļa vietnes izstrādi automatizēt, veicot satura analīzi un unikāla stila ģenerēšanu, savstarpēji kombinējot elementus. Darba gaitā tiek veikta izpēte ar mērķi izstrādāt minētās sistēmas konceptu. Lai pārbaudītu koncepta realizēšanas iespējas, tiek izlemts veidot sistēmas prototipu. Sākotnējie centieni identificēt kritiskos tīmekļa vietnes stila ģenerēšanas atribūtus atklāja šādas problēmas:

- Tīmeklī ir pieejami daži tīmekļa vietnes ģenerēšanas risinājumi, bet nav zināmi to darbības principi, efektivitāte un tehnoloģijas, uz kuriem šie risinājumi balstīti.
- Koncepts ir salīdzinoši jauns, nav definēti konkrēti rīki, algoritmi un resursi, ar kuru palīdzību notiek vietnes stila ģenerēšana. Sastopami atsevišķi rīki, kuri varētu palīdzēt problēmas risināšanā, bet tie jāpārbauda reālos dzīves apstākļos.
- Nav vispārpieņemtu standartu un vadlīniju, pēc kurām izveidot risinājumu un izstrādāt prototipu.
- Koncepts nav plaši aprakstīts nozares specifiskos dokumentos.

Problēmu savlaicīga apzināšana ļauj vieglāk novērtēt turpmāko darbu un pētījuma strukturēšanu. Meklējot risinājumus minētajām problēmām tiek strādāts pie risinājuma izveides un pētījuma problēmas atrisināšanas.

1.3. Risinājuma izveide

Risinājuma izveidē nākas saskarties ar vairākām būtiskām problēmām, kuru pārvarēšana ir darba pamatā. Lai nodrošinātu pētījuma atbilstību un pielietojumu dzīves apstākļos, risinājums tiek meklēts reālas sistēmas ietvaros. Piedāvātajam risinājumam ir jābūt īstenojamam un jāspēj pārvarēt pētījumā definētās problēmas. Risinājums jāpēta vairākos līmeņos, lai nodrošinātu tā pilnīgu apskatu, projektējot un realizējot sistēmu. Risinājuma izveides problēmu identificēšanas procesa rezultātā tika apkopotas šādas risināmās problēmas:

- Nav aprakstītas pieejas, kā veikt tīmekļa vietnes izkārtojuma ģenerēšanu.
- Jānodrošina iespēja vienam saturam ģenerēt dažādus izkārtojumu.
- Nav zināms, kā izveidot lapas struktūru un nodrošināt to ar tīmekļa vietnes saturu.
- Jāizveido risinājums tīmekļa vietnes stila un tā atsevišķu elementu ģenerēšanai.
- Jāveic izpēte par satura analīzi, tā iespējām un izmantošanu sistēmas vajadzībām.

Minētās problēmas ir tehniska rakstura un to atrisināšana ir daļa no sistēmas izveides un prototipa izstrādes. Problēmu apzināšana ļauj sagatavoties turpmākajam darbam, veicot attiecīgu izpēti un priekšdarbus.

2. PĒTĪJUMA METODOLOĢIJA

Nodaļā aprakstīta pētījumā izmantotā metodoloģija: pētījuma problēma, jautājumi un metodes, kuras tiek izmantotas, lai atbildētu uz pētījuma jautājumiem un atrisinātu problēmu.

2.1. Problēmas formulējums

Nepieciešamība pēc risinājuma, kas veic tīmekļa vietņu dizaina ģenerēšanu, tiek pamatota ar trūkumiem klasiskajā tīmekļa vietnes izstrādes procesā. Unikālas tīmekļa vietnes izstrāde prasa daudz resursu, process ir sarežģīts, laikietilpīgs un dārgs un, lai tajā ieviestu izmaiņas, nākas atkārtoti ieguldīt resursus. Izmantojot tipveida tīmekļa vietnes, procesā vajadzīgs mazāk resursu, bet gala produkts nav unikāls, bieži ir grūti modificējams, un vietnes izveidē nepieciešamas specifiskas zināšanas. Tīmekļa vietņu dizaina ģenerēšana ir jauns, inovatīvs veids, kā lēti un ātri radīt unikālu tīmekļa vietnes stilu..

Sākotnējie centieni identificēt kritiskos tīmekļa vietnes dizaina ģenerēšanas atribūtus atklāja šādas problēmas:

PP1: ir pieejami daži gatavi risinājumi, bet nav zināmi to darbības principi, izmantotās tehnoloģijas, gala rezultāta kvalitāte un efektivitāte.

PP2: tīmekļa vietnes ģenerēšanas koncepts ir jauns, nav definētas konkrētas vadlīnijas, rīki un modeļi, kā izstrādāt šādu sistēmu.

PP3: nav zināms, cik efektīvs ir šāds risinājums, ar kādām problēmām nāksies saskarties un vai piedāvātais koncepts atrisina problēmas un nodrošina sistēmu, kura patiesi ir spējīga ģenerēt saturam atbilstošu tīmekļa vietnes stilu un struktūru.

2.2. Pētījuma jautājumi

Maģistra darba ietvaros tiek veikts pētījums par tīmekļa vietņu dizaina ģenerēšanu ar mērķi rast vismaz daļēju risinājumu nodaļā 2.1. identificētajām problēmām. Maģistra darbs sastāv no divām daļām, kurās tiek izpētīti esošie risinājumi un literatūra, un praktiskās daļas, kurā tiek izstrādāts koncepts. Pirmajā darba daļā tiek veikts literatūras apskats ar mērķi izpētīt pašreizējo situāciju un apkopot pieejamās tehnoloģijas, ar kuru palīdzību iespējams izstrādāt sistēmu tīmekļa vietņu stila ģenerēšanai. Tiek veikta esošo risinājumu izpēte, to iespējas un trūkumi. Darba otrajā daļā tiek strādāts pie sistēmas koncepta izveides, tiek aprakstītas galvenās sistēmas komponentes un koncepta pamata funkcija. Darba otrās daļas pamatā ir tehnoloģijas un rīki, kuri tika apskatīti darba pirmajā daļā, un zināšanas, kuras tika iegūtas, pētot esošos risinājumus.

Turpmākie pētījumi tiek vadīti pēc mērķiem, kuri formulēti kā pētījuma jautājumi. Pētījuma uzdevums ir atbildēt uz šiem jautājumiem. Pēc katra jautājuma tiek definēti pētījuma apakšmērķi un paskaidroti vēlamie rezultāti. Apakšmērķi un to iznākumi vēlāk izmantoti pētījumā, lai noteiktu, kādā līmenī sasniegti pamata mērķi.

Pētījuma jautājumi:

PJ1: Kādas ir pašreizējās pieejas, rīki un risinājumi tīmekļa vietņu ģenerēšanai?

Jautājuma mērķis ir izveidot stabilu pamatu zināšanām, lai veiktu turpmāku izpēti. Jautājumam ir šādi apakšmērķi:

PJ1.1: Kādi risinājumi ir izstrādāti?

PJ1.2: Kādi rīki ir pieejami?

PJ1.3: Kādas vadlīnijas ir definētas?

PJ2: Kā automātiski ģenerēt tīmekļa vietnes izkārtojumu?

Jautājuma mērķis ir identificēt izstrādes metodes, ar kurām iespējams ģenerēt tīmekļa vietnes izkārtojumu. Jautājumam ir šādi apakšmērķi:

PJ2.1: Kā notiek vietnes HTML izveide?

PJ2.2: Kā tiek salikts vietnes izkārtojums?

PJ2.3: Kā vietne tiek nodrošināta ar datiem?

PJ3: Kā automātiski ģenerēt tīmekļa vietnes stilu?

Jautājuma mērķis ir atrast veidu, kā ģenerēt tīmekļa vietnes stilu. Jautājumam ir šādi apakšmērķi:

PJ3.1: Kā tiek ģenerētas vietnes krāsas?

PJ3.2: Kā tiek atlasīti fonti?

PJ3.3: Kā tiek ģenerētas lapas malas un polsterējumi?

PJ4: Kā veikt tīmekļa vietnes satura analīzi?

Jautājuma mērķis ir apzināt satura analīzes iespējas un tās izmantot sistēmas izveidē. Jautājumam ir šādi apakšmērķi:

PJ4.1: Kā notiek teksta analīze?

PJ4.2: Kā notiek attēlu analīze?

Pētījuma gaitā tiem meklētas atbildes uz pētījuma jautājumiem paralēli veidojot sistēmas aprakstu un izstrādājot tās prototipu.

2.3. Pētījuma mērķi

Pētījumā tiek izvirzīti vairāki mērķi, kurus sasniedzot tiek atbildēts uz pētījuma jautājumiem. Definēti šādi pētījuma mērķi:

M1: Veikt literatūras izpēti, lai iegūtu padziļinātas zināšanas par tīmekļa vietnes stila ģenerēšanu, noskaidrot pamata tehnoloģijas un pieejamos rīkus.

M2: Apskatīt un salīdzināt esošos risinājumus, izvērtēt risinājumu priekšrocības un trūkumus.

M3: Ierosināt vismaz daļēju risinājumu nosauktajām tīmekļa vietnes stila ģenerēšanas problēmām.

M4: Aprakstīt ierosinātās sistēmas komponentes, veidojot sistēmas kopējo pārskatu un pamatu tās prototipa izveidei.

M5: Izveidot prototipu tīmekļa vietnes stila ģenerēšanas sistēmai.

M6: Apkopot iegūtos rezultātus un secinājumus.

Pētījuma mērķi tiek sasniegti secīgi, katra mērķa realizēšana ir pamats nākošā mērķa sasniegšanai. Visu mērķu īstenošanas rezultāts ir pētījuma jautājumu atbildes un definētās problēmas risinājums.

2.4. Pētījuma sfēra

Maģistra darbā tīmekļa vietņu stila ģenerēšana tiek analizēta no programmatūras inženierijas perspektīvas. Pētījuma centrā ir tīmekļa vietnes izkārtojuma un stila ģenerēšana. Izkārtojuma ģenerēšanas kontekstā tiek apskatīts režģa izkārtojums, modulāra vietnes struktūra, satura izkārtojums, PHP veidņu dzinējs un attēlu apstrāde. Vietnes stila ģenerēšanā apskatīta krāsu izgūšana no attēliem, fontu izvēle, malas un polsterējuma izmēri, CSS pirmsprocesori un satura analīzes iespējas. Mazāk apskatīta vietnes aizmugursistēmas izstrāde, CMS (*content management system*), datu ieguve un apstrāde.

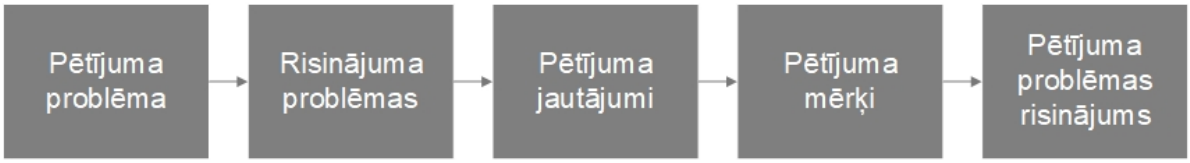
Pētījuma ietvaros tiek izstrādāts prototips, kura mērķis ir praktiski pielietot piedāvāto risinājumu reālos dzīves apstākļos, kā arī apskatīt tehnoloģijas, ar kuru palīdzību iespējams ģenerēt vietnes izkārtojumu un dizainu, lai sniegtu papildus ieskatu pētījuma risinājumā. Prototipa mērķis nav izstrādāt pilnībā pabeigtu sistēmu, bet gan demonstrēt vismaz daļu risinājuma funkciju un iespēju [36].

2.5. Pētījuma struktūra

Lai sasniegtu mērķus un rastu atbildes uz sākotnēji izvirzītajām problēmām, pētījums tiek veikts vairākos līmeņos. Tas sastāv no vairākiem soļiem; katrā tiek rastas atbildes uz pētījuma jautājumiem. Izpildot visus pētījuma soļus, tiek rastas atbildes izvirzītajām problēmām un veikti secinājumi. Pētījuma soļi:

1. **Iepazīšanās ar tēmu:** tēmas sfēras noteikšana, pamata jēdzienu noskaidrošana, kopējā darba apzināšana un pamata literatūras izpēte.
2. **Pašreizējo risinājumu izpēte:** esošo risinājumu detalizēta izpēte. Galvenais mērķis ir noskaidrot risinājumu piedāvātās funkcijas, izmantotos rīkus un praktisko pielietojumu pētījuma izstrādē, analizēt risinājumu priekšrocības un trūkumus.
3. **Pamata tehnoloģiju noskaidrošana:** galveno tehnoloģiju, kuras var tikt izmantotas pētījuma realizēšanā, izpēte. Mērķis ir noskaidrot tehnoloģiju kopumu, ar kura palīdzību iespējams atrisināt pētījumā izvirzītās problēmas.
4. **Risinājuma izveide:** piedāvāt risinājumu, kurš atrisinātu pētījuma problēmas. Piedāvātais risinājums ir tīmekļa vietnes stila ģenerēšanas sistēmas koncepts, kur tiek aprakstītas tā galvenās idejas.
5. **Prototipa izstrāde:** prototipa izstrādes veikšana, pamatojoties uz iepriekšējos soļos gūto pieredzi un informāciju. Galvenais mērķis ir likt lietā apkopoto informāciju, atrastos risinājumus un tehnoloģijas, lai izveidotu prototipu, kurš risinātu pētījuma problēmas un atbildētu uz izvirzītajiem jautājumiem.
6. **Rezultātu apkopošana un secinājumi:** pētījuma noslēguma daļa, kurā tiek atbildēts uz izvirzītajiem pētījuma jautājumiem; secinājumu veikšana, pētījuma laikā iegūtās pieredzes apkopošana un izstrādātā prototipa izvērtēšana.

Pētījums tiek strukturēts 6 atsevišķos soļos, kurus izpildot, tiek rasts pētījuma problēmas risinājums. Pētījuma soļi tiek veikti secīgi, katra soļa rezultāts ir pamats darbam nākamajā solī.



2.1. att. **Pētījuma problēmas procesa soļi**

Lai pētījuma kopējo darbu būtu vieglāk apzināt, tas tiek sadalīts 5 daļās, kur katra daļa palīdz problēmas atrisināšanā (2.1. att.). Pētījuma sākumā tiek definēta galvenā problēma, kuras risinājuma darba gaitā tiek meklēts. Pētījuma problēmai tiek ierosināts risinājums kuram tiek apzinātas vairākas problēmas. Lai atrisinātu šīs problēmas, tiek uzdoti pētījuma jautājumi; atbilde uz katru jautājumu ir daļa no kopējā atrisinājuma. Atbilžu rašanai tiek definēti galvenie darba mērķi. Secīgi sasniedzot mērķus, tiek rasts pētījuma problēmas risinājums.

3. TEORĒTISKAIS APSKATS

Nodaļa sniedz pamatzināšanas par šī darba galvenajām tēmām. Pirmā daļa ir pārskats par procedurālo ģenerēšanu un tās pielietojumiem. Otrā daļa sniedz ieskatu lapas izkārtojuma veidošanas rīkos un pieejās.

3.1. Procedurālā ģenerēšana

Skaitļošanas kontekstā procedurālā ģenerēšana (*procedural generation*) ir satura radīšanas metode, kurā izmanto matemātiskos algoritmus, lai ģenerētu saturu, balstoties uz noteiktiem parametriem. Procedurālās ģenerēšanas mērķis ir automātiski radīt saturu, kas parasti tiktu radīts manuāli. Izmantojot matemātiskos algoritmus, iespējams izveidot gandrīz bezgalīgu skaitu dažādu neliela apjoma satura variāciju. Datorgrafikā to sauc arī par nejaušo ģenerēšanu (*random generation*) un plašāk izmanto tekstūru un 3D modeļu radīšanā. Videospēļu veidotāji šo metodi lieto, lai automatizēti izstrādātu lielu satura apjomu, piemēram, spēles kartes, labirintus, reljefu u.c. Neliels failu izmērs, liels satura apjoms un spēles gaitas neparedzamība ir procedurālās ģenerēšanas priekšrocības, tādēļ tā ir plaši pielietota tieši videospēļu jomā [1].

Termins "procedurāls" apzīmē konkrētas funkcijas aprēķināšanas procesu. Piemēram, fraktāļi ir ģeometriski veidojumi, kurus var ģenerēt procedurāli. Parasti tekstūras un reljefus uzskata par procedurāli ģenerētu saturu. Procedurāli ģenerēt iespējams arī skaņu, plašāks tās pielietojums ir runas un mūzikas sintezēšanā. Arī dažādos elektroniskās mūzikas žanros izmanto procedurālo ģenerēšanu, viens no māksliniekiem, kurš šādu pieeju izmanto mūzikā un ir popularizējis terminu "ģeneratīvā mūzika" (*generative music*), ir Braiens Eno (*Brian Eno)*[1] [1].

Procedurālā ģenerēšana var kļūt par vērtīgu izstrādes daļu. Balstoties uz tās principiem, ir iespējams izveidot veselas sistēmas un video spēles. Videospēle *No Man's Sky*[2] pacēla latiņu augstāk un procedurāli ģenerēja veselu visumu ar 18,446,744,073,709,551,616 unikālām planētām, floru un faunu [26]. Alternatīvi, izmantojot starpprogrammatūru, rīkus un spraudņus, ir iespējams piešķirt dažādību galaproduktam, bet kopumā ļaut procedurālai ģenerēšanai būt nelielai daļai no izstrādes procesa [15].

Plašāku pielietojums satura radīšanā procedurālajai ģenerēšanai ir augsnes, piemēram, koku, akmeņu, kalnu un lapu, īpašību (tekstūra, struktūra u.c.) izstrādē. *SpeedTree*[3] ir plaši

[1] https://en.wikipedia.org/wiki/Brian_Eno
[2] http://www.no-mans-sky.com/
[3] http://www.speedtree.com/

izmantota starpprogrammatūra, kura spēj ģenerēt veģetāciju, izmantojot procedurālo ģenerēšanu kopā ar autora radītiem mākslas darbiem. *SpeedTree* spēj izstrādāt tūkstošiem unikālu augstas izšķirtspējas tekstūru reāllaikā un simulēt augšanas procesus un sezonālas pārmaiņas. Lai nevajadzētu katru vides aspektu un detaļu modelēt pašam, *SpeedTree* un procedurālā ģenerēšana atvieglo reālajā dzīvē sastopamu variāciju izveides iespējas, tai pat laikā nodrošinot augstu satura kvalitāti [15].

3.2. *Constraint Cascading Style Sheets* (CCSS)

Constraint Cascading Style Sheets (CCSS)[4] izmanto ierobežojumus, lai deklaratīvā veidā norādītu uzturamās attiecības un nebūtu nepieciešams pašrocīgi rakstīt šīs attiecību uzturēšanas procedūras. Ierobežojums matemātiskā nozīmē ir apgalvojums, kuru ir nepieciešams uzturēt patiesu, piemēram, a = 1/2b. Ierobežojumu pieeja ir noderīga programmēšanas valodās, saskarnes lietotnēs un interaktīvās grafiskās lietotnēs, lai specificētu loga un lapas izkārtojumu. Definējot dažādus ierobežojumus, iespējams aprakstīt izkārtojumu un kontrolēt dažādas lapas iespējas, piemēram, objektu novietojumu un fonta īpašības. Lielākoties ir vērtīgi norādīt obligātos un vēlamos ierobežojumus. Tie ļauj dizainerim specificēt vēlamās sistēmas īpašības, nevis noteikt veidu, kādā tās tiek uzturētas. Lielākā ierobežojumu priekšrocība ir tā, ka tie ļauj specificēt izkārtojumu daļēji un tos iespējams paredzamā veidā lietot kopā ar citām daļējām specifikācijām [3].

CCSS ir uz ierobežojumiem balstīts papildinājums, kas paredzēts *Cascading Style Sheets* (CSS)[5] īpašību paplašināšanai. Ļaujot izmantot ierobežojumus un stila lapu priekšnoteikumus, CCSS paplašina CSS objektu aprakstāmību un ļauj dizaineriem rakstīt stila lapas, kuras kombinē elastību un prognozējamību ar lietotāja iestatījumiem un tīmekļa pārlūka ierobežojumiem [17].

Viens no ierobežojumu lietošanas sarežģījumiem ir tas, ka tie savā starpā var konfliktēt. Lai to risinātu, tiek ieviesta ierobežojumu hierarhija. Ierobežojumu hierarhija sastāv no kolekcijas, kur katrs ierobežojums ir vai nu obligāts, vai vēlams un katram ir norādīts tā svars. Svars ar nosaukumu *REQUIRED* norāda to, ka konkrētais ierobežojums ir obligāti jāizpilda. Citas svara vērtības norāda ierobežojuma priekšrocības. Var tikt lietots jebkāds skaits dažādu svara vērtību. Ierobežojumam ar augstāku svaru būs prioritāte, salīdzinot ar zemāka svara ierobežojumiem. Sistēmai ir jācenšas iespējami apmierināt šīs prasības. Definējot sistēmu ar dažādiem ierobežojumiem, risinātājam jāatrod risinājums mainīgo vērtībām, kuras precīzi izpilda obligātos ierobežojumus un iespējami apmierina vēlamos ierobežojumus, dodot

[4] https://constraints.cs.washington.edu/web/ccss-uwtr.pdf
[5] https://www.w3.org/Style/CSS/

priekšroku ierobežojumiem ar lielāku svaru. Gala risinājuma izvēle ir atkarīga no salīdzinājuma funkcijas, kura tiek lietota, lai izrēķinātu labāko variantu, ņemot vērā, cik veiksmīgi ir apmierināts konkrētais ierobežojums. Jo vairāk ierobežojumu tiek izpildīts, jo labāks ir variants [3, 17].

3.2.1. Cassowary Constraint Solver (CCS)

Cassowary[6] ir inkrementāls ierobežojumu rēķināšanas rīks, kas efektīvi atrisina lineāras vienādību un nevienādību sistēmas. Ierobežojumi var būt prasības vai priekšrocības. Ierobežojumu kolekcija var saturēt ciklus (vairākas vienlaicīgas vienādības un nevienādības vai liekus ierobežojumus) un konfliktējošas prioritātes. Klienta kods specificē uzturamos ierobežojumus, un risinātājs atjaunina ierobežotos mainīgos, ievietojot vērtības, kuras izpilda definēto ierobežojumu. Tiek aprakstītas vēlamās attiecības starp mainīgajiem, atzīmējot svarīgākās saistības, un *Cassowary* atrod optimālo risinājumu, balstoties uz ievadītajiem datiem. Kad vērtības un ierobežojumi tiek mainīti, *Cassowary* ir īpaši efektīvs jaunu atbilžu aprēķināšanā, balstoties un pēdējo zināmo risinājumu. Tāpēc šis rīks ir ideāli piemērots izkārtojumu sistēmām [2, 19, 52].

Ierobežojumu risinātāji ir iteratīvi algoritmi, kuri tiecas pēc ideāla atrisinājuma, bieži lietojot *simplex* algoritma[7] variantus. Šādi risinātāji ir efektīvi gadījumos, kad izveidots tāds noteikumu komplekts, kur vēlams katram noteikumam rast atrisinājumu, bet visus potenciālos atrisinājumus ļoti grūti ņemt vērā [19].

Cassowary ir inkrementāla, labi zināma un plaši pētīta *simplex* algoritma versija, ko lieto, lai risinātu lineāru vienādību un nevienādību ierobežojumu kolekcijas. Plaši pieejamās standartveida *simplex* algoritma implementācijas nav piemērotas interaktīvām lietotnēm, piemēram, tīmekļa pārlūkam. *Cassowary* izvēlas salīdzinātāju ar labāko potenciālo risinājumu, kurš apmierina visus ierobežojumus. Salīdzinātājs izskaitļo risinājuma kļūdas, summējot spēka vērtības un kļūdas gadījumos, kad ierobežojums netiek izpildīts, tādējādi atrodot risinājumu ar vismazāk kļūdām [3].

Cassowary un citi hierarhisko ierobežojumu rīki piedāvā unikālu mehānismu, kas nosaka labāko iespējamo risinājumu gadījumā, kad noteikumu komplekta elementi savstarpēji konfliktē. Ļaujot autoram noteikt ierobežojumu svarīgumu, rīks rod risinājumu, nosakot, kurš ierobežojums būs pārāks par citiem. Šāda veida situācijas UI sistēmu izveidē rodas nepārtraukti. Piemēram: es vēlētos, lai šis objekts būtu savā dabīgajā platumā, bet tikai tad, ja tas ir mazāks par 600px un ne mazāks par 200px [19].

[6] https://constraints.cs.washington.edu/solvers/cassowary-tochi.pdf
[7] https://en.wikipedia.org/wiki/Simplex_algorithm

Cassowary ir ticis pielāgots lietotāju saskarņu lietojumprogrammām. *Cassowary* kopā ar CSS tiek izmantots CCSS implementācijā. CCSS nodrošina izkārtojuma ierobežojumu atbalstu un ļauj tīmekļa izstrādātājiem elastīgu tīmekļa vietnes izkārtojuma aprakstu. CCSS implementācijā *Cassowary* lieto, lai atrisinātu ierobežojumus un aprēķinātu lapas gala izkārtojumu [2].

3.2.2. Grid Style Sheets (GSS)

Grid Style Sheets (GSS)[8] ir CSS priekšprocesors un *JavaScript* (JS)[9] bibliotēka, kura izmanto JS portētu ierobežojumu risināšanas algoritmu *Cassowary.js*[10], kuru *Apple*[11] izmanto *iOS*[12], *macOS*[13] un *Auto Layout*[14] risinājumos. GSS pamats ir modernizēta CCSS implementācija. GSS un *Cassowary* ir bāzēti uz ierobežojumu programmēšanu (*Constraint Programming*) - paradigmu, kur izstrādātāji uzdod jautājumu "kas" un paļaujas uz matemātiskiem risinājumiem, lai noskaidrotu "kā". Tradicionālās objektorientētās programmēšanas valodās izstrādātāji koncentrējas tieši uz jautājumu "kā". Šādi ierobežojumu programmēšana tiek piemērota deklaratīvām valodām, piemēram, CSS [7].

GSS atjaunina CSS izkārtojumu un aizvieto tīmekļa pārlūkprogrammas izkārtojuma dzinēju (*layout engine*) ar tādu, kurš balstās uz CCSS un izmanto CCS algoritmu. GSS papildina CSS ar relatīvu pozicionēšanu un izmēru noteikšanu, kā arī ar avotu secības neatkarību, tādējādi izstrādātājs var centrēt jebkuru elementu citā, izmantojot tikai vienu koda rindu [7].

Ierobežojumi nosaka noturīgas vai nenoturīgas attiecības starp dažādiem mainīgajiem. Ne tikai elementa pozīcija un izmēri var tikt ierobežoti, bet arī tā skaitliskā vērtība. Piemērā visi HTML paragrāfa tagi ir ierobežoti starp līnijas izmēru, kas ir lielāks par 16px un mazāks par 1/12 no loga izmēra:

```
p[line-height] >= 16;
p[line-height] <= ::window[height] / 12;
```

Sintakse ierobežojuma deklarācijai ir šāda:

```
p[line-height] >= 16;
```

"p" ir HTML taga selektors. Ierobežojums atbildīs katram "p" tagam konkrētajā lapā. "[ ]" ir vērtības piekļūšanas sintakse. "line-height" ir vērtība, kuru GSS aprēķinās. ">=" definē

[8] https://gridstylesheets.org/
[9] https://www.javascript.com/
[10] https://github.com/slightlyoff/cassowary.js
[11] http://www.apple.com/
[12] https://en.wikipedia.org/wiki/IOS
[13] https://en.wikipedia.org/wiki/MacOS
[14] https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/AutolayoutPG/

nevienlīdzības ierobežojumu. “16” ir skaitliskā vērtība ierobežojumam pikseļos (px) (pamata GSS mērvienība) [16].

Risinātājam ierobežojumi tiek pievienoti pakāpeniski, pēc tam tas aprēķina izpildāmu, visiem ierobežojumiem piemērotu risinājumu. Ierobežojumu programmēšana pievēršas mērķu izvirzīšanai, nevis to īstenošanai, kā tas būtu imperatīvās programmēšanas[15] pieejā, kur galvenā ir implementācija, taču problēmas jārisina pašam izstrādātājam. GSS atrod risinājumu, kurš vislabāk apmierina definētos ierobežojumus. GSS izmanto “mīkstos” ierobežojumus, kas nozīmē, ka konkrēts ierobežojums ir vēlams, bet ne obligāts. Rezultātu ir iespējams ietekmēt, nosakot ierobežojumu spēka prioritātes. Spēcīgāki ierobežojumi ir pārāki par vājākiem, kā rezultātā veidojas ierobežojumu hierarhija. Ierobežojumu risināšanas algoritms *Cassowary*, kuru izmanto GSS, spēj aprēķināt tikai lineārus aritmētiskus ierobežojumus. Iespējams izmantot vienkāršus matemātiskos operatorus, bet izteiksmēm jābūt lineārām, piemēram y = a + b. GSS spēj veikt visas darbības, izņemot dalīšanu un reizināšanu ar divām ierobežotām vērtībām [16].

[15] https://en.wikipedia.org/wiki/Imperative_programming

4. ESOŠO RISINĀJUMU IZPĒTE

Nodaļā ir apkopota informācija un pētījumu rezultāti par tīmekļa vietnes izstrādes iespējām, izmantojot AI, teksta analīzi, čatbotus un citas pieejas. Sniegts ieskats risinājumos, kuri automatizē tīmekļa vietņu izstrādi un ģenerē vietnes kodu ar dažādiem algoritmiem un pieejām, lai maksimāli atvieglotu klienta vēlmju izpildi.

4.1. The Grid

The Grid[16] ir viena no pasaulē pirmajām mākslīgā intelekta (AI) dizaina platformām, kura autonomi rada tīmekļa vietņu dizainu. *The Grid* izmanto automātiskā dizaina sistēmu, kurā ir iekodēti iepriekš noteikti ierobežojumi. Platforma analizē saturu, krāsu paleti un augšupielādētos attēlus, pēc veiktās analīzes izvēlas dizainu un izveido tīmekļa vietni, kas virzīta uz cilvēk orientētām vērtībām un ierobežojumiem [5]. *The Grid* izmanto režģa izkārtojumu: sistēma izkārto saturu pēc hierarhijas, kura tiek noteikta, iepriekš izvēloties specifisku noteikumu kopu, ko *The Grid* sauc par izkārtojuma filtru (*Layout Filters*) [4].

Sistēmai ir vairākas specifiskas funkcijas [4]:

- **Saturam pielāgots dizains**: vietnē ievietotais saturs nosaka katras lapas izskatu un izkārtojumu.
- **Inteliģenta krāsu noteikšana un korekcija:** sistēma automātiski analizē attēla krāsas un pielāgo attēla fona, saistītā teksta un tā ietvara krāsas. Process notiek automātiski vai manuāli norādot krāsu paleti.
- **Sejas atpazīšana un automātiskā attēlu kadrēšana**: visi attēli tiek automātiski iekadrēti, apgriezti un pielāgoti lapas izmēram un tās redzamajam laukumam.
- **E-komercijas iespēja**: sistēma nosaka, vai nepieciešams ieviest e-komercijas pakalpojumu. Ja rakstā redzama produkta cena, pirkšanas poga vai cita darbība ar preci, tad sistēma automātiski pievieno e-komercijas iespējas.
- **Responsīvs dizains**: *The Grid* veidotās vietnes vienmēr būs responsīvas jeb adaptīvas, saturs un izkārtojums tiks pielāgots redzamajam laukumam. Papildus tiek nodrošināta satura rediģēšana no mobilās lietotnes.
- **Automatizēti A/B testi:** *The Grid* automātiski testē variācijas izveidotajā vietnē un mēra saņemto reakciju. Procesa gaitā tiek rasts veiksmīgākais dizains.

Tīmekļa vietnes izstrādes process, izmantojot *The Grid* platformu, sākas ar lietotāja vietnes veida izvēli. Lietojot izkārtojuma filtrus (kuri ir kā adaptīvas pašģenerējošas veidnes,

[16] https://thegrid.io/

kas reaģē uz ievadīto saturu), *The Grid* izveido meklēšanas dzinēju optimizētu (SEO) vietni, kura, vadoties pēc augšupielādētā satura, ir spējīga pašpielāgoties. Sākotnējā *The Grid* versija satur ierobežotu izkārtojuma filtru skaitu, kas vēlāk tiek papildināti. Pēc filtra izvēles lietotājs norāda vietnes veidu. Tiek piedāvātas vairākas izvēles iespējas [4]:

- Kopfinansējuma vietne;
- *Clickbait* vietne;
- Vietne, kura paplašina sasniedzamību;
- Vietne fanu/sekotāju iegūšanai;
- E-komercijas vietne.

Kad ir noteikts vietnes mērķis, tiek augšupielādēts saturs un, izmantojot algoritmu, *The Grid* izstrādā unikālu, estētisku tīmekļa vietni [4].

4.1.1. The Grid API

The Grid Application Programming Interface (API)[17] ir balstīts uz JSON[18] pieprasījumiem un atbildēm, izmantojot *HTTP REST*[19] un *OAuth2*[20] autorizāciju. *The Grid* API iespējams lietot jebkurā programmēšanas valodā, izmantojot bibliotēkas, kuras atbalsta šīs tehnoloģijas. Populāras izvēles ir *Python*[21], *Ruby*[22], *JavaScript, Java*[23] un *C#*[24] [23].

Visi API piekļuves punkti izmanto CORS[25] atbalstu, kas nozīmē, ka API var izmantot jebkurā tīmekļa pārlūkā un vietnē, arī vietnēs, kas radītas ar *The Grid* platformu.

The Grid API nodrošina šādas funkcijas [23]:

- Tīmekļa vietņu automātisks dizains un izkārtojums;
- Satura analīze un importēšana;
- Attēlu apstrāde;
- Tīmekļa vietņu apkalpošana;
- Maksājumu apstrāde.

The Grid ir viena no pirmajām šāda veida sistēmām, kas to padara par spēcīgu rīku ar plašu funkcionalitāti. Sistēmas izstrādes laikā tika radīti vairāki rīki un algoritmi, kuri atbalsta šādas un līdzīgas sistēmas izstrādi, kas noteikti ir sistēmas papildus pienesums.

[17] https://developer.thegrid.io/
[18] http://www.json.org/
[19] https://en.wikipedia.org/wiki/Representational_state_transfer
[20] https://oauth.net/2/
[21] https://www.python.org/
[22] https://www.ruby-lang.org/en/
[23] https://en.wikipedia.org/wiki/Java_(programming_language)
[24] https://en.wikipedia.org/wiki/C_Sharp_(programming_language)
[25] https://en.wikipedia.org/wiki/Cross-origin_resource_sharing

4.1.2. ColorVerse

ColorVerse ir *The Grid* algoritms, kurš no attēliem izgūst krāsas un izveido tūkstošiem dažādu iespējamo krāsu palešu. *The Grid* izmanto *ColorVerse*, lai izvēlētos piemērotāko krāsu paleti konkrētajai vietnei, vadoties pēc augšupielādētajiem attēliem. *ColorVerse* apstrādā arī pašus attēlus, piemērojot atbilstošus filtrus, lai attēls pēc iespējas labāk iederētos tīmekļa vietnē. *ColorVerse* piedāvā lietotājam plašas iespējas pielāgot krāsas tipu, košumu, blāvumu un citus parametrus [5].

4.1.3. TypographyVerse

TypographyVerse ir algoritms, kas automatizē *The Grid* tīmekļa vietnes fontu izvēli. Tā vietā, lai manuāli izvēlētos galvenes, virsrakstu, teksta un citu elementu fontu, var izvēlēties virzienu, kādā meklēt, pārējo paveic *TypographyVerse*. Pievienojot vairākus desmitus fontu, *TypographyVerse* izveido 100000 [5] potenciālo variāciju, nodrošinot katras vietnes unikālu izskatu.

4.1.4. Problēmas ar The Grid

The Grid spēj radīt tīmekļa vietni un tās kodu, jo izmanto algoritmus, tas pielāgojas un pastāvīgi pieņem lēmumus. Taču radīt labu kodu, kurš atbilst standartiem, ir skaidrs un viegli saprotams, nav tik vienkārši. Katras vietnes kods sniedz unikālu informāciju par vietni. *The Grid* gadījumā daudzām dažādām vietnēm kods būs līdzīgs un nekvalitatīvs, jo tas ir ģenerēts un nav speciāli pielāgots vienas lapas vajadzībām.

4.1.4.1. W3C validācija

Liela daļa *The Grid* ģenerētā koda nespēj iziet W3C validāciju[26], mēdz tikt novērotas dažādas problēmas: elementi ir nepareizi lietoti, atribūti neatrodas savās vietās. Tīmekļa pārlūkam jāspēj vietne attēlot korekti un apstrādāt kļūdas, bet tas nemaina faktu, ka kļūdas ir atrodamas.

Skenējot *The Grid* ģenerētās stila lapas ar CSS validatoru[27], redzams, ka teju 90 kļūdas ir radušās no galvenā iekļautā CSS koda vien. *The Grid* stila lapu koda struktūra neatbilst standartiem un neiziet validācijas procesus [6].

4.1.4.2. Vietnes ielādes ātrums

Lai atsvērtu vājo vietnes kodu, tiek izmantoti *JavaScript polyfills*, kas nodrošina elastīgāku vietnes izstrādi, bet šāda pieeja padara vietni grūti apstrādājamu. Ģenerētajā vietnē

[26] https://validator.w3.org/
[27] http://www.css-validator.org/

ir daudz dažādu lieku objektu un tā nav optimizēta [6]. Šī problēma skars vietnes ar lielu satura apjomu un 5 sekunžu lapas ielādes ātruma standartu, kam priekšroku dod *Google*[28] [14, 53].

4.1.4.3. Semantisks kods

The Grid problēma ir nevis nespēja radīt atbilstošu kodu, bet gan nespēja radīt semantisku kodu. Semantisks kods tiek augstu vērtēts tādēļ, ka datu strukturēšanā tiek izcelta to nozīme. Semantisks kods izveido hierarhiju, kas ir daudz vairāk par vienkāršu faktu kopu. *The Grid* nerada semantisku kodu, jo tā sistēma neatpazīst satura nozīmi.

The Grid izmanto GSS (*Grid Style Sheets*), kas balstās uz *JavaScript polyfills*, lai samazinātu šķirtni starp to, ko CSS spēj konsekventi paveikt, un to, kā sistēmas vēlas vietni pārveidot. GSS piedāvā pieņemamu pieeju, ja nav būtiska *JavaScript* izkārtojuma pielāgošanas neefektivitāte. Tomēr šajā pieejā ir konceptuāla problēma: ir ļoti grūti, pat neiespējami, izstrādāt sistēmu ar kompromisiem, saglabājot satura integritāti [6]. Risinājums būtu izstrādāt sistēmu, kas ne tikai saglabā saturu, bet to atpazīst un izprot.

Izstrādājot sistēmu, kuras prioritāte ir dizaina elastīgums, neņemot vērā attēloto saturu, rodas problēmas. *The Grid* vietņu veidošanā izmanto standartizētu pieeju, un tajā nav radošuma, tikai kods, kurš rada lietojamu, bet ne izcilu, tīmekļa vietni. Kaut arī *The Grid* sola atvieglot izstrādes procesu, koncentrējoties uz saturu, tas nenozīmē, ka vienmēr tiek pieņemti loģiski dizaina lēmumi [4].

4.1.4.4. Atgriezeniskās saites neesamība

Algoritms ne vienmēr izdara visu pareizi, un rezultāts bieži nav tāds, kādu sagaida lietotājs. *The Grid* algoritms darbojas neatkarīgi un kļūdas gadījumā netiek piedāvāta iespēja to labot vai izteikt viedokli par piedāvātā risinājuma efektivitāti. *The Grid* lielākoties nodrošina izstrādātā dizainu labu izskatu, bet brīžos, kad lietotājs vēlas, lai tīmekļa vietne izskatītos labāk par pārējām, ar algoritmu vien nepietiek [8].

4.2. *WIX*

Platforma *Wix*[29] ir radījusi mākslīgā dizaina inteliģenci ADI (*Artificial Design Intelligence*), kas saīsina vietnes izstrādes, dizaina un satura radīšanas laiku un samazina vietnes izstrādes procesā sastopamo šķēršļu skaitu. ADI nodrošina inteliģentas dizaina izvēles, kuru pamatā ir 90 miljoni *Wix* platformas lietotāju [9]. ADI apvieno mākslīgo intelektu un cilvēka radītu dizainu un izstrādā algoritmu, kurš nodrošina efektīvu tīmekļa pieredzi katras vietnes dizainā, balstoties uz vienkāršiem lietotāja kritērijiem. ADI izstrādē tika izmantoti 10 gadu laikā

[28] https://www.google.com
[29] https://www.wix.com/

iegūti dati no 88 miljoniem *Wix* vietņu, kā rezultātā tika izstrādātas "tīmekļa dizainera smadzenes" [13]. Līdzīgi kā cilvēks, iegūstot vairāk datus, šīs "smadzenes" var pieņemt arvien gudrākus lēmumus un turpināt attīstību, pilnveidojot dizainera spējas.

Wix ADI noskaidro lietotāja vēlmes, uzdodot dažādus jautājumus, pēc kuriem vēlāk vadās, izstrādājot unikālu vietnes dizainu. ADI izvēlas piemērotas kombinācijas katram lietotājam, balstoties uz neskaitāmu kombināciju un iespēju analīzi. Uzdodot lietotājam dažus vienkāršus jautājumus, ADI iegūst informāciju par personas vai biznesa vajadzībām un izveido pielāgotu vietni. *Wix* ADI sāk vietnes izstrādi ar pieciem jautājumiem [12]:

- Kādam nolūkam tīmekļa vietne tiks izmantota?
- Vai nepieciešamas speciālas funkcijas un iespējas (interneta veikals, blogs, rezervācijas pakalpojumi u.c.)?
- Kāds ir lietotāja biznesa nosaukums?
- Kur atrodas lietotāja bizness?
- Kāds dizaina stils lietotajam patīk?

Pēc atbilžu saņemšanas ADI izveido t.s. "struktūras", kuras sastāv no teksta, attēliem, vietņu un lapu virsrakstiem – pabeigtas tīmekļa vietnes sastāvdaļām. Nākamais solis ir filtrēšanas process, kurā tiek dzēstas lietotāja vietnē nevajadzīgas lietas. Pēc tam tiek izvēlētas krāsu paletes un vietnes dizains. Visbeidzot tiek apkopotas sekcijas un lapas, lai pabeigtu vietni. Pēc izvēles var veikt papildus pielāgojumus savām vajadzībām [12].

Izvēloties no miljoniem augstas kvalitātes kombināciju un iespēju, ADI apvieno optimālu dizainu un satura elementus, lai radītu unikālu tīmekļa vietni. ADI ne tikai izveido vietnes dizainu, bet arī ievāc nepieciešamo informāciju no tīmekļa un sociālajiem medijiem, lai ievietotu informāciju pareizajās vietas un piedāvātu atbilstošāko saturu, kuru lietotājs var izmantot un pielāgot [11]. Ja lietotājs nav apmierināts, ADI strādā ar lietotāja sniegtajām atsauksmēm, līdz tiek izstrādāta vietne, kura atbilst prasībām. ADI spēj izstrādāt vietni autonomi vai parādīt, kā to darīt. Kad vietne ir pabeigta, tad lietotājam ir iespēja to papildus pielāgot [10].

ADI ir vairākas specifiskas funkcijas:

- **Katra *Wix* ADI radītā tīmekļa vietne ir unikāla:** ADI algoritms no miljoniem kombināciju rada ekskluzīvu, unikālu vietni. Tā kā tam piemīt mākslīgais intelekts, ADI laika gaitā mācās un pielāgo vietni, ņemot vērā lietotāja vajadzības un biznesa kategoriju.
- **ADI ievāc saturu no tīmekļa:** ADI nevis paļaujas uz lietotāja ievākto saturu, bet iegūst to no dažādiem interneta avotiem un sociālajiem tīkliem. Lietotājs var izvēlēties papildināt saturu vai saglabāt to.

- **Lietotājs var pielāgot savu vietni**: ADI izveido lapu, bet lietotājs to var mainīt un pielāgot, pievienojot attēlus, tekstus, mainot tēmas u.c. *Wix* piedāvā bibliotēku, no kuras izvēlēties attēlus.
- **ADI uzsver estētisko pievilcību**: *Wix* vēlas, lai katra vietne izskatītos pēc iespējas labāk, balstoties uz biznesa vajadzībām.

Papildus ADI platformai *Wix* piedāvā mākoņservisā bāzētu platformu, kas ļauj izstrādāt HTML5 tīmekļa un mobilās vietnes. Platforma izmanto "velc un nomet" (*drag and drop*) rīkus. Lietotājs var izmantot tādas funkcijas kā e-komercijas rīki, kontaktu formas, e-pasta mārketings, forumi un citas aplikācijas no *Wix* un trešajām pusēm [10].

4.2.1. Problēmas ar Wix ADI

Wix ADI ir spēcīgs tīmekļa vietnes izstrādes rīks, jo spēj izstrādāt vietnes dizainu un mācīties no miljoniem citu vietņu. Bet ADI nebūt nav pilnīgs, un tā izstrādātie risinājumi ir grūti pielāgojami. Ja ADI uzreiz izstrādā vietni, kura atbilst visām prasībām, tad tā ir liela veiksme. Gala produktu ir grūti mainīt konkrētām vajadzībām, un gadījumos, kad nepieciešams kas pilnīgi cits, izstrādātais risinājums nepiedāvā tik lielas pielāgošanas iespējas, kādas varētu vēlēties.

4.2.1.1. Vietnes saites (URL)

Wix ADI nepiedāvā iespēju izveidot cilvēklasāmus URL. URL struktūra ir ne tikai grūti lasāma, bet arī ietekmē SEO un *Google* lapas vērtējumu (*page rank*). *Google* zemu vērtē tās vietnes, kurām URL nav cilvēklasāms un kuras neiekļauj konkrētus atslēgas vārdus, pēc kuriem var meklēt vietni [14].

4.2.1.2. Nerediģējams avota kods

Wix ADI nav rediģējama avotu koda. Tas ne vienmēr ir vajadzīgs, bet gadījumos, kad ģenerētais kods ir nepilnīgs un nepieciešams veikt papildus modifikāciju, ADI tas nav izdarāms [21].

4.2.1.3. Grūti maināma lapas veidne

Wix piedāvā plašu lapas veidņu izvēli, taču, tiklīdz vietne ir izveidota un tās tēma izvēlēta, nav iespējams mainīt tās dizainu vai izvēlēties citu tēmu. Lai izvēlētos citu vietnes dizainu, viss process ir jāsāk no jauna, katra lapa jāizveido vēlreiz un viss saturs manuāli jāievada atkārtoti [22].

4.3. *RightClick.io*

RightClick.io[30] ir automatizēts tīmekļu vietnes izstrādes rīks, kuru darbina AI. *RightClick.io* ir izstrādājis čatbotu ar nosaukumu *RC Bot*, kas palīdz tīmekļa vietnes izstrādē. *RC Bot* izmanto mašīnmācīšanos un dabiskās valodas apstrādes algoritmus, lai izprastu lietotāja rakstīto un paveiktu nepieciešamos uzdevumus. *RC Bot* efektīvi apvieno sarunas intelektu un sabiedrisko etiķeti, kas nozīmē tīmekļa vietņu dizainera augstu precizitāti. *RightClick.io* ir unikāls, jo nodrošina tīmekļa vietnes izstrādi, vadoties pēc sarunas ar *RC Bot*. *RightClick.io* netiek piedāvāts ierastā tīmekļa vietnes vadības panelis, kas samazinātu vietnes izstrādes rīka sarežģītību, tā vietā ir iespēja vietni rediģēt, izmantojot peles labo taustiņu [24, 25].

RightClick.io inteliģentā sistēma spēj apstrādāt lielu apjomu sarežģīta vietnes koda, pateicoties cilvēka-mašīnas aizmugursistēmai. *RightClick.io* programmatūra saglabā milzīgus datu apmērus, ieraksta un atceras katra klienta prioritātes, pieņem gudrus lēmumus – cilvēkam grūti paveicamas darbības, un spēj tās atkārtot, nepieļaujot kļūdas. *RightClick.io* inovatīvā komanda ir piešķīrusi inteliģentajam izstrādes rīkam konteksta izpratni un empātiju, kas ir būtiski faktori cilvēciskai saskarsmei. Tas tiek panākts, izstrādājot programmatūru, kura sadala uzdevumu pašos vienkāršākajos mikroelementos, lai cilvēkam būtu iespēja iejaukties vietās, ar kurām ierīces nespēj tikt galā. Tātad ir radīta sistēma, kura sniedz precīzu, uzticamu un inteliģentu atbildi dažu sekunžu laikā jebkurā diennakts stundā, jebkurā pasaules malā [24].

RightClick.io piedāvā vairākas funkcijas:

- ***RC Bot*:** unikāls čata robots, kurš sarunā noskaidro klienta vēlmes un izveido unikālu, tām pielāgotu tīmekļa vietni. *RC Bot* simulē sarunu, kāda tā būtu ar īstu tīmekļa dizaineri, šādā veidā likvidējot manuālu izvēli.
- **Labais peles klikšķis**: *RightClick.io* ir atbrīvojušies no tīmekļa vietnes vadības paneļa un tā vietā ir ieviesuši peles labā taustiņa papildus funkciju. Visa nepieciešamā informācija nav jāievada un jārediģē atsevišķās vadības paneļa formās. *RightClick.io* satura rediģēšanu nodrošina uzreiz pašā vietnē, veicot labo peles klikšķi uz rediģējamā elementa.
- **Logo ģenerators**: ja lietotājam nav vietnes logo, *RightClick.io* piedāvā vienkāršu logo radīšanas rīku. Ievadot tekstu, mainot tā fontu un citus parametrus, tiek radīts unikāls logo, kuru izmantot vietnē.
- **Adaptīvas krāsu shēmas:** *RightClick.io* vadās pēc attēlu krāsām un pielāgo vietnes krāsu shēmas logo un citu attēlu krāsām.

[30] https://rightclick.io/

Spilgtākā *RightClick.io* iezīme ir tās izveidotais čata robots. Tas ir interesants veids, kādā nodrošināt tīmekļa vietnes izstrādi. Bet izveidotais risinājums ir iespējams arī sistēmas lielākais klupšanas akmens.

4.3.1. Problēmas ar RightClick.io

RightClick.io ir unikāls rīks ar kura palīdzību veikt tīmekļa vietnes izstrādi, bet arī šim rīkam ir novērojamas vairākas nepilnības, kuras šo procesu apgrūtina. Sistēma veic tīmekļa vietnes izveidi pēc iepriekš noteikta scenārija, kurš var neatbilst lietotāja vēlmēm un dotā brīža nepieciešamībām. *RightClick.io* čatbotu ir viegli samulsināt ar nestandarta atbildēm un brīžiem izdodas nonākt sarunas strupceļā, piemēram, atbildot uz jautājumu, par to vai lietotājam ir tīmekļa vietnes logo, ar noraidošu atbildi atvērās sistēmas logs, kurā jāpievieno vietnes logo.

4.3.1.1. Sistēma ir lēna

Sarakste ar *RightClick.io* čatbotu reizēm var būt lēna. Sarakstoties ar šo čatbotu ir brīžiem tas kļūst lēns un ilgi "domā". Uzdodot tam smagāku jautājumu vai neprecīzi tam sniedzot atbildes ir novērojama intensīva datu apstrāde, kura aizņem daudz laika. Lietojot sistēmu ir novērojama aizture pat uz vienkāršākajiem jautājumiem un atbildēm: atbildot uz čatbota jautājumu par lietotāja vārdu tas var apdomāties vairāku sekunžu garumā. Kas brīžiem ir kaitinoši.

Arī pats izstrādes process ir lēns, tā uzbūve un pamatprincipi noved pie salīdzinoši lēna tīmekļa vietnes izveides procesa. Veicot tīmekļa vietnes izstrādi ar *RightClick.io* ir jāatbild uz vairākiem jautājumi, ir jāatlasa opcijas no piedāvātajām un nav iespējas izlaist atsevišķus soļus, kuri lietotājam var nešķist būtiski.

4.3.1.2. Saturs tiek pievienots pēc stila izveides

Veicot tīmekļa vietnes izveidi ar *RightClick.io* vispirms tiek izveidota tīmekļa vietnes struktūra un dizains, bet tikai pēc tam tiek domāts par vietnes saturu. Izmantojot šādu pieeju rodas sajūta, ka visiem lietotājiem tiek piedāvāts viens un tas pats tīmekļa vietnes stils, kuru vēlāk iespējams modificēt. Šāda pieeja ir izmantojama, bet var rasties situācijas, kur tīmekļa vietne neatbilst saturam, kuru tai paredzējis lietotājs.

5. TĪMEKĻA VIETNES ĢENERĒŠANAS RISINĀJUMS

Pētījuma praktiskās daļas mērķis ir izveidot risinājumu, kas nodrošina automātisku tīmekļa vietnes izkārtojuma un stila ģenerēšanu. Lai atrisinātu pētījumā izvirzīto problēmu autors piedāvā risinājumu, kurš realizē minēto funkcionalitāti.

Lai samazinātu tīmekļa vietnes izstrādei nepieciešamos resursus, laiku un sarežģītību, autors piedāvā sistēmu, kura veic tīmekļa vietnes ģenerēšanu, balstoties satura analīzi. Šī sistēma automatizē tīmekļa vietnes izstrādi, to sadalot atsevišķos procesos, kuru rezultāts ir unikāla tīmekļa vietnes struktūra un pielāgots vietnes dizains. Tīmekļa vietnes ģenerēšanas risinājuma mērķis ir saīsināt nepieciešamo vietnes izstrādes laiku, samazināt izmaksas un specifisku zināšanu nepieciešamību, nododot vietnes izkārtojuma un stila izstrādi algoritmu pārziņā.

Risinājums ģenerē tīmekļa vietni, par pamatu ņemot tīmekļa vietnes datus. Izmantojot procedurālo ģenerēšanu, tīmekļa vietne tiek veidota no atsevišķiem blokiem, tos kombinējot dažādos veidos un piemērojot tiem dažādus stilus, rezultātā radot unikālu vietnes izkārtojumu un stilu. Sistēma spēj ģenerē izkārtojumu un stilu atkārtoti, tādējādi veidojot daudz dažādu iespējamo vietnes stilus, no kuriem var izvēlēties atbilstošāko.

Lai izstrādātais risinājums būtu funkcionāls, kvalitatīvs un nodrošinātu minēto funkcionalitāti, tiek izvirzītas šādas sistēmas pamatnostādnes:

- Sistēmai jāspēj ģenerēt tīmekļa vietnes izkārtojumu un stilu;
- Sistēmai jāspēj ģenerēt dažādus lapas izkārtojumus;
- Sistēmai jāspēj ģenerēt dažādus lapas stilus;
- Sistēmas ģenerētajai tīmekļa vietnei jābūt lietojamai un pilnvērtīgai;
- Sistēmas ģenerētajam tīmekļa vietnes dizainam jābūt adaptīvam;
- Sistēmas struktūrai jābūt modulārai;
- Sistēmai jāveic satura analīze un vadoties pēc tās rezultātiem jāspēj veikt dizaina lēmumi;
- Sistēmas ģenerētajai tīmekļa vietnei jābūt viegli modificējamai.

Sistēmas daļām jāspēj darboties neatkarīgi un jānodrošina risinājums, kur iespējams papildināt vai mainīt kādu sistēmas daļu, būtiski neietekmējot pārējo daļu darbību. Šāda risinājuma struktūra ļauj kopējā sistēmā integrēt atsevišķus, kopējā risinājuma mērķu sasniegšanai pielāgotus risinājumus no trešajām pusēm. Sistēma sastāv no vairākām komponentēm, kur katra atsevišķa komponente nodrošina vienu vai vairākas būtiskas sistēmas īpašības. Visas sistēmas sastāvdaļas tiek integrētas vienā veselā sistēmā. Tālāk aprakstītas atsevišķas sistēmas komponentes un to nozīme kopējās sistēmas darbībā.

5.1. Vietnes izkārtojuma ģenerēšana

Nodaļas sākumā izvirzītais mērķis sastāv no vairākām atsevišķām daļām, kuru sasniegšanas rezultāts ir sistēmas prototips. Risinājuma pamata funkcija ir tīmekļa vietnes izkārtojuma un stila ģenerēšana. Svarīgs risinājuma aspekts ir spēja ģenerēt unikālus un izskatā dažādus vietnes izkārtojumus vienam datu komplektam. Viena izkārtojuma ģenerēšana konkrētiem datiem ir salīdzinoši vienkāršs process, bet ģenerēt vairākus būtiski atšķirīgus vietnes izkārtojumus ir krietni sarežģītāk.

Lai ģenerētu atšķirīgus vietnes izkārtojumus, tīmekļa vietne tiek uztverta kā atsevišķu bloku kopa, kur katrs bloks tiek dalīts atsevišķos elementos. Izkārtojumu ģenerēšanai vajadzīga sistēma, kas izmanto modulāru struktūru, īstenotu ar moduļu bibliotēku, kurā glabāti atsevišķi bloki un bloku elementi. Gatavā vietne tiek komplektēta no atsevišķiem blokiem, un bloki tiek veidoti no bloku elementiem, rezultātā attīstot unikālu vietnes izkārtojumu.

Vietnes izkārtojuma izveides pamatā ir procedurālās ģenerēšanas pieeja un uz nejaušības principa balstīts bloku un to elementu atlasīšanas process. Lai nodrošinātu saturam atbilstošu vietnes dizainu, tiek izmantota teksta un attēlu atpazīšana, kuras mērķis ir izprast saturu un pēc tā atlasīt atbilstošākos blokus un elementus, ar kuriem ģenerēt vietnes izkārtojumu.

5.2. Vietnes stila ģenerēšana

Atšķirīgam vietnes izskatam ar dažādiem izkārtojumiem vien ir par maz. Lai nodrošinātu patiesi dažādus vietnes izskatus, nepieciešams padomāt par vietnes stilu un dažādu elementu attēlošanu. CSS piedāvā plašu atribūtu klāstu, kurus iespējams mainīt, lai iegūtu vizuāli atšķirīgus vietnes izskatus. Tomēr, veicot izpēti, tiek secināts, ka lielākā nozīme vizuālā tēla izveidē ir šādiem atribūtiem:

- Teksta fonts un tā izmērs;
- Krāsu palete;
- Malas, polsterējums un rāmis.

Teksts ir viena no svarīgākajām tīmekļa vietnes sastāvdaļām. Tas aizņem lielu daļu vietnes laukuma, un, ja vien vietne nav specifiska attēlu galerija u.tml., teksta veidā tiek nodota liela daļa vietnes informācijas un tam tiek pievērsta pastiprināta uzmanība. Virsraksti, saturs, navigācija un citi elementi parasti satur tekstu, līdz ar to fonts un tā izmērs ir faktori, kuri spēcīgi ietekmē lapas izskatu. Nedaudz izmainot fontu un tā izmērus, iespējams divām vienādām vietnēm radīt pilnīgi atšķirīgu stilu.

Vēl viens būtisks stila aspekts ir vietnes krāsas. Katram vietnes elementam ir sava krāsa: teksta krāsa, fona krāsa, pogu, malu un citas krāsas. Ģenerējot vietni un mainot tās krāsas, var

iegūt daudzas kombinācijas, kuras struktūras ziņā var būt vienādas, bet dažādu krāsu salikumu dēļ izskatās atšķirīgi. Mainot vietnes krāsu paleti un piemeklējot saturam atbilstošus krāsu salikumus, iespējams iegūt daudz unikālu stilu.

Katram elementam ir savs atribūtu komplekts: malas (*margin*), polsterējums (*padding*) un rāmis (*border*). Nedaudz izmainot atribūtu izmērus un savstarpējās attiecības, rodas dažādas elementa variācijas, tāpat, mainot bloka atribūtus, pārvēršas bloka izskats.

Lai sekmīgi ģenerētu unikālus vietnes dizainus, tiek mainīts ne vien tās izkārtojums, bet arī minētie elementu atribūti. CSS piedāvā plašu atribūtu klāstu, bet, pastiprināti pievēršot uzmanību šiem elementiem, iespējams iegūt vislielākās pārmaiņas un vizuālās atšķirības dizainā.

5.3. Modulāra uzbūve

Lai īstenotu sistēmu, kura tīmekļa vietni veido no atsevišķiem blokiem, svarīgs tās aspekts ir modularitāte. Šāda tipa sistēmai ir jāsastāv no savstarpēji neatkarīgiem blokiem, kurus pēc vajadzības iespējams noņemt, pievienot vai modificēt. Modulāra sistēmas uzbūve attiecas gan uz veidu, kādā tiek ģenerēta tīmekļa vietne (sastāv no moduļu bibliotēkas), gan arī uz pašu vietnes ģeneratoru, kam jābūt modulāram. Vietnes ģenerēšana ir komplekss process, kurā tiek izmantotas vairākas komponentes. Sistēmai attīstoties vai ģenerējot sarežģītāku vietni, jāpievieno jaunas funkcijas un jāmaina vecās komponentes, lai sasniegtu labāku rezultātu.

Modularitātes princips ir izveidot sistēmu, kuras moduļus iespējams optimizēt neatkarīgi no citiem moduļiem. Ja kāds no moduļiem vairs nedarbojas, tas neietekmē citu moduļu darbību. Moduļi palīdz izstrādāt, uzturēt un saprast kompleksas un neatkarīgas sistēmas.

Modulāras sistēmas pamatprincipi [42]:

1. **Modulāra dekompozīcija:** Modulāra dekompozīcija tiek sasniegta tad, ja programmatūras izstrādes metode palīdz sadalīt lielas problēmas mazākās un vienkāršākās problēmās, kuras savienotas vienkāršā struktūrā un kuras ir gana neatkarīgas, lai tās varētu individuāli izstrādāt.
2. **Modulāra kompozīcija:** Programmatūras izstrādes metode apmierina modulāru kompozīciju, ja no konkrētās programmatūras blokiem iespējams izveidot jaunu un neatkarīgu sistēmu, tos savstarpēji kombinējot, potenciāli citādos apstākļos, nekā tas sākotnēji paredzēts.
3. **Modulāra izpratne:** Modulāra izpratne tiek sasniegta tad, ja sistēmas izstrādātājs spēj saprast katru programmatūras moduli atsevišķi, nezinot neko par citiem moduļiem vai zinot informāciju tikai par dažiem moduļiem.

4. **Modulāra nepārtrauktība:** Modulāra nepārtrauktība nozīmē gadījumu, ja sistēmas arhitektūras izmaiņas tajā ietekmē tikai vienu moduli vai nelielu skaitu moduļu.
5. **Modulāra aizsardzība:** Modulāra aizsardzība tiek nodrošināta, ja sistēmas arhitektūra ir izstrādāta tā, ka kļūdas gadījumā tā tiek ierobežota konkrētā modulī vai atsevišķos blakus moduļos.

Tipiskā sistēmas gadījumā risinājums tiek izveidots, balstoties uz iepriekš zināmām vajadzībām un kritērijiem, un modulāra pieeja šādā gadījumā nav vitāli nepieciešama, tomēr sniegtu priekšrocības. Risinājuma sistēmas pamatideja balstās uz modularitātes principu, kur gala rezultāts tiek sasniegts, savstarpēji kombinējot dažādus moduļus, un modulāra sistēmas arhitektūra nodrošina sistēmas sekmīgu darbību.

5.4. Sistēmas modificēšana

Nodaļā 5.3. tika aprakstīta sistēmas modularitāte; būtiska modulāras sistēmas iezīme ir iespēja sistēmu vienkārši modificēt. Piedāvātā risinājuma sistēmas modifikācijas iespēja sadalīta trīs atsevišķās daļās:

- Ģenerētās tīmekļa vietnes modificēšana;
- Iekšējās sistēmas un tās funkciju modificēšana;
- Tīmekļa vietnes pamatelementu modificēšana.

Ne vienmēr piedāvātais risinājums ir ideāli atbilstošs lietotāja vēlmēm, tādēļ ir ļoti svarīgi nodrošināt iespēju modificēt ģenerēto tīmekļa vietni. Lietotājs var vēlēties pievienot papildus saturu, līdz ar to sistēmā jābūt šādai iespējai. Sistēmai jānodrošina atsevišķu elementu modificēšana: ja lietotājam nepatīk izkārtojums, krāsa vai atsevišķi elementi, tad jāsniedz iespēja mainīt šos elementus, neietekmējot kopējo vietnes kompozīciju un stilu. Tāpat risinājumam jānodrošina iespēja atkārtoti ģenerēt visas vietnes izkārtojumu un dizainu. Risinājuma mērķis ir piedāvāt lietotājam tīmekļa vietni, kura atbilst tā vēlmēm, un iespēja atkārtoti ģenerēt vietnes dizainu ir efektīvs veids, kā lietotājam piedāvāt dažādas iespējas, no kurām izvēlēties sev tīkamāko.

Ģenerējot dažādas vietnes, mainās arī pašas sistēmas funkcijas un iespējas. Sistēma attīstās, ieviešot sarežģītākus dizainus un papildus iespējas. Nepieciešama sistēmas struktūra, kuru ir iespējams viegli modificēt un ieviest jaunas, prasībām atbilstošas funkcijas. Vietnes prototipa pamatā ir neliels pamatfunkciju klāsts, kuras vēlāk nepieciešams papildināt. Lai nodrošinātu iespēju sistēmu attīstīt, paplašināt un mainīt, svarīgs tās faktors ir vienkārša modificēšana.

Piedāvātā risinājuma pamatā ir bibliotēka ar atsevišķiem moduļiem, no kuriem tiek ģenerēta tīmekļa vietne. Atsevišķie moduļi ir jāuztur, bieži veicot izmaiņas, lai tiem pievienotu

jaunas iespējas un elementus. Lai panāktu pēc iespējas plašāku unikālu dizainu un izkārtojumu piedāvājumu, sistēmai nepieciešams pievienot arvien jaunus moduļus. Ja sistēma ir statiska un atsevišķi tās elementi ir savstarpēji integrēti, tad šāds uzdevums ir teju neiespējams.

5.5. Satura analīze

Jebkuras tīmekļa vietnes pamatā ir tās saturs, kas diktē noteikumus, un pati tīmekļa vietne ir veids, kā pēc iespējas efektīvāk atspoguļot saturu. Ģenerējot tīmekļa vietnes stilu, saturs kļūst otršķirīgs, jo vispirms tiek domāts par vietnes izkārtojumu un stilu, bet tikai pēc tam tiem tiek pievienots konkrēts saturs. Šāda pieeja, kaut iespējama, noteikti nav labākais risinājums. Standartizēts dizains var būt piemērots vairumā gadījumu, bet daudzkārt šāda pieeja nespēj apmierināt prasības.

Lai tīmekļa vietnes dizains ne tikai izskatītos labi, bet atbilstu saturam, ir jānodrošina izvērtēšanas mehānisms, kas nosaka, kāds dizains saturam der un kāds ne. Lai noskaidrotu, kāds dizains saturam piemērots, ir jāizprot pats saturs. Tāpēc jāveic satura analīze. Tādējādi iespējams noskaidrot dažādus parametrus, uz kuriem tiek balstīta konkrēta satura dizaina izvēle. Saturu var analizēt divējādi: pēc būtības un pēc parametriem.

Analizējot saturu pēc tā parametriem, var noskaidrot, vai tas fiziski iederēsies ģenerētajā dizainā. Par piemēru var ņemt tekstu, kurš tiktu ievietots tīmekļa vietnē. Tekstam var noteikt tā apjomu, vārdu skaitu, potenciālo garumu, rindkopas un citus parametrus, kuri var ietekmēt to, kā šis saturs izskatīsies tīmekļa vietnē. Attēlam var noteikt tā izmērus, malu attiecību, krāsas u.c. parametrus. Piemēram, ja tiek konstatēts, ka saturā ir daudz mazu attēlu, tad var pieņemt, ka dizains, kura pamatā ir lieli attēli, nebūs efektīvs risinājums.

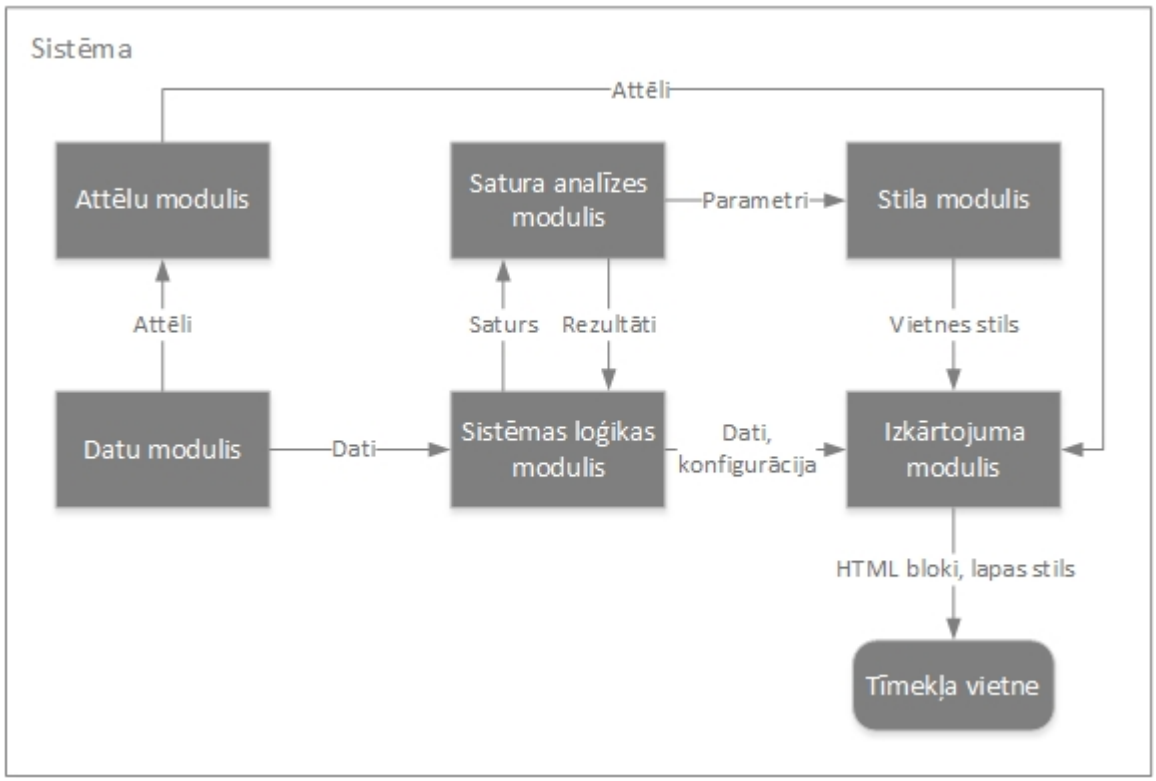
Satura analīze pēc būtības ir sarežģītāks process, bet arī šīs analīzes rezultāti var būtiski ietekmēt ģenerētā dizaina kvalitāti. Veicot satura analīzi pēc būtības, tiek noskaidrots saturā pateiktais. Ja tiek analizēts teksts, tad no tā var izgūt galvenos atslēgas vārdus, datumus, cilvēku vārdus, tā noskaņu un citus elementus. Veicot attēla analīzi, var noteikt tā krāsas, tajā attēloto, (ainava, portrets, dzīvnieki u.c.). Vadoties pēc šis informācijas, var pieņemt loģiskus dizaina lēmumus, nodrošinot saturam atbilstošu vietnes stilu un paaugstinot kopējo tīmekļa vietnes kvalitāti.

6. SISTĒMAS KOMPONENTES

Piedāvātā sistēma sastāv no vairākām komponentēm, kuru savstarpēja mijiedarbība nodrošina sistēmas pamata funkcijas. Sistēma sastāv no 6 atsevišķiem moduļiem, kuri atbild par sistēmas funkcijām:

- Izkārtojuma modulis: modulis atbild par vietnes elementu izkārtojumu, tā pamatā ir HTML failu bibliotēka, režģa izkārtojums, GSS un PHP veidņu dzinējs.
- Stila modulis: bloks atbild par vietnes stila izveidi, tā pamatā ir CSS,, atsevišķu bloku stils un CSS pirmsprocesors.
- Attēlu modulis: bloks atbild par attēlu apstrādi.
- Datu modulis: atbild par datu struktūru, un datu padošanu atsevišķām sistēmas komponentēm.
- Sistēmas loģikas modulis: sistēmas pamats, kurš apvieno visus blokus un nodrošina sistēmas darbību.
- Satura analīzes modulis: modulis veic satura analīzi, uz kuriem tiek balstīta dizaina lēmumu pieņemšana.

Sistēma tiek iedalīta atsevišķos moduļos, lai to būtu vieglāk izprast un aprakstīt. Moduļi savā starpā var būt cieši integrēti un atkarīgi viens no otra.



6.1. att. **Moduļu projektējums**

Attēlā 6.1. attēloti sistēmas pamata moduļi un to mijiedarbība. Sistēmas pamatā ir loģikas modulis, kurš apvieno visas sastāvdaļas un nodrošina sistēmas kopējo darbību. Sistēmas loģikas modulis izmanto datu moduli, lai pieņemtu dizaina un struktūras lēmumus. Datu modulis satur sistēmas datus un nodrošina attēlus, kurus attēlu modulis apstrādā un ievieto izkārtojumā. Stila modulis izveido tīmekļa vietnes stilu. Sistēmas loģikas modulis padod datus un konfigurācijas datus izkārtojuma modulim, kurš, izmantojot stila moduli, izveido lapas izkārtojumu un kopējo vietnes stilu. Satura analīzes modulis veic satura analīzi datus padodot loģikas modulim, kurš tālāk veic to apstrādi Visa procesa rezultāts ir tīmekļa vietnes stils.

6.1. Vietnes izkārtojums (*page layout*)

Tīmekļa vietnes pamatā ir informācija, kas var sastāvēt no teksta, attēliem, video un citiem elementiem. Galvenais tīmekļa vietnes uzdevums ir pasniegt informāciju ērtā un viegli uztveramā veidā. Lai padarītu informāciju viegli uztveramu un attēlotu to strukturētā un loģiskā veidā, ir nepieciešams izveidot lapas izkārtojumu. HTML un CSS ir divas pamata tehnoloģijas tīmekļa vietnes izstrādei. HTML kods nodrošina lapas struktūru un apraksta tās informāciju. Par vizuālo lapas noformējumu un izkārtojumu atbild CSS.

6.2. att. **HTML kods un lapas izkārtojums**

HTML ir satura iezīmēšanas valoda, paredzēta lapas informācijas aprakstīšanai. HTML izmanto tagus jeb atzīmes, kurus kombinējot ar saturu tiek veidota lapas struktūra. HTML apraksta saturu mašīnai saprotamā veidā, papildus dalot lapu loģiskos blokos.

Attēlā 6.2. ieskicēts HTML kods un lapas struktūra, kuru šis kods definē. Viena no HTML īpašībām ir modularitāte un iespēja sadalīt lapu blokos, kurus vēlāk iespējams atsevišķi apstrādāt. HTML nodrošina semantiskus elementus, ar kuriem nosaukt dažādas lapas daļas [30]:

- <*header*> - Dokumenta vai sekcijas galvene;
- <*nav*> - Konteineris navigācijas elementiem;
- <*section*> - Dokumenta sekcija;
- <*article*> - Neatkarīgs un pašpietiekams raksts;
- <*aside*> - Dokumenta satura mala;
- <*footer*> - Dokumenta vai sekcijas kājene.

CSS ir valoda, kuru izmanto, lai aprakstītu lapas attēlošanu, tai skaitā krāsas, izkārtojumu, fontus u.c. CSS apraksta HTML elementu attēlojumu uz displeja, papīra vai citos medijos. CSS kontrolē lapas izkārtojumu un spēj aprakstīt vairākas lapas vienlaicīgi, definē lapas stilu un apraksta variācijas dažādām ierīcēm un ekrānu izmēriem [31]. CSS un HTML ir savstarpēji neatkarīgi. Abu atdalīšana ļauj viegli uzturēt dažādas lapas, atkārtoti izmantot stilu un pielāgot lapas dažādām vidēm, efektīvi atdalot saturu un struktūru no to prezentācijas [32].

CSS ir atbildīgs par lapas izkārtojumu, jo tieši ar CSS tiek aprakstīta elementu atrašanās vieta, izmēri, savstarpējās attiecības un darbība dažādos apstākļos.

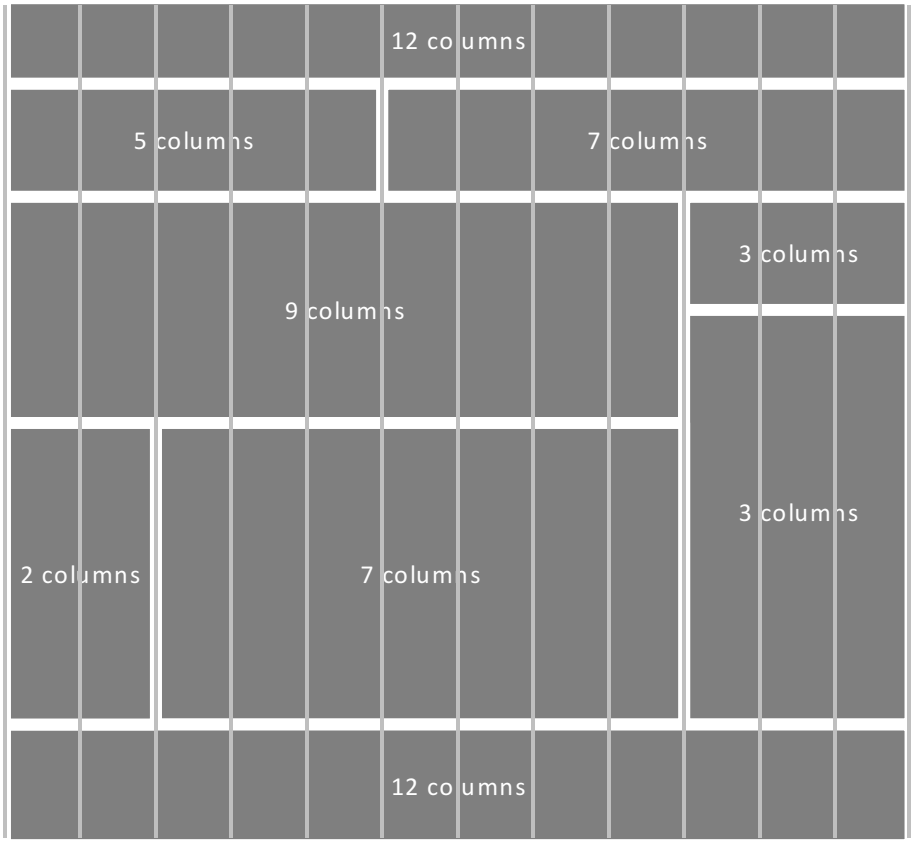
6.1.1. Režģa sistēma (grid system)

Režģa sistēma ir struktūra, kas ļauj izkārtot saturu gan horizontāli, gan vertikāli viegli pārvaldāmā veidā. Papildus ērtai satura izvietošanai režģa sistēmas kods ir pārizmantojams un nodrošina augstu pārnesamību, lai to būtu viegli un ērti integrēt jaunos projektos. Režģa sistēma var tikt implementēta tīmekļa vietnē, lai nodrošinātu HTML izkārtojumu [33].

Režģa sistēmai ir vairākas būtiskas priekšrocības [33]:

- Režģa sistēma **paaugstina produktivitāti**, nodrošinot vienkāršu un paredzamu HTML izkārtojumu. Lapas struktūru var viegli aprakstīt, nedomājot par izkārtojuma precizitāti un starppārlūku savietojamību.
- Režģa sistēma nodrošina **daudzpusīgu veidu**, kādā būvēt lapas izkārtojumu, jo to var izmantot dažādās kolonnu un režģa kombinācijās. Režģa sistēma nodrošina papildus režģa iekļaušanu sarežģītākiem lietojuma gadījumiem. Neatkarīgi no tā, kādas ir izkārtojuma prasības, režģa sistēma tām visdrīzāk atbildīs.
- Režģa sistēma ir **ideāla adaptīvā dizaina izkārtojumam**. Šis ir lauciņš, kurā režģa sistēma ir pārāka pār citām. Režģa sistēma ir ļoti viegls veids, kādā radīt mobilām ierīcēm piemērotus, dažādiem ekrānu izmēriem pielāgojamus izkārtojumus.

Režģa sistēmai ir divas pamata komponentes: rindas un kolonnas. Rindas ir domātas, lai pielāgotu kolonnas. Kolonnas veido gala struktūru un tajās glabājas lapas saturs. Dažas režģa sistēmas iekļauj papildus konteinerus, kuri kalpo kā izkārtojuma atdalītāji no pārējās vides.



6.3. att. **Režģa izkārtojuma struktūra**

Režģa sistēma nodrošina 12 kolonnas, kuru kopējais platums atbilst 100% lapas (6.3. attēls). Kolonnas var izmantot pa vienai vai veidot platāku kolonnu, savienojot vairākas. Režģa sistēma nodrošina adaptīvu dizainu, un uz dažādiem ekrāniem kolonnas izvietojas dažādi. Uz liela ekrāna labāk izskatās trīs kolonnās organizēts saturs, bet uz mazāka ekrāna – vienā kolonnā sakārtots saturs.

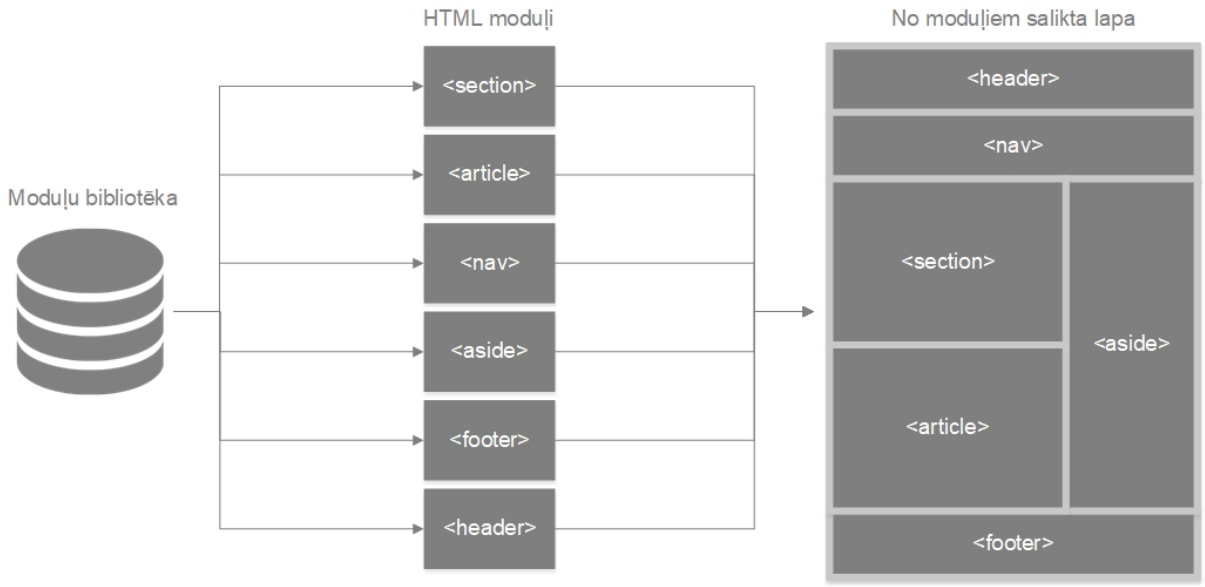
6.1.2. Izkārtojuma modularitāte

Sistēmas modularitāte tiek panākta, grupējot saistītas funkcijas un datu struktūras vienkopus, veidojot atkārtoti izmantojamas vienības jeb moduļus, kuri apkopoti moduļu bibliotēkās. Modulāra sistēmas struktūra nodrošina vairākas priekšrocības [34]:

- **Elementu un funkciju atkārtota izmantojamība**: modulāra struktūra sniedz iespēju vienu elementu izmantot atkārtoti gan viena, gan vairāku projektu ietvaros. Vienreiz izstrādājot navigācijas elementu, to var ērti izmantot citur, taupot resursus un laiku.

- **Viegla uzturēšana**: izmaiņas ir neizbēgamas jebkura projekta gaitā. Modulāra projekta struktūra ir unikāla ar to, ka izmaiņu ieviešana un labojumu veikšana ir salīdzinoši vienkārša. Izmaiņas tiek veiktas vienuviet, kas palīdz izvairīties no kļūdām. Moduļa izmaiņas neietekmē citus moduļus, tāpēc process minimizē resursus un kļūdas iespējamību.
- **Resursu taupīšana –** viens elements vai funkcija modulārā struktūrā fiziski atrodas tikai vienā vietā, bet var tikt izmantots atkārtoti, rezultātā tiek uzturēta salīdzinoši neliela bibliotēka, kura aizņem mazāk vietas un kuru ir vieglāk uzturēt.

Šādas moduļu bibliotēkas ir ļoti svarīgas sistēmā, kura dinamiski attīstās un salikta no vairākām daļām. Koncepta gadījumā ir svarīgi, lai sistēmas komponentes būtu neatkarīgas, viegli izmantojamas un viena sastāvdaļa varētu tikt izmantota vairākkārt. HTML struktūrā ir viegli izdalīt atsevišķas komponentes, veidojot pēc vajadzības izmantojamu un savstarpēji kombinējamu HTML elementu bibliotēku.

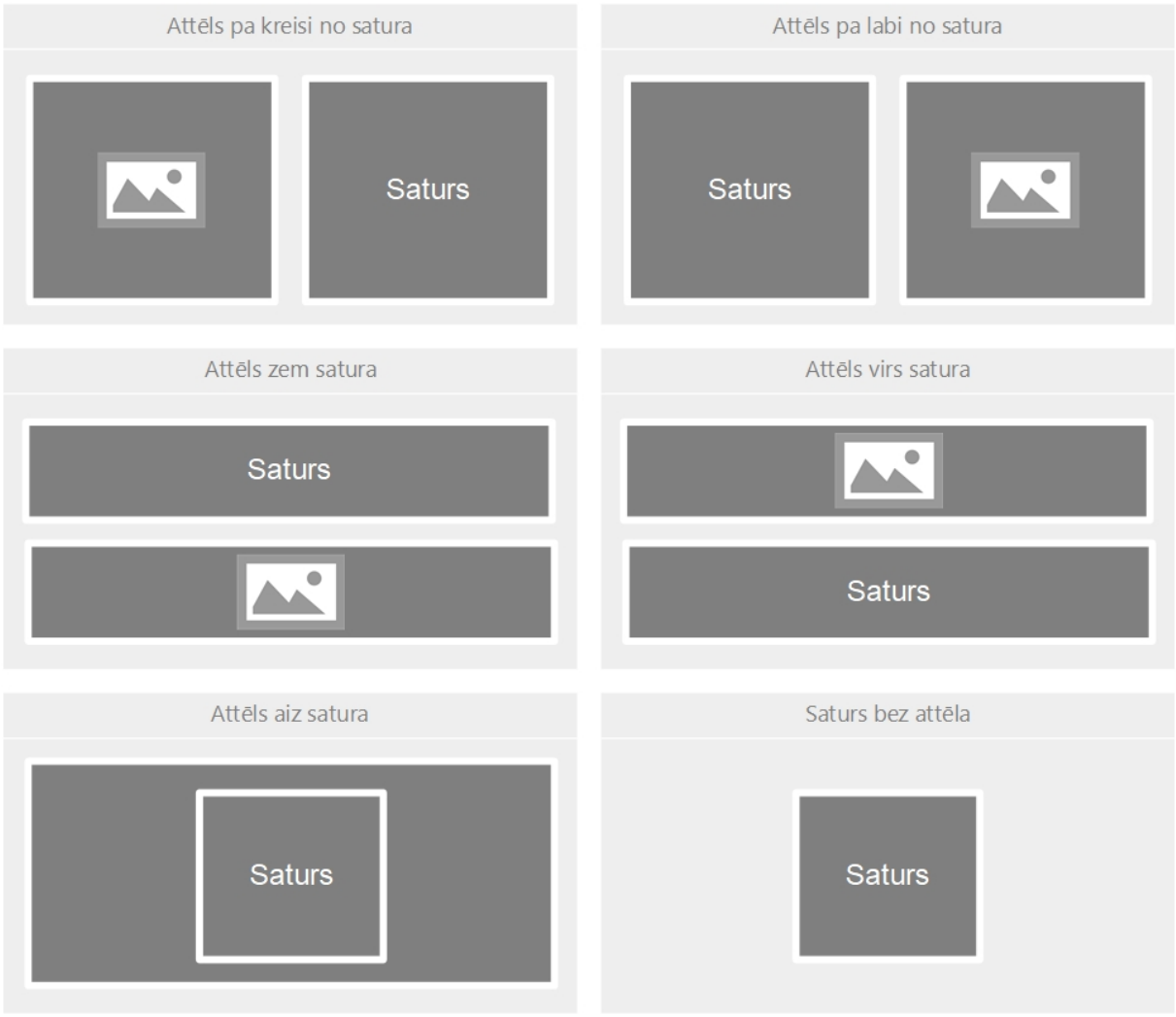


6.4. att. **Tīmekļa vietnes modulāra izveide**

Sistēmas pamatā tiek veidoti HTML moduļi, kur katrā modulī ir vienam lapas elementam atbilstošs, neatkarīgs HTML kods. Katrs modulis ir unikāls un neatkarīgs, tāpēc var tikt ievietots jebkurā lapas vietā bez papildus modifikācijām. Modulāra struktūra šādā sistēmā ir vitāla nepieciešamība, jo tīmekļa vietne tiek salikta no atsevišķiem blokiem un, lai iegūtu dažādus lapas izkārtojumus, elementu variācijas tiek veidotas, saliekot tos dažādās kombinācijās (6.4. att.). Modulārā struktūrā pamatā ir jāuztur moduļu bibliotēka un jādomā par katru moduli atsevišķi, nevis par visu lapu kopā. Viens no šādas sistēmas plusiem ir tas, ka izmaiņu nepieciešamības gadījumā ir jāmaina konkrēts modulis, nevis visa lapa un, tā kā modulis ir neatkarīgs, izmaiņas neietekmē citus moduļus.

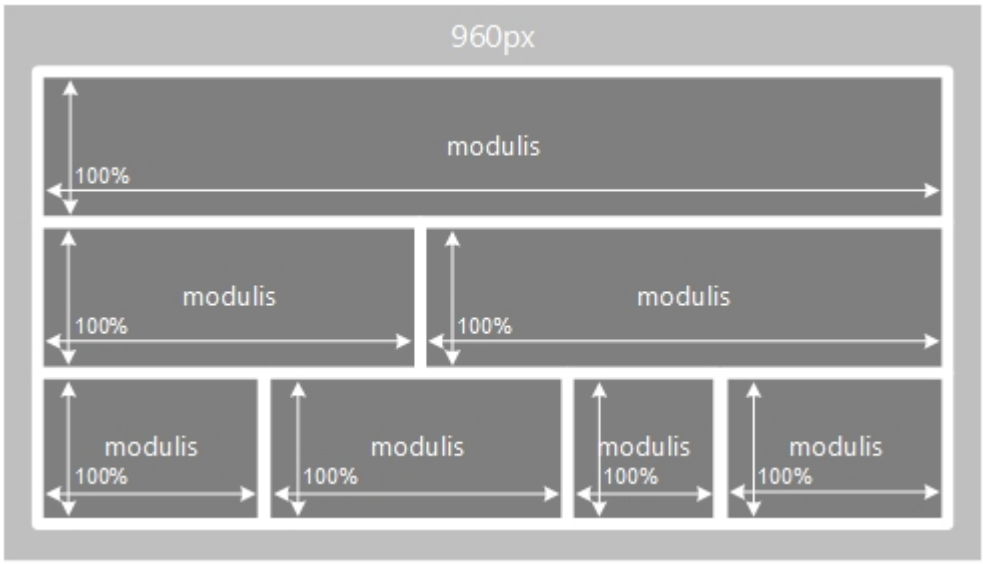
6.1.3. Satura izkārtojums

Modulāra sistēmas struktūra ļauj manipulēt ar elementiem un tos izvietot dažādās kombinācijās. Lai nodrošinātu unikālu dizainu, ir viegli savstarpēji kombinēt dažādus elementus un izveidot dažādus satura izkārtojumus. Pat ar nelielu elementu skaitu iespējams iegūt salīdzinoši lielu apjomu dažādu lapas izkārtojuma veidu. Augot moduļu bibliotēkas elementu skaitam, eksponenciāli aug lapas izkārtojumu skaits.



6.5. att. **Satura un attēla savstarpējais novietojums**

Ņemot par piemēru tikai divus elementus - satura bloku un attēla bloku - attēlā 6.5. redzams, ka tos ir iespējams savā starpā kombinēt dažādos veidos, radot 6 atšķirīgus satura izkārtojumus. Koda bāze ir viena: 2 moduļi, kurus apvienojot iespējams iegūt jaunas izkārtojuma kombinācijas, neveicot pašu elementu modifikāciju.



6.6. att. **HTML modulis atkārtoti izmantots dažādās pozīcijās**

Svarīgs aspekts moduļu izstrādē ir to pašpietiekamība un neatkarība no citiem moduļiem. Katrs HTML modulis aizņem 100% no tam piešķirtā platuma un 100% no tam piešķirtā augstuma (6.6. att.). Šādi moduļu izmēri nodrošina katra HTML elementa pielāgojamību dažādām situācijām. Modulis ir neatkarīga vienība, kurš aizpilda visu tam paredzēto vietu lapā un kuru nav nepieciešams papildus modificēt. Katrs modulis ir izmantojams dažādos gadījumos, un nav atsevišķi jādomā par elementu platumu un augstumu. Kombinējot divus dažādus moduļus, tie maina kopējo izkārtojumu atkarībā no elementu savstarpējās kompozīcijas. Attēlā 6.6. redzams, ka elementa modulis aizņem 50% no kopējā lapas platuma, ja tam blakus atrodas cits modulis, bet 100% no lapas platuma, ja tam blakus neatrodas citi moduļi. Moduļa relatīvais platums paliek nemainīgs: 100% no atvēlētās vietas.

6.2. Attēlu apstrāde

Tīmekļa vietnē svarīgi ir attēli. Attēls ir satura elements, kurš veido gan dizainu, gan lapas izkārtojumu, gan arī nodod svarīgu informāciju. Ģenerējot vietnes un radot izkārtojumu ar algoritma palīdzību, noteikti ir jāpievērš vairāk uzmanības attēliem. Izstrādājot tīmekļa vietni pēc pasūtījuma, tiek veidots dizains, tiek atlasīti atbilstoši attēli un apstrādāti, lai tie būtu pareizās proporcijās un izmēros, kā rezultātā attēls ir pielāgots dizainam un otrādi. Ģenerējot tīmekļa vietni, manuāla attēlu apstrāde nav iespējama, un jāņem vērā, ka attēli izmēru un kvalitātes ziņā atšķiras. Par vietnes attēliem rūpējas pats lietotājs, tāpēc var tikt augšupielādēti attēli ar dažādu izšķirtspēju, formu un izmēru.

6.2.1. Attēlu apgriešana

Lai panāktu estētisku lapas izskatu, attēliem jāiekļaujas dizainā, to nesabojājot. Viens attēls ģenerētajā tīmekļa vietnē tiek izmantots dažādās vietās un izmēros. Lai mainītu attēla izmēru, saglabājot tā proporcijas, tas ir jāapgriež. Ja attēlam piemēro citu izmēru bez apgriešanas, tad zūd attēla proporcijas un tas kļūst sagrozīts.

(a) (b) (c)

6.7. att. **Attēls dažādos izmēros:**

(a) oriģināls attēls, (b) attēls proporcijā 1:1 (c) izgriezts attēls proporcijā 1:1

Attēlā *6.7.* attēlota situācija, kur nepieciešams attēlu pielāgot izmēriem, kuru proporcijas ir 1:1. Izmainot attēla (a) platumu un augstumu, tiek iegūts attēls (b), kurš atbilst proporcijām, bet tiek vizuāli saspiests. Lai novērstu attēla izkropļošanu, nepieciešams to izgriezt pēc izmēriem, rezultātā iegūstot attēlu (c).

6.2.1.1. Saturu izprotoša attēlu apgriešana

Automātiska attēlu apgriešana, vadoties tikai pēc nepieciešamajiem izmēriem, ne vienmēr sniedz vēlamo rezultātu. Ja sistēma apgriež attēlu, neņemot vērā tajā attēloto, tad, iespējams, attēla apgrieztajā versijā nebūs redzami paši svarīgākie elementi vai apgrieztais attēls būs nederīgs.

(a) (b) (c)

6.8. att. **Attēls automātiski apgriezts pēc izmēriem:**

(a) oriģināls attēls, (b) attēls apgriezts bez satura atpazīšanas (c) attēls apgriezts ar satura atpazīšanu

Attēlā 6.8. redzama attēla (a) apgriešana, kurai nav pielietota satura atpazīšana (b) un attēla apgriešana, kurai ir pielietota satura atpazīšana (c). Kā redzams, attēls (b) ir apgriezts precīzi, izmēri ir ievēroti un proporcijas ir pareizas, bet galarezultāts ir nederīgs, jo attēlā nav redzama svarīgākā bildes daļa. Lai novērstu šādu iznākumu, attēlā (c) tiek izmantota satura atpazīšana, kur izmantota tā pati attēlu apgriešana kopā ar satura atpazīšanas algoritmu. Ieviešot šādu algoritmu, rezultāts ir daudz labāks: attēls saglabā izmērus, proporcijas un redzamu galveno daļu. Satura atpazīšanas algoritms izvērtē attēlu un izskaitļo galveno attēlā redzamo, un parūpējas par to, lai pēc apgriešanas attēla svarīgākā daļa būtu redzama.

6.2.1.2. Attēlu apgriešana izmantojot sejas atpazīšanu

Vairumā gadījumu attēlu apgriešanai pietiek ar pamata satura un svarīgāko objektu atpazīšanu. Taču gadījumos, kur attēlā ir redzami cilvēki, šāda pieeja var būt neapmierinoša. Cilvēki attēlos parasti ir svarīgākā daļa, un, ja attēls tiek izgriezts tā, ka redzama ainava, bet ne cilvēks, tad attēls ir nepilnīgs un apgriešana bijusi neveiksmīga.

Problēmas risinājums ir sejas atpazīšanas algoritmi, kuri nodrošina attēlā atrasto seju redzamību. Algoritms atpazīst sejas un atlasa attēla koordinātas, kuras apgrieztajā attēlā noteikti jāiekļauj. Pieeja ir noderīga arī portretos, kur tiek atrasta cilvēka seja un attēls tiek apgriezts tā, lai seja tajā būtu redzama.

6.3. PHP veidņu dzinējs

Izstrādājot modulāru sistēmu, kurā atkārtoti izmanto dažādus elementus un ģenerē kopējo sistēmas izkārtojumu un dizainu, ir svarīgi atdalīt sistēmas loģiku no attēlošanas loģikas. Koncepta gadījumā ir nepieciešama sistēmas arhitektūra, kura balstās uz atsevišķi uzturamiem attēlošanas elementiem, atsevišķu datu apstrādes un iekšējo sistēmas loģiku. PHP veidņu dzinējs nodrošina konsekventu sintaksi datu, kurus sagatavo aizmugursistēmas loģika, attēlošanai, rezultātā nodrošinot viegli modificējamus un vairākkārt izmantojamus, viegli lasāmus un uzturamus HTML moduļus, kuros nav sistēmas loģikas. Turklāt sistēmas aizmugurloģikas daļai nav jāuztur sistēmas skati, tikai jāapstrādā un jāpadod dati [35].

HTML sintakses nepilnība ir tāda, ka tā nepiedāvā dabisku veidu, kā iekļaut atsevišķus HTML elementus vienā lapā. Modulāras sistēmas gadījumā, kurā HTML struktūra ir sadalīta atsevišķos moduļos un gala produkts tiek sakombinēts no dažādiem moduļiem, ir nepieciešams rīks, kurš spēj tīmekļa vietni salikt kopā no atsevišķiem failiem un padot tiem datus. Tad tiek izmantots PHP veidņu dzinējs, kurš nodrošina atsevišķu failu iekļaušanu lapā un datu padošanu no sistēmas aizmugurloģikas.

PHP veidņu dzinējam ir vairākas koncepta sistēmas izstrādē nepieciešamas funkcijas. Funkciju klāsts atkarībā no dzinēja var mainīties, bet tā galvenā ideja un pielietojums pamatā ir nemainīgs:

- PHP un HTML faili ir fiziski atdalīti. Sistēmā tiek atsevišķi uzturēti HTML faili, kuri nodrošina sistēmas skatus, un PHP faili, kuri nodrošina sistēmas loģiku.
- HTML failu iekļaušana. Sistēma balstās uz HTML moduļu bibliotēku, un PHP veidņu dzinējs nodrošina HTML failu iekļaušanas funkciju.
- Nepārtraukta datu padošana. PHP veidņu dzinējs nodrošina konsekventu sintaksi un veidu, kādā tiek padoti dati no sistēmas aizmugurloģikas uz sistēmas skatiem.
- HTML attēlošanas loģika. PHP veidņu dzinējs nodrošina HTML failus ar papildus loģikas elementiem, ar kuru palīdzību iespējams manipulēt ar padotajiem datiem.
- Koda pārizmantošana. PHP veidņu dzinējs ļauj atkārtoti izmatot HTML veidnes, padodot tām dažādus datus.
- Modularitāte. PHP veidņu dzinējs palīdz strukturēt sistēmu modulārā veidā, kur sistēmas daļas ir atdalītas viena no otras.

PHP veidņu dzinējs sniedz vairākas būtiskas funkcijas, bet konkrētā veidņu dzinēja izvēle jābalsta uz sistēmas vajadzībām. Atkarībā no konkrēta PHP veidņu dzinēja tiek piedāvātas tam specifiskas funkcijas, kuras var ietekmēt kopējo sistēmas struktūru un realizāciju.

6.4. CSS pirmsprocesors (*preprocessor*)

CSS ir primitīva un nepilnīga valoda, tajā trūkst noderīgu īpašību un iezīmju, kuri atvieglotu tās izmantošanu. Funkciju izveide, koda atkārtota izmantošana vai elementu mantošana ar standarta CSS ir ļoti grūti sasniedzama. Projektam kļūstot lielākam un sarežģītākam CSS pārizmantojamība kļūst arvien grūtāka. Arī tīmekļa tehnoloģijas nemitīgi attīstās, tīmekļa pārlūki ievieš jaunu elementu atbalstu, HTML un CSS iepazīstina ar jaunām specifikācijām, beigās radot dažādiem elementiem, kurus ir grūti uzturēt [40].

Pirmsprocesori nodrošina papildus CSS iespējas, lai tas būtu ērtāk izmantojams, vieglāk uzturams un pārizmantojams. CSS pirmsprocesoriem un standarta CSS ir atšķirīga sintakse, bet pēc būtības tā līdzinās CSS oriģinālajai sintaksei. Ir vairāki CSS pirmsprocesori, piemēram, *Sass*[31], *Less*[32], *Stylus*[33] u.c., bet tiem ir kopīgas iezīmes, kuras padara to izmantošanu par nepieciešamību [40, 41].

[31] http://sass-lang.com/
[32] http://lesscss.org/
[33] http://stylus-lang.com/

Primsprocesori nodrošina šādus funkcionalitāti [40, 41]:

- Mainīgie: iespēja definēt CSS mainīgo vērtības, piemēram, pamata krāsas, fonta izmērs, platums, garums u.c.
- Koda iegulšana: tā kā CSS nenodrošina vizuālu elementu hierarhiju, tas ir grūti lasāms. Koda iegulšana nodrošina vizuālu hierarhiju un uzlabo koda lasāmību.
- Definīcijas: definīcijas ir funkcijām līdzīga struktūra, definīcijām var padot parametrus, pēc kuriem tiek ģenerēts CSS kods.
- Iekļaušana: pirmsprocesori ļauj iekļaut iepriekš definētus selektorus un to atribūtus, ļaujot pārizmantot stilu, tādējādi samazinās uzturamais koda apjoms.
- Loģikas atribūti: tiek nodrošināti loģikas elementi, piemēram, *if*, *else* izteiksmes un matemātiskās darbības.
- Failu importēšana: ar pirmsprocesoru palīdzību iespējams CSS failu sadalīt vairākos failos un tos importēt pēc nepieciešamības. Atsevišķus failus ir vieglāk uzturēt un labot, kā arī netiek lieki izmantots stils.

CSS pirmsprocesors ir atsevišķa programmēšanas valoda, kura izmanto CSS ideoloģiju, tai pievienojot noderīgas funkcijas. Pirmsprocesori ir efektīvi, jo tiek izmantoti CSS vietā. Projekta ietvaros tiek uzturēts tikai pirmsprocesora kods, kas ir mazāks un vieglāk uzturams, bet tā rezultāts ir tīrs CSS kods, ģenerēts no pirmsprocesora koda.

6.6. Sistēmas dati

Dati ir viena no svarīgākajām sistēmas komponentēm, jo tiešā veidā ietekmē sistēmas izskatu, bloku izvietojumu, ģenerētās vietnes krāsas un citus sistēmas aspektus. Dati ir sistēmas pamatā un tiek izmantoti kā pamats tālākai tīmekļa vietnes izveidei.

Sistēmas prototips neizmanto datu bāzi, tās vietā tiek izmantots JSON[34] datu formāts, lai nodrošinātu sistēmas modularitāti un savienojamību ar citām sistēmām. JSON ir populārs, vienkāršs un viegli modificējams datu formāts, ko atbalsta daudzas sistēmas. Tas ir viegli lasāms, saprotams, mašīnām vienkārši parsējams un ģenerējams. JSON formāts ir pilnībā neatkarīgs no programmēšanas valodām, bet tas izmanto vairākās programmēšanas valodās pazīstamas konvencijas [37].

JSON formātu atbalsta dažādas NoSQL datu bāzu pārvaldības sistēmas un PHP ietvari Tas tiek uzskatīts par nozares standartu darbā ar datiem, tāpēc ir stabilu izvēle prototipa izstrādei [38]. Kā piemēru JSON izmantošanai savstarpējā sistēmu integrācijā var minēt

[34] http://www.json.org/

WordPress[35] REST API[36]. *WordPress* REST API atgriež datus JSON formātā, tāpēc *WordPress* var tikt izmantots kā aizmugursistēma, kas rūpējas par datiem, apvienojumā ar sistēmas prototipu, kurš apstrādā un vizualizē datus [39].

Sistēmas moduļi tiek izstrādāti vienreiz neatkarīgi no tā, kāds saturs tajos tiek ievietots. Moduļi tiek veidoti tā, lai tie būtu spējīgi piemēroties dažādām saturam un satura kombinācijām. Tam jābūt dinamiskam, lai derētu gan saturam ar 50 vārdiem, gan ar 300 vārdiem. Šādā pieeja nodrošina to, ka lietotāja pieredze būs nemainīga un sistēmā netiek uzturēti atsevišķi moduļi dažādiem satura parametriem, bet viens modulis, kurš spēj pielāgoties dažādām situācijām. Šādā veidā sistēma ir vieglāk uzturama un apjoma ziņā mazāka.

Izstrādājot sistēmu klasiskajā veidā, bieži vispirms tiek domāts par saturu un tikai pēc tam par tā stilu. Ģenerējot vietnes stilu un izkārtojumu, ir jādomā otrādi: vispirms par stilu, pēc tam par saturu. Vienam satura blokam ir jāspēj pielāgoties visām potenciālajām iespējām. Izstrādājot HTML moduli, ir jāparedz vieta dažādiem satura elementiem, kā arī to neesamības iespēja. Par piemēru var ņemt tīmekļa vietnes rakstam paredzētu moduli. Raksts var saturēt attēlus, tajā var tikt norādīts autors, datums, atsauces un citi elementi, par visiem šiem elementiem bloku izstrādē atsevišķi jādomā. Bet vietnes raksts var saturēt arī tikai tekstu, līdz ar to modulis ir jāpielāgo dažādām situācijām, lai to būtu iespējams izmantot neatkarīgi no tam padotā satura.

6.7. Satura analīze

Sistēmā liela nozīme ir saturu raksturojošām iezīmēm. Lai izvairītos no gadījumiem, ka ģenerētais tīmekļa vietnes izkārtojums un stils nespēj pielāgoties saturam, jāveic satura īpašību analīze. Satura fizisko īpašību analīze notiek vairākos līmeņos, jo pats saturs sastāv no daudziem elementiem. Katrs atsevišķs elements nosaka tam piemērojamo stilu un izkārtojumu, un visi elementi kopā ietekmē saturam adekvātu stilu.

Viens no analizētajiem satura parametriem ir elementi un to skaits. Dažādiem lapas izkārtojumiem nepieciešami dažādi elementi un to skaits. Piemēram, izkārtojumā, kur fonā tiek izmantoti vairāki attēli, ir svarīgi, lai saturā šie attēli būtu. Līdzīgi ir ar citiem elementiem, navigācijas elementu skaits ietekmē tai piemēroto moduļu veidu. Ja satura rakstam ir pogas, attiecīgi jāpiemeklē moduļi, kuri atbalsta raksta dizainu ar pogām. Vēl viens aspekts ir satura apjoms un garums. Tekstam ar dažādu garumu ir piemērojami dažādi moduļi, kuri tam atbilst.

[35] https://wordpress.org/
[36] https://developer.wordpress.org/rest-api/

6.7.1. Teksta analīze

Teksts ir svarīgākais tīmekļa vietnes satura elements, kurš gan sniedz daudz informācijas lasītājam, gan arī informē par pašu tīmekļa vietni, tās nozīmi un mērķi. Teksta analīze ir sarežģīts process, bet tās rezultāti ir ļoti noderīgi, lai izstrādātu saturam atbilstošu vietnes dizainu. Teksta analīzes rezultātā tiek iegūti šādi parametri:

- **Vārdu skaits:** tiek noskaidrots katra satura teksta vārdu skaits, kas ietekmē dizainu, nosakot piemērotāko dizainu garākam vai īsākam tekstam. Lielam vārdu skaitam vajadzīgs dizains, kas atbalsta bloga tipa struktūru, turpretī nelielam vārdu skaitam var piemērot citu dizainu.
- **Atslēgas vārdi**: vadoties pēc atslēgas vārdiem, var noteikt satura tematu un piedāvāt tam atbilstošu dizainu. Piemēram, ja atslēgas vārdos tiek minēta daba, dzīvnieki vai augi, tad piemērots ir dabas tematikai atbilstošs dizains.
- **Vārdu klasifikācija:** veicot vārdu klasifikāciju, iespējams noteikt analizējamā teksta elementus. Tekstā var būt, piemēram, personas, datumi, organizācijas, kas palīdz atbilstoša dizaina izvēlē.

Teksta analīze nav precīza, un tās rezultāti var būt dažādi. Ir viegli noteikt teksta vārdu skaitu, kas palīdz dizaina izveidē, bet, nosakot atslēgas vārdus un vārdu klasifikāciju, iegūtos rezultātus var būt grūti apstrādāt un izmantot dizaina lēmumu pieņemšanā.

6.7.2. Attēlu analīze

Attēli ir nozīmīgs satura elements, tie var būt pašsaprotami, papildināt saturu un veikt citas funkcijas satura kontekstā. Lai ģenerētu dizainu, kurš lieto satura attēlus, ir jāveic to parametru analīze. Attēlu analīzes rezultātā tiek iegūti šādi parametri:

- **Attēla izmēri**: pēc attēla izmēriem ir iespējams noteikt, cik lieli attēli izmantoti. Ja tiek izmantoti lieli attēli, tad ģenerētais dizains var izmantot šos attēlus tīmekļa vietnes fonā un citos elementos, veidojot krāšņu dizainu. Ja saturā ir mazi attēli, tad dizains tiek veidots bez lieliem attēliem: attēli papildina tekstu, bet tie netiek izmantoti vietnes fonā un citos elementos.
- **Attēlu skaits**: ja saturā ir daudz attēlu, tad tiek piemērots dizains, kurš tos izceļ un ievieto dažādos elementos. Ja attēlu skaits ir neliels, tad dizainā tiek likts lielāks uzsvars uz tekstu..
- **Attēla saturs:** attēla saturs, līdzīgi kā teksta saturs, var ietekmēt dizaina izveidi. Ja attēlos ir, piemēram, portreti, ainavas vai dzīvnieki, tad var izvēlēties šādam saturam atbilstošu dizainu.

- **Attēla krāsas:** attēla krāsas ietekmē apkārtējo elementu stilu un konkrētu bloku izskatu. Attēla akcentu krāsas var izmantot dizaina elementos, tos papildus izceļot. Attēla pamatkrāsas izmantojamas kā bloku fons un citu elementu noformējums.

Attēli ietekmē gan saturu, gan dizainu. Analizējot attēlus un pielāgojot tiem dizainu, iespējams izveidot funkcionālas un saturam atbilstošas kombinācijas. Attēlu analīze, līdzīgi kā teksta analīze, var sniegt rezultātus, kuri neatbilst dizaina elementiem.

7. SISTĒMAS PROTOTIPA IZSTRĀDE

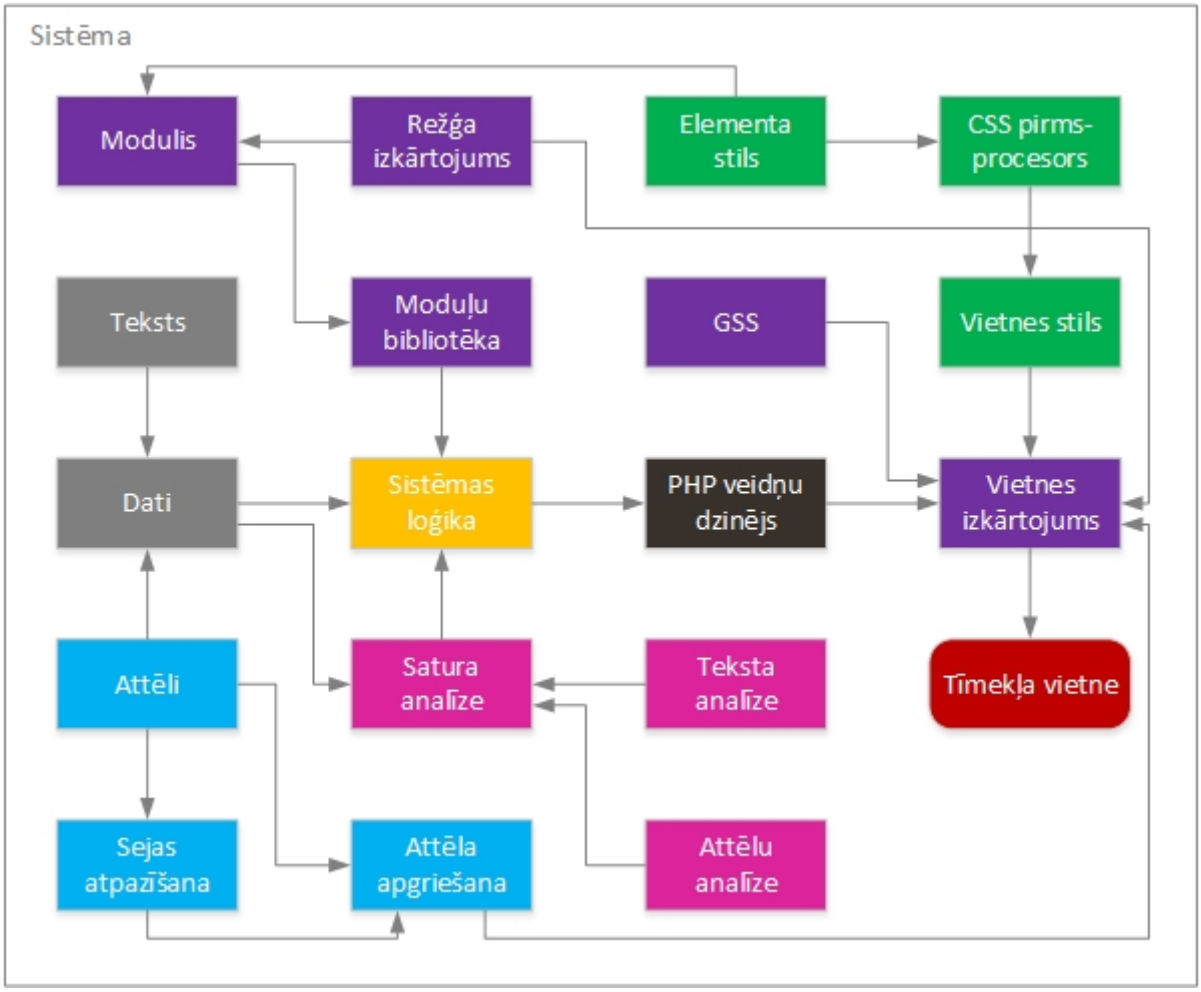
Izpētot problēmu, tika apskatīti vairāki esošie risinājumi un to galvenās idejas. Tika apkopoti rīki un pieejas, kuras varētu noderēt risinājuma izstrādē un gūtas plašākas zināšanas šādu un līdzīgu sistēmu strukturēšanā un izveidē. Lai atbildētu uz problēmas jautājumiem un risinātu pētījumā konstatētās problēmas, tika nolemts izstrādāt savu risinājumu, kurš apmierina izvirzītās prasības. Praktiskai pieredzei, zināšanu un teorētiskā risinājuma koncepta pielietojumam reālos dzīves apstākļos tika nolemts izveidot nelielu risinājuma prototipu, kuram jāparāda, ka piedāvātais koncepts ir īstenojams. Resursu taupīšanas dēļ tika izlemts izmantot pēc iespējas vairāk trešo pušu risinājumu.

Risinājuma prototips ģenerē tīmekļa vietnes izkārtojumu un stilu, vadoties pēc satura analīzes, kā rezultātā tiek piedāvāts unikāls un optimāls dizains. Sistēma no lietotāja teksta izgūst atslēgas vārdus, uz kuru pamata piedāvā atbilstošākos vietnes izkārtojumus konkrētajam saturam. No piedāvātajiem attēliem tiek izgūta krāsu palete, kura tiek izmantota vietnes stila ģenerēšanai. Attēli tiek papildus apstrādāti, lai tie spētu iekļauties kopējā vietnes stilā.

Risinājuma prototips ir izstrādāts ar PHP, darbam ar HTML failiem izmantots PHP veidņu dzinējs *Twig*[37]. Prototips izmanto SASS pirmsprocesoru darbam ar CSS failiem. Ģenerētās vietnes izkārtojumam lietots *Bootstrap*[38] un tā režģa sistēma, kura nodrošina satura dalīšanu kolonnās. Papildus elementu pozicionēšanai tiek izmantots GSS, lai definētu elementu savstarpējās attiecības un kopējo novietojumu lapā. Manipulācijai ar DOM elementiem tiek izmantots *JavaScript* un *jQuery*[39]. Attēlu apgriešanai tiek izmantota *JavaScript* bibliotēka *smartcrop.js*[40], seju atpazīšanu nodrošina *jQuery* bibliotēka *jQuery Face Detection*[41] un *Tracking.js*[42]. Prototips datu uzglabāšanai izmanto JSON failu struktūru. Prototips vietnes izkārtojumu un stilu ģenerē procedurāli. Sistēmas struktūra veidota uz MVC arhitektūras pamata, kas izdala atsevišķi sistēmas loģiku un attēlošanu.

Risinājums sastāv no 18 atsevišķām komponentēm, kuras atbildīgas par konkrētu funkciju un visas kopā veido sistēmas loģiku. Vadoties pēc 5.3. nodaļā aprakstītās modularitātes sistēmas, struktūra ir sadalīta, atvieglojot sistēmas izstrādi, pilnveidi un uzturēšanu. Attēlā 7.1. attēlotas sistēmas komponentes un to savstarpējās attiecības. Attēlā ieskicēta datu plūsma starp dažādās komponentēm.

[37] https://twig.sensiolabs.org/
[38] http://getbootstrap.com/
[39] https://jquery.com/
[40] https://github.com/jwagner/smartcrop.js/
[41] https://github.com/jaysalvat/jquery.facedetection
[42] https://trackingjs.com/



7.1. att. **Sistēmas komponentes**

Tīmekļa vietnes ģenerēšana ir sadalīta vairākos soļos, un katrā solī tiek veikta konkrēta darbība, kuras rezultāts ir nākamā soļa pamatā. Nepieciešamības gadījumā viens vai vairāki soļi var tikt izlaisti, piemēram, ja saturā nav attēlu vai nepieciešama tikai atsevišķu bloku ģenerēšana. Risinājuma prototips ģenerē tīmekļa vietni 5 soļos:

1. **Satura analīze un marķēšana**

   Mērķis: Veikt satura analīzi un marķēšanu, balstoties uz vietnes mērķi, teksta garumu, nozīmi, attēlu izmēriem, kontekstu u.c.

   Nozīme: Vadoties pēc marķējumiem, tiek nodrošināta iespēja pieņemt atbilstošus lēmumus par vietnes dizainu un izkārtojumu.

2. **Kompozīcijas izveide**

   Mērķis: Izveidot lapas kompozīciju, vadoties pēc satura. Tiek izveidota satura plūsma un noteikta satura aizņemtā vieta lapā, ņemot vērā marķējumus un lapas nozīmi.

Nozīme: Tiek piedāvāta lapas kompozīcija, kura atbilst saturam un pēc kuras tiek veidots lapas izkārtojums. Izkārtojumā tiek izcelts galvenais saturs attiecībā pret citiem elementiem.

3. **Izkārtojuma izveide**

   Mērķis: Izveidot lapas izkārtojumu, vadoties pēc satura plūsmas un marķējumiem.

   Nozīme: Tiek izveidots lapas izkārtojums.

4. **Stila piešķiršana**

   Mērķis: Stila definēšana (krāsas, fonti, dekorācijas un atstarpes) katrai sekcijai un kopējam izkārtojumam.

   Nozīme: Individuālu bloku stila izveide un visu bloku kopējā dizaina izveide.

5. **Attēlu apstrāde**

   Mērķis: Apstrādāt attēlus, veicot to analīzi.

   Nozīme: Attēlu apstrāde palīdz izveidot vietnei vienotu stilu, kur saturs ir savstarpēji integrēts un iederīgs.

Visu soļu veiksmīgas izpildes gadījumā tiek attēlota tīmekļa vietne ar tai ģenerēto izkārtojumu un stilu. Prototipa īstenošanas gadījumā soļi nav fiziski atdalīti, tie savā starpā integrējas un viens solis var tikt veikts paralēli otram, piemēram, izkārtojuma izveide notiek paralēli stila piešķiršanai.

7.1. Moduļu bibliotēka

Svarīga risinājuma sastāvdaļa ir moduļu bibliotēka. Tā sastāv no atsevišķiem HMTL moduļiem un tiem piederīgā CSS stila. Katrs konkrētais modulis atrodas savā datnē, kura nosaka moduļa veidu (piemēram – sekcija, attēls, raksts u.c.), un katram modulim ir savs fails ar tā HTML kodu, kā arī atsevišķs fails ar konkrētā moduļa izkārtojumam un pamata stilam nepieciešamo CSS kodu.

Moduļi un to datnes ir grupēti atsevišķos līmeņos. Katrs modulis vienā līmenī ir pašpietiekams un neatkarīgs no citiem moduļiem. Katrs augstāka līmeņa modulis sevī ietver zemāka līmeņa moduļus. Moduļi tiek iedalīti līmeņos, lai netiktu vairākkārt uzrakstīts viens koda gabals, kurš var atrasties dažādās vietās. Par piemēru var ņemt raksta moduli (attēls 7.2.), kur konkrētais modulis satur divus zemāka līmeņa moduļus: attēla moduli un sekcijas moduli. Šim raksta modulim ir vairākas variācijas, bet katra variācija satur attēlu un sekciju. Lai katrā atsevišķajā raksta modulī netiktu dublēts attēla un sekcijas kods, tas tiek iznests zemāka līmeņa modulī, kurš iekļauts šajā augstāka līmeņa raksta modulī. Ieguvums jūtams, pieaugot koda apjomam. Ja, piemēram, ir definēti 10 dažādi raksta moduļi, tad 10 atšķirīgu attēla kodu gabalu un 10 sekcijas kodu gabalu vietā ir tikai 1 attēla un 1 sekcijas modulis, rezultātā kods tiek

saīsināts 9 reizes. Sistēmai palielinoties un nodrošinot arvien jaunus izkārtojumus, šāda struktūra ir ļoti vērtīga.

```
<article class="row article-3">
    <div class="row article-row">
        {% for key, section in post.data.sections %}
            {% if section.image%}
                {% include 'partials/article/image/image-'~ post.image_template[0] ~'.twig' with {
                    'image': section.image
                } %}
            {% endif %}
        {% endfor %}
    </div>
    <div class="row article-row">
        {% for key, section in post.data.sections %}
            {% if section.title%}
                {% include 'partials/article/section/section-'~ post.section_template ~'.twig' with {
                    'section': section
                } %}
            {% endif %}
        {% endfor %}
    </div>
</article>
```

7.2. att. **Raksta modulis**

Katra moduļa nosaukums ir unikāls. Tas sastāv no moduļa pamata satura (attēls, raksts, teksts, sekcija, lapa u.c.) un moduļa koda, pēc kura var atlasīt specifisku moduli. Izstrādāta moduļu bibliotēka, kur katra moduļa unikālais identifikators (article-0, article-1, article-2, image-0, image-1 utt.) apzīmē moduļa variāciju. Atlasot moduļus, tiek izvēlēts modulis ar konkrētu nozīmi un numuru. Šāda nosaukumu struktūra izveidota ērtai moduļu variāciju atlasei. Veidojot izkārtojumu, augstāks modulis ietver citus moduļus, piemēram, satura modulis ietver attēla moduli. Attēla modulim ir vairākas variācijas (image-0, image-1 u.c.), bet tā nosaukuma pamats ir viens (image-), tāpēc, atlasot konkrētu variāciju, jāpadod tikai vēlamā moduļa numurs.

Katram modulim ir atbilstošs masīvs ar tā īpašībām. Moduļa īpašības apraksta dažādus parametrus, kuri ir svarīgi izkārtojuma un stila ģenerēšanā. Moduļu parametri apraksta to, kāds saturs ar šo moduli ir savietojams, kam tas ir paredzēts un ko tas atbalsta. Dažādiem moduļiem var būt dažādi to raksturojoši parametri, bet pamatā visi moduļi balstās uz sekojošiem parametriem:

- **Moduļa kods**: nepieciešams, lai identificētu moduli;
- **Moduļa izmērs**: modulim var būt dažādi izmēri, piemēram, teksta modulis var atbalstīt dažādu garumu tekstus, ja tas spēj pielāgoties saturam ar 500 vārdiem, tad moduļa izmērs ir 500. Cita moduļa izmērs var būt atkarīgs no citiem faktoriem, piemēram, navigācijas moduļa izmērs atkarīgs no navigācijas elementu skaita, ko tas spēj izmantot.

- **Atslēgas vārdi**: specifiskiem moduļiem ir atslēgas vārdi, kuri apraksta to nozīmi, piemēram, modulis var būt atslēgas vārds "portrets", kas nozīmē to, ka šis modulis atbilst saturam, kurā ir attēls, kurā redzams cilvēka portrets, vai, piemēram, modulis ir piemērots lieliem attēliem, tad tam tiek pievienots atslēgas vārds "galerija";
- **Attēlu skaits**: dažādi moduļi var atbalstīt dažādu skaitu attēlu, piemēram, ja modulis kā fona attēlu izmanto 4 attēlu kombināciju, tad moduļa attēlu skaits ir 4;
- **Moduļa elementi**: modulim var būt atsevišķi elementi, kurus tas atbalsta. Raksta modulim var būt autors, datums, attēls, teksts vai citi elementi. Sekcijas modulim var būt pogas, forma u.c. elementi.

Moduļa parametri kalpo kā palīglīdzeklis, lai ģenerētu atbilstošāku tīmekļa vietnes stilu. Veicot satura analīzi tiek izgūti dažādi saturu raksturojoši parametri (7.5. nodaļa). Salīdzinot satura parametrus ar moduļu parametriem var atrast piemērotākos stila risinājumus. Ne vienmēr moduļi atbildīs saturam un saturs moduļiem, tādēļ parametri ir kā pievienotā vērtības un moduļu uzbūve atbalsta dažādas satura variācijas. Ja saturam ir 100 vārdi, vai 300 vārdi, tas moduļa kontekstā neko daudz var neietekmēt, līdz ar to moduļi ir universāli un tos var izmantot dažādās situācijās.

Moduļu bibliotēku veido vairākas atsevišķas komponentes, kuras tiek aprakstītas tālākajās nodaļās. Moduļu bibliotēka balstās uz PHP veidņu dzinēju *Twig*, ar kuru HTML kodu var dalīt atsevišķos blokos un veikt citas darbības. *Bootstrap* un tā režģa izkārtojums nodrošina efektīvu izkārtojuma dalīšanu neatkarīgos blokos, un SASS piedāvā dalīt CSS kodu un citas noderīgas iespējas.

7.1.1. Twig

Twig ir PHP veidņu dzinējs, kura mērķis ir atdalīt sistēmas loģiku no tās attēlojuma. *Twig* sadala HTML un PHP kodu divās atsevišķās daļās un izveido konsistentu datu padošanas mehānismu starp tām, kā arī nodrošina prototipa izstrādē būtiskas papildus funkcijas. *Twig* tika izvēlēts, jo autors ir ar to strādājis iepriekš, tas ir labi dokumentēts un nodrošina nepieciešamās funkcijas sekmīgai sistēmas prototipa īstenošanai. *Twig* vietā iespējams izmantot arī citus PHP veidņu dzinējus, bet šajā gadījumā tie netiek apskatīti [43].

Lietojot *Twig,* HTML struktūru iespējams dalīt atsevišķos moduļos jeb veidnēs. *Twig* nodrošina modulāru struktūru un koda pārizmantojamību, kas ir svarīgi faktori nodaļā 5.3. aprakstītajā risinājuma izveidē. Prototipa moduļu bibliotēka balstās uz *Twig* failiem, kuri pēc būtības ir HTML faili ar papildus loģikas funkciju. Attēlā 7.2. ir redzams *Twig* fails, kurā iekļauta standarta HTML struktūra un papildus attēlošanas loģika, kā, piemēram, "*for*" cikls, "*if*" un "*else*" deklarācijas u.c. *Twig* veidnēs var iekļaut citas veidnes un padot tām

nepieciešamos datus. Prototipa gadījumā ar *Twig* palīdzību veidota moduļu bibliotēka, kura abstrahējas no datu apstrādes un aizmugursistēmas loģikas, tādā veidā nodrošinot citkārt grūti īstenojamu atsevišķu moduļu neatkarību un pārizmantošanu.

7.1.2. Bootstrap un režģa izkārtojums

Bootstrap ir HTML, CSS un JS ietvars, paredzēts responsīvu vietņu izstrādei. *Bootstrap* satur plašu HMTL elementu klāstu un responsīvu režģa sistēmu. *Bootstrap* iekļauj responsīvu, 12 kolonnu izkārtojumā balstītu plūstošā režģa sistēmu[43]. Režģa sistēma izmanto iepriekš definētas klases un miksinus, lai nodrošinātu vienkāršu, efektīvu satura izkārtojumu. Ar režģa palīdzību vietnes saturs tiek izkārtots rindās un kolonnās [44].

Bootstrap sistēmas prototipā tiek izmantots režģa sistēmas un piedāvāto elementu dēļ. Ar režģa sistēmu var efektīvi veidot vietnes struktūru, balstoties uz atsevišķiem elementiem. Risinājums ģenerē vietnes izkārtojumu no atsevišķiem blokiem. Lai atvieglotu bloku savstarpējā novietojuma un attiecību izveidi, tiek izmantota *Bootstrap* režģa sistēma. Pateicoties režģim, katrs HTML modulis ir neatkarīgs un pašpietiekams. Pats modulis aizņem 100% tam atvēlētās vietas, līdz ar to tas pielāgosies dažādām vietām, radot atbilstošu izkārtojumu. Šāda pieeja ļauj uzturēt vienu moduli vairākiem gadījumiem, katrs modulis var ieņemt no 1 līdz 12 kolonnām, plašāk koncepts aprakstīts 6.1.1. nodaļā.

Risinājuma gadījumā iespējams izmantot teju jebkuru režģa sistēmu vai līdzīgus risinājumus. *Bootstrap* un tā režģu sistēma prototipā tiek izmantota, jo autors ar to ir iepriekš strādājis, tā ir labi dokumentēta, plaši izmantota un atbilst visām vajadzībām. *Bootstrap* priekšrocība ir iespēja izmantot tā elementus un elementu stilu vietnes izkārtojuma un stila ģenerēšanai. *Bootstrap* un tā režģa sistēma atbalsta responsīvu dizainu izveidi, rezultātā nav nepieciešams atsevišķs risinājums un papildus resursi, lai nodrošinātu vietnes izkārtojumu mobilām ierīcēm.

7.1.3. SASS

SASS ir CSS papildinājumu bibliotēka, kas standarta CSS sintaksei pievieno dažādas iespējas – mainīgie, pārmantošana, miksini, mantošana un citi, kas padara SASS par spēcīgu rīku. SASS ir CSS pirmsprocesors ar atsevišķu, CSS balstītu, bet nedaudz atšķirīgu sintaksi. SASS rezultāts ir tīrs CSS, ģenerēts no SASS koda, plašāk par CSS pirmsprocesoru konceptu aprakstīts nodaļā 6.4.

Prototipa ietvaros SASS pilnībā aizvieto CSS kodu (tas vēlāk tiek ģenerēts no SASS). SASS ļauj vietnes stilu dalīt atsevišķos blokos un tos pārmantot, kas prototipa gadījumā ir

[43] http://getbootstrap.com/css/#grid

noderīgi. Katra *Twig* moduļa SASS stils (fails) apraksta konkrētā moduļa atribūtus (izkārtojumu, novietojumu, iekšējo elementu attiecības u.c.). Ar SASS iespējams vienu stila elementu iekļaut dažādās vietās un pārizmantot, tādējādi nodrošinot sistēmas modulāro uzbūvi un moduļu bibliotēkas struktūru. Svarīga SASS priekšrocība ir mainīgo izveide. Ar SASS tiek izveidoti sistēmas stila mainīgie, kurus izmanto gan atsevišķi moduļi, gan kopējais lapas stils. SASS ļauj definēt lapas krāsu paleti, fontus, izmērus un citus elementus. Prototipa gadījumā vienreiz tiek ģenerēta lapas krāsu palete, un SASS ar mainīgo palīdzību nodrošina šīs paletes izmantošanu visiem elementiem. Noderīgs SASS atribūts ir elementu hierarhijas koks, kas vizuāli attēlo elementu stila struktūru un samazina uzturamo elementu skaitu.

Risinājuma gadījumā SASS vietā var izmantot citus CSS pirmsprocesorus, svarīgi, lai tiktu nodrošināta mainīgo definēšana, stila dalīšana atsevišķos failos un elementu stila atkārtota izmantošana. SASS tika izvēlēts, jo tas ir viens no attīstītākajiem CSS pirmsprocesoriem, kurš tiek pilnveidots jau 9 gadus [45] un bieži izmantots nozarē, tam ir plašs atbalsts un stabila dokumentācija. Vēl viens iemesls SASS izvēlei ir autora darba pieredze ar šo bibliotēku.

7.1.4. Grid Style Sheets (GSS)

GSS līdzīgi kā SASS ir CSS pirmsprocessors, bet papildus pirmsprocessora sniegtajām priekšrocībām GSS izmanto JavaScript, lai manipulētu ar elementiem. Plašāk par GSS ir aprakstīt nodaļā 3.2.2., kur aprakstīta tā funkcionalitāte un pamatprincipi.

Prototipa ietvaros GSS tiek izmantots, lai aprakstītu elementu savstarpējās attiecības. GSS ir ērts rīks ar kuru pozicionēt savstarpēji saistītus elementus. Ar GSS ir viegli aprakstīt elementu savstarpējās attiecības un novietojumu. GSS var izmantot, lai vertikāli un horizontāli nocentrētu atsevišķus elementus. Moduļu gadījumā konkrētam modulim var būt definēts viens izmērs, bet saturs, kurš tajā tiek ievietots var būt dažāds, piemēram, teksts var būt ar lielu vārdu skaitu un aizpildīt visu moduli, vai teksts var saturēt lielu vārdu skaitu un aizpildīt tikai daļu no moduļa. Ja par piemēru ņem gadījumu ar nelielu vārdu skaitu, tad teksts izskatīsies labāk, ja tas būs vertikāli nocentrēts blokam pa vidu, nevis atrastos bloka augšpusē atstājot tukšu bloka apakšdaļu. Ar GSS var risināt šādus gadījumus, nosakot to, ka elementam jābūt centrētam vertikāli bloka ietvaros. Papildus minētajām piemēram, GSS var izmantot arī elementu izmēriem. Vadoties pēc konkrēta elementa izmēriem ar GSS var noteikt citu elementu izmērus.

GSS dinamiska satura gadījumā palīdz pozicionēt elementus un savstarpēji savienojot dažādus elementus ir iespējams iegūt tīmekļa vietnes stilu, kurš spēj pielāgoties dažādiem gadījumiem atrisinot atsevišķu elementu novietošanas problēmas.

7.2. Attēlu apstrāde

Reti kurā saturā nav iekļauti vizuāli materiāli – attēli. Tie aizņem daudz vietas, tiek izmantoti kā fons, paskaidrojums, galerija vai cits elements. Attēli ir ne tikai satura daļa, bet arī izvietojuma elementi, uz kuriem balstās kopējā vietnes struktūra. Ideālos apstākļos attēli ir vienāda izmēra, tie ir profesionāli uzņemti un atbilst visām vadlīnijām, bet risinājuma gadījumā attēlus nodrošina lietotājs. Lietotāja nodrošināti attēli nav standartizēti, tiem ir dažāds izmērs, formāts, izšķirtspēja, krāsas un mērķis. Ģenerējot tīmekļa vietni, noteikti ir jāpievērš pastiprināta uzmanība šim satura veidam. Attēli glabā daudz informācijas, un jebkurā dizainā ir svarīgi, lai šī informācija būtu redzama un izskatītos labi. Dažādiem vietnes izkārtojumiem un dizainiem piemēroti dažādi attēlu izmēri un formāti. Lai ģenerētais izkārtojums būtu uztverams un izskatītos labi, attēli jāapstrādā papildus.

Risinājuma ietvaros attēli tiek apgriezti, izmantojot satura un sejas atpazīšanu, lai panāktu satura attēlu iederību lapas izkārtojumā un svarīgu attēla elementu saglabāšanu. Lai attēli dabiskāk iederētos ģenerētajā lapas stilā, tiem nepieciešamības gadījumā tiek piešķirti krāsu filtri, padarot vizuāli atšķirīgus attēlus saderīgākus. Vēl viens attēlu apstrādes aspekts ir krāsu izgūšana. Lai nodrošinātu, ka bloks ar ievietoto attēlu izskatās labi, attēla pamata krāsas tiek izmantotas apkārtējos elementos, radot tieši šim attēlam unikālu stilu.

Attēlu apgriešanas funkcijas īstenošanai tiek izmantots *JavaScript*. Brīdī, kad attēls tiek ielādēts tīmekļa vietnes DOM *JavaScript* kods nosaka attēlam paredzētās vietas, izmērus, un, vadoties pēc tiem, apgriež attēlu. Oriģinālais attēls tiek aizvietots ar HTML5 *canvas* elementu, kurā tiek uzzīmēts jaunais attēls jaunajā izmērā. Risinājums nav visveiksmīgākais, bet prototipa gadījumā vairāku attēlu apgriešana var prasīt zināmu laiku. Lai tīmekļa vietnes ielāde netiktu aizkavēta attēlu apstrādes dēļ, vispirms tiek ielādēti redzami attēli un pēc tam tie tiek apstrādāti.

7.2.1. Smartcrop.js

Smartcrop.js ir *JavaScript* attēlu apgriešanas bibliotēka, kura izmanto satura atpazīšanu. *Smartcrop.js* ievieš algoritmu, kurš pēc attēla satura atrod labāko apgriešanas rezultātu. *Smartcrop.js* darbības princips ir vienkāršs un lielākoties efektīvs. *Smartcrop.js* apstrādā attēlu 7 soļos [46]:

1. Attēla šķautņu atrašana, izmantojot Laplacian šķautņu noteikšanas funkciju[44];
2. Ādas krāsai līdzīgas krāsas reģionu noteikšana;
3. Augsta piesātinājuma reģionu noteikšana;
4. Priekšrocība opcijās norādītajiem reģioniem (piemēram, atrastās sejas);

[44] http://www.owlnet.rice.edu/~elec539/Projects97/morphjrks/laplacian.html

5. Izgriežamā laukuma kandidātu ģenerēšana;
6. Izgriežamā laukuma kandidātu novērtēšana. Parametrs ir saturs, kas koncentrēts laukuma vidū un netuvojas laukuma malām.
7. Tiek izvēlēts rezultāts ar visaugstāko vērtējumu.

Smartcrop.js lietošana ir ļoti vienkārša. *Smartcrop.js* nodrošina parametru (attēla augstums, platums un svarīgākās koordinātas) padošanas funkciju. Rezultāts ir attēla x un y koordinātas, kā arī platums un augstums, pēc kuriem apgriezt attēlu [46].

7.3. att. **Attēla izgriešanas laukuma vērtēšana**

Attēlā 7.3. redzams attēls, kura apgriešanā tiek izmantots *smartcrop.js*. Sarkanais kvadrāts iezīmē iespējamo attēla izgriešanas laukumu, sānos redzami vērtējumi dažādiem izgriešanas laukumiem. Tiek izvēlētas koordinātas un izmēri laukumam ar visaugstāko vērtējumu. Attēlu var izgriezt dažādos izmēros un proporcijās, piemērā tas apgriezts proporcijā 1x1.

(a)

(b)

7.4. att. **Attēla satura analīze:**
(a) Attēlam tiek veikta satura analīze, (b) attēls pēc apgriešanas veikšanas

Attēlā 7.4. (a) ir redzama satura analīzes rezultāti. Attēlā tiek noteikti galvenie laukumi un elementi ar spilgtāko kontrastu. Analīzes mērķis ir novērtēt, vai galvenais saturs atrodas centrā un vai tas ir iespējami tālu no malām. Minētajā piemērā redzama labākā izgriešanas laukuma varianta satura analīze. Attēlā 7.4. (b) redzams gala rezultāts pēc *smartcrop.js* algoritma, kura rezultāti tika padoti attēla apgriešanas funkcijai. *Smartcrop.js* piedāvā iespēju izcelt atsevišķus attēla rajonus, kuriem vajadzētu atrasties izgrieztajā laukumā. Šāda uzbūve nodrošina izmantot sejas atpazīšanas funkciju, lai cilvēku sejām būtu priekšroka attiecībā pret citu saturu.

Risinājumā tiek izmantota *smartcrop.js* bibliotēka vienkāršā principa un izmantošanas dēļ. Prototipa gadījumā bibliotēka piedāvāja visas nepieciešamās funkcijas, līdz ar to nebija nepieciešami trešās puses risinājumi. Viens no *smartcrop.js* trūkumiem ir tā ātrdarbība, jo, testējot nelielu attēlu skaitu (2 – 5), attēli tiek apgriezti pietiekoši ātri, bet, palielinoties attēlu skaitam, tas notiek lēni un neefektīvi.

7.2.2. Sejas atpazīšana

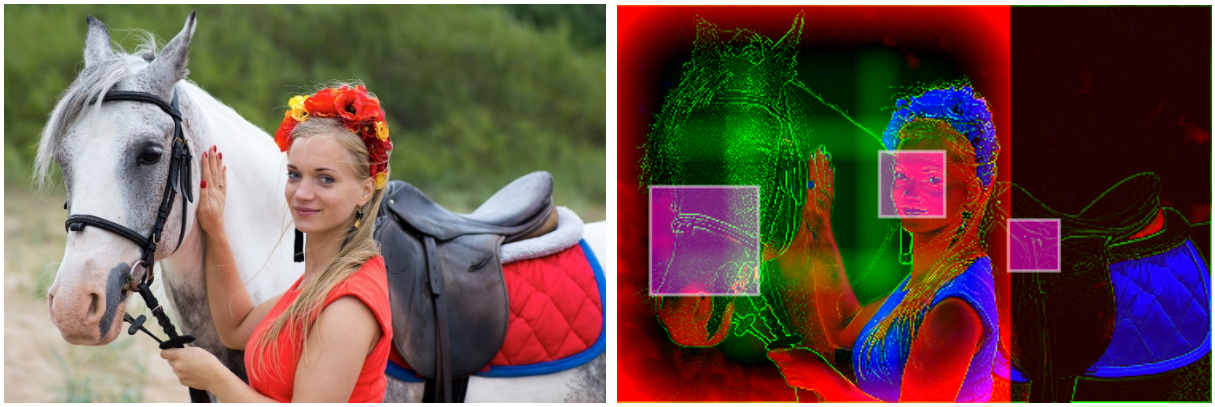
Teju jebkurā attēlā ar cilvēku ir svarīgi, lai arī pēc apgriešanas cilvēks būtu redzams. *Smartcrop.js* nodrošina attēlu apgriešanu, vadoties pēc kopējās attēla kompozīcijas, bet ne vienmēr šāda pieeja sniedz vēlamo rezultātu. Pastiprināta uzmanība ir jāpievērš attēlā redzamajām sejām un jānodrošina sejas pilnīga redzamība arī pēc attēla apgriešanas. Lai panāktu seju īpašu apstrādi, *Smartcrop.js* satura atpazīšanas funkcijai tiek izvēlēti reģioni, kuriem jāpievērš pastiprināta uzmanība, šajā gadījumā - sejas. Sejas atpazīšanai var izmantot vairākus risinājumus, prototipa ietvaros tika testētas un izmantotas divas bibliotēkas.

7.2.2.1. jQuery Face Detection.

jQuery Face Detection ir *jQuery* bibliotēkai paredzēts spraudnis, kas nodrošina seju atpazīšanu attēlos, video un citos medijos. Spraudnis nodrošina funkciju, kurai tiek padots attēls, un tā atgriež atrasto seju masīvu. Katram masīva elementam ir x un y koordinātas, augstums, garums un citi parametri. Spraudnis piedāvā asinhronu attēlu apstrādi un citas ērtas iespējas, tāpēc tas ir spēcīgs sejas atpazīšanas rīks [47].

Veicot spraudņa testēšanu ar dažādiem attēliem, rezultāti nebija pārliecinoši. Vairumā gadījumu *jQuery Face Detection* spraudnis neatrada nevienu seju attēlā, lai gan attēli bija skaidri, sejas tajos izteiktas un viegli saskatāmas. Attēlā 7.2. tiek izmantoti *Smartcrop.js* un *jQuery Face Detection*. Attēlā 7.2. (a) redzams oriģinālais attēls, kuram veic apgriešanu un sejas atpazīšanu. Attēlā 7.2. (b) redzams apgriešanas laukums un atpazītās sejas (violeti taisnstūri ar baltu apmali). Kā redzams, *jQuery* spraudnis ir atradis 3 atsevišķas sejas, kuras tiek padotas attēla apgriešanas funkcijai, taču tikai 1 no 3 atrastajām sejām patiesi ir cilvēka seja.

Minētais piemērs labi atspoguļo *jQuery Face Detection* nepilnības, bieži tiek pieļautas kļūdas. Par spīti spraudņa nepilnībām, to apvienojot ar satura atpazīšanu un *Smartcrop.js*, iegūtais rezultāts ir gana labs lietošanai.



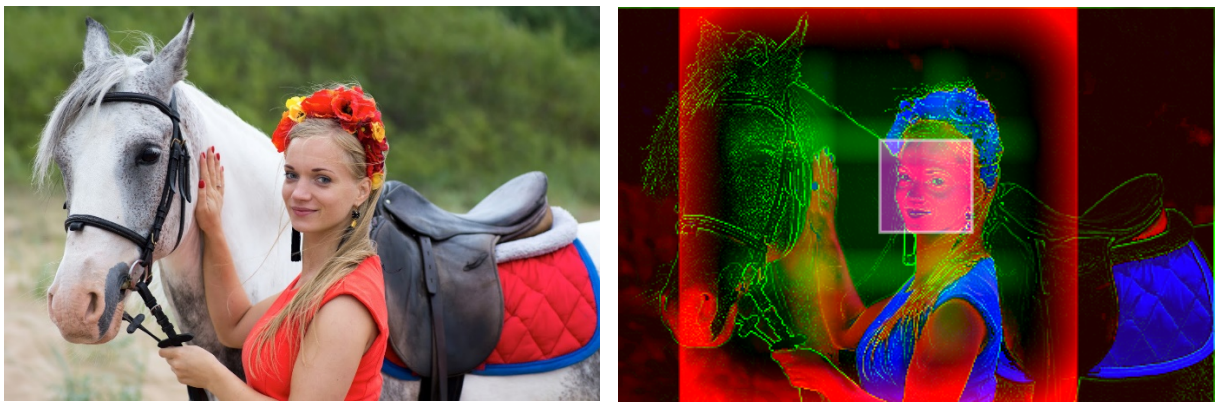
(a) (b)

7.5. att. ***jQuery* sejas atpazīšanas pielietošana attēlam:**
(a) Oriģināls attēls, (b) Attēlam pielietota sejas atpazīšana

jQuery Face Detection nav veiksmīgākais risinājums, tādēļ tika nolemts to izmantot kopā ar alternatīvu risinājumu *Tracking.js*. Integrējot abus risinājumus vienā sistēmā, tiek samazināts kļūdas faktors un neatrasto seju gadījumu skaits.

7.2.2.2. Tracking.js

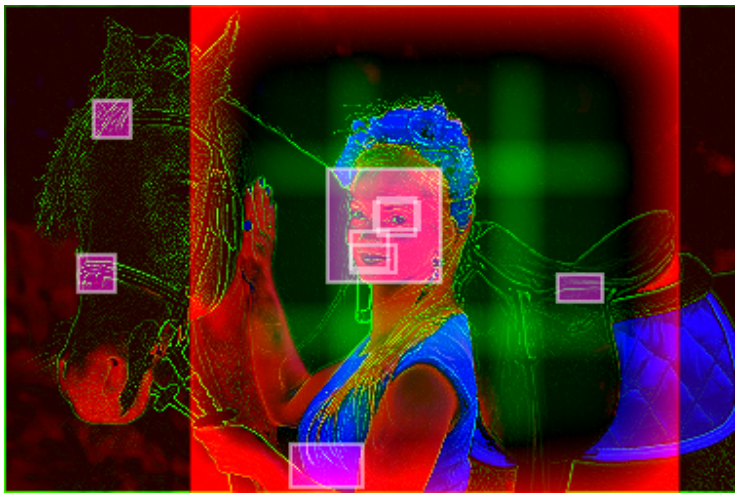
Tracking.js ir dažādu datora redzes algoritmu un metožu bibliotēka. Pateicoties HMTL5, *Tracking.js* nodrošina reāllaika sejas atpazīšanu, krāsu izsekošanu un citas attēlu atpazīšanas funkcijas. *Tracking.js* ir krietni pārāks par *jQuery Face Detection*, jo ir daudz precīzāks un tā kļūdas gadījumi ir retāki. *Tracking.js* piedāvā daudz plašāku funkciju klāstu, tāpēc ir piemērota izvēle sistēmas izveidē. *Tracking.js* ir krietni sarežģītāks par *jQuery* spraudni, bet nodrošina būtiskas funkcijas: papildus sejas atpazīšanai tas piedāvā noderīgo acu un mutes atpazīšanas funkciju. Ne vienmēr sistēma precīzi spēj noteikt seju, un tādos gadījumos noder papildu meklēšana pēc acīm vai mutes. Risinājuma kontekstā šī funkcija ir ļoti noderīga, jo pat vienas acs atrašana var būtiski ietekmēt attēla apgriešanas galarezultātu [48].



(a) (b)

7.6. att. ***Tracking.js*** **sejas atpazīšanas pielietošana attēlam:**
(a) Oriģināls attēls, (b) Attēlam pielietota sejas atpazīšana

Veicot testēšanu ar *Tracking.js*, rezultāti ir krietni labāki par *jQuery Face Detection*. Attēlā 7.6. (b) redzams, ka seja tika atrasta bez kļūdām un atrastās sejas laukums ir lielāks par *jQuery* spraudņa atrasto. *Tracking.js* atrada daudz vairāk seju nekā *jQuery*. Sejas tika atrastas biežāk un attēli, kuros tās tika atrastas, pēc struktūras ir sarežģītāki par tiem, kuros sejas spēja atpazīt *jQuery* spraudnis. Arī *Tracking.js* pieļauj kļūdas un reizēm sejas neatpazīst, bet kopējais sniegums ir augstu vērtējams.



7.7. att. ***Tracking.js*** **sejas, mutes un acu atpazīšana**

Tracking.js ir pārāks, jo spēj atrast papildus objektus. Attēlā 7.7. redzama sejas, mutes un acu atpazīšana. Arī kļūdu skaits ir ievērojami lielāks, jo tiek meklēti 3 atsevišķi objekti, bet attēlā atrastā seja acis un mute ir izteikti,, un tieši šai vietai tiks pievērsta lielāka uzmanība. Sistēmas prototipā meklēšana vispirms notiek pēc sejas, ja tā netiek atrasta, tiek izmantots acu un mutes meklētājs. Meklēt pēc visiem trīs objektiem reizē nav izdevīgi, jo, kā redzams attēlā

7.7. rodas daudz kļūdu. Taču gadījumā, ja seja netiek atrasta, ir noderīgi pieslēgt papildus meklēšanas parametrus, jo tiek samazināts sejas neatrašanas un nepilnīgas attēlu apgriešanas gadījumu skaits.

Prototipā *Tracking.js* tiek izmantots kopā ar *jQuery Face Detection*, lai iespējami samazinātu kļūdas gadījumu skaitu. Vispirms tiek meklēts ar *Tracking.js*, ja tas seju neatrod, tad izmanto *jQuery Face Detection*. Ja arī *jQuery* spraudnis neko neatrod, tad meklē pēc *Tracker.js* acu un mutes atpazīšanas.

7.3. Tīmekļa vietnes izkārtojuma ģenerēšana

Sistēmas prototipa svarīgākā daļa ir lapas izvietojuma un stila ģenerēšana. Idejas pamatā ir procedurālā ģenerēšana, kas no neliela sākuma elementu komplekta, savstarpēji kombinējot atsevišķus elementus un struktūras, spēj ģenerēt dažādus rezultātus. Šajā gadījumā lapas izkārtošanas ģenerēšanas pamatā tiek izmantota HTML moduļu bibliotēka. No atsevišķiem HTML moduļiem tiek veidota lapas struktūra un dizains. Tīmekļa vietnes izkārtojuma ģenerēšana notiek vairākos soļos, kuri aprakstīti tālāk.

7.3.1. Satura apstrāde

Pirmais solis izkārtojuma izveidē ir satura apstrāde. Lapas saturs tiek iegūts no JSON faila un padots funkcijai, kura secīgi to caurskata. Funkcijas mērķis ir izpētīt saturu to nepieciešamības gadījumā modificēt. Viena no veicamajām modifikācijām ir raksta attēlu secības maiņa, lai ik reizes izveidotu jaunu saturā ievietoto attēlu secību, nodrošinot papildus dizaina variācijas.

7.3.2. Potenciālo moduļu atlasīšana

Apstrādājot saturu, paralēli tiek atlasīti potenciālo moduļu kandidāti, balstoties uz satura parametriem. Piemēram, apstrādājot satura rakstu, tiek noteikts, ka konkrētais raksts satur 5 attēlus, līdz ar to tiek atlasīti bloki, kuri spēj izkārtojumā izmantot 5 vai mazāk attēlus. Atlasot potenciālos moduļus, tiek ņemts vērā satura tips, tā elementi, elementu skaits u.c. Darbības rezultāts ir iespējamo moduļu masīvs, kur katrs elements satur moduļa numuru, augstumu un tā elementu numurus.

Moduļu struktūra līdzinās kokam. Augstāka līmeņa modulis var ietvert citus moduļus, kuriem arī ir iespējamas variācijas, tādēļ, norādot attiecīgus moduļu numurus, tiek norādīti arī iekšējo moduļu numuri. Par piemēru var ilustrēt situāciju: modulī “section-0” var atrasties satura moduļi “content-0” un “content-1” un attēlu moduļi “image-0” un “image-1”, līdz ar to saturam, kuram der modulis “section-0” , ir vairākas iespējamās variācijas:

- “section-0” ar “content-0” un “image-0”;
- “section-0” ar “content-1” un “image-0”;
- “section-0” ar “content-0” un “image-1”;
- “section-0” ar “content-1” un “image-1”.

Minētais piemērs ir vienkāršs un sastāv tikai no 3 atsevišķiem moduļiem, kuri jau nodrošina 4 dažādas izkārtojuma iespējas. Ieviešot sistēmā vairāk moduļu, iespējamo kombināciju skaits eksponenciāli aug. Ja sekcijai ir 2 dažādi moduļi, saturam 3 dažādi moduļi un arī attēlam 3 dažādi moduļi, tad vienam rakstam tiek nodrošināti 18 unikāli moduļi, katrs ar citu izkārtojumu un attiecīgi piemērojamu stilu.

7.3.3. Konkrēta moduļa izvēle

Kad atlasīti visi potenciālie moduļi, nākamais solis ir konkrētu moduļu izvēle. Vienam rakstam var būt vairāki atšķirīgi moduļi, piemēram, rakstam ar attēlu var piešķirt moduli, kur attēls ir pa labi, pa kreisi, zem vai virs satura. Šajā solī tiek izvēlēts konkrēts modulis un piešķirti tajā ievietojamie dati. Konkrēta moduļa atlase var notikt vairākos veidos. Lai nodrošinātu izkārtojuma un dizaina daudzveidību, moduļi tiek atlasīti pēc nejaušības principa. No potenciālo moduļu masīva kāds tiek nejauši izvēlēts, šādā veidā nodrošinot dažādas moduļu kombinācijas un iespējas. Izvēlētos moduļus saglabā datu masīvā kopā ar attēlojamajiem datiem.

Moduļu atlasīšana notiek visiem lapas elementiem, tai skaitā lapas pamatam, galvenei, kājenei u.c. Lapas pamatā arī ir moduļi, kuri definē galveni, kājeni, navigāciju un citus svarīgus elementus. Katrai lapai var piešķirt jebkuru moduļu kombināciju, piemēram, galvene var atrasties lapas vidū, lapai var netikt piešķirta kājene, var būt sānu josla un citas variācijas. Katras lapas pamatā ir bāzes klase, kura atbild par skriptu ielādi, metadatiem, stila ielādi un citiem bāzes elementiem, kuri nepieciešami ģenerētās vietnes lietošanai.

7.3.4. Izkārtojuma salikšana no moduļiem

Tīmekļa vietnes izkārtojuma ģenerēšana ir salīdzinoši vienkārša. Jau ir atlasīti visi dati, tiem piekārtoti moduļi, kuros dati tiks ievietoti un kuri veidos lapas HTML kodu. Atlasīto moduļu un datu masīvu padod *Twig* dzinējam (vairāk par *Twig* 7.1.1. nodaļā), kurš nodrošina moduļu iekļaušanu un datu attēlojumu. *Twig* dzinējam tiek padoti moduļu kodi, pēc kuriem vadoties, tiek iekļauti attiecīgie moduļi.

Lapas izkārtojums sākas ar pamata moduli, kurš definē lapas struktūru un galvenos lapas blokus. Izkārtojums tiek attīstīts kokam līdzīgā struktūrā: pamata modulis, izmantojot to kodus, iekļauj citus moduļus un padod tiem datus. Nākamie moduļi ietver citus nepieciešamos

moduļus, un process turpinās, līdz ir iekļauti visi vajadzīgie moduļi un vietnes izkārtojums ir pabeigts. Moduļu iekļaušana notiek secīgi, padodot datus un iekļaujot tos HTML kodā.

```
{# NAVIGATION #}
    {% include 'partials/navigation/navigation-'~ navigation.template ~'.twig'  with {
        'navigation': navigation.data,
        'page': page
    } %}
{# HEADER #}
    {% include 'partials/header/header-'~ header.template ~'.twig'  with {
        'header': header.data
    } %}
{# CONTENT #}
    {% block content %}
        Sorry, no content
    {% endblock %}
{# FOOTER #}
    {% include 'partials/footer/footer-'~ footer.template ~'.twig' with {
        'footer': footer.data,
        'social': social
    } %}
{# SCRIPTS #}
```

7.8. att. ***Twig*** **moduļu iekļaušana**

Attēlā 7.8. redzams par moduļu iekļaušanu un datu padošanu atbildīgs kods. Mainīgie "*navigation,template*", "*header.template*" un "*footer.template*" satur moduļu numurus, kuri apzīmē iekļaujamos moduļus. Vēl redzams, ka moduļa iekļaušana notiek, tam padodot datus, kuri pēc tam tiek izmantoti citu moduļu iekļaušanai un to attēlošanai.

7.4. Sistēmas dati

Sistēmas prototipā tiek izmantots viens JSON fails, kas satur visus datus par tīmekļa vietni, kā arī datne ar attēliem. Koncepta gadījumā ir svarīgi izveidot viegli modificējamu, pārskatāmu un vienkārši manipulējamu datu sistēmu. JSON datu paraugs pievienots 1. pielikumā. Dati sastāv no tīmekļa vietnes pamata atribūtiem, galvenes, navigācijas, rakstiem, kājenes un sociālo tīklu informācijas. Raksts ir sadalīts atsevišķās sekcijās, katra no tām var saturēt tikai attēlu, tekstu vai citu iespējamo atribūtu.

Sistēmas dati ir definēti brīvā formā, un to organizēšana tika izveidota tieši prototipam, lai nodrošinātu tīmekļa vietnes pamata informāciju, pēc kuras vadoties izstrādāt sistēmu. Datus ir iespējams pārveidot citās struktūrās, piemēram, izmantojot sistēmu kopā ar *WordPress* REST API to var pielāgot API datu struktūrai, tādā veidā nodrošinot ērtu un vienkāršu integrāciju trešās puses aizmugursistēmās.

7.5. Satura analīze

Saturs tiek analizēts, secīgi caurskatot katra elementa dažādus aspektus. No satura teksta tiek izgūts vārdu skaits, atslēgas vārdi un veikta vārdu klasifikācija. Teksta analīzei tiek izmantots PHP *Text Analysis*[45] un *Stanford Named Entity Recognizer* (NER)[46]. PHP *Text analysis* izgūst vārdu skaitu un atslēgas vārdus, bet NER vārdus klasificē. Analizējot attēlus, tiek noteikts attēlu skaits, izmēri, pamata krāsas un redzamie objekti. Attēlu analīzei tiek izmantots *Google Cloud Vision* API[47], kurš nodrošina plašas attēla analizēšanas iespējas un funkcijas.

Satura analīzes mērķis ir uzlabot tam atbilstoša dizaina ģenerēšanu. Vadoties pēc elementu skaita, atslēgas vārdiem, attēlu izmēriem un citiem faktoriem, tiek atlasīti piemērotākie moduļi, no kuriem tiek ģenerēts dizains un izkārtojums. Satura analīze nenodrošina ģenerētā dizaina pilnīgu atbilstību saturam, bet tā noteikti uzlabo iznākumu. Satura analīze ir tikai viena kopējās sistēmas daļa. Vēl ir jāizveido moduļi, kuri ir pielietojami dažādam saturam, kā arī nejaušības princips konkrēta moduļa izvēlē.

7.5.1. PHP Text Analysis

PHP *Text Analysis* ir bibliotēka, kura veic informācijas izguvi (*Information Retrieval*) un dabiskās valodas apstrādi (*Natural Language Processing*), izmantojot PHP valodu. Bibliotēka nodrošina plašu funkciju klāstu, bet prototipa ietvaros tiek izmantotas divas sistēmas funkcijas. Teksta atslēgas vārdu izgūšanai tiek lietots *Rapid Automatic Keyword Extraction* (RAKE) un teksta sadalītājs (*tokenizer*), ar kura palīdzību tiek skaitīti teksta vārdi.

RAKE ir algoritms, kurš izgūst atslēgas vārdus no dokumenta. Atslēgas vārdi ir viena vai vairāku vārdu secība, kura precīzi atspoguļo saturu. RAKE ir plaši izmantots dažādos veidos. Algoritma rezultātu un precizitāti ietekmē tādi faktori kā, piemēram, rakstības stils, tēma un atslēgas vārdu mērķis [50].

Teksta sadalītājs sadala tekstu daļās, ar kurām var veikt tālākas darbības. Tekstu var sadalīt vārdos, teikumos, zilbēs vai citās apstrādājamās daļās. Prototipā teksts tiek dalīts vārdos un iegūts kopējais vārdu skaits. Iespējams izskaitīt arī unikālus vārdus, palīgvārdus un citus teksta elementus.

[45] https://github.com/yooper/php-text-analysis
[46] https://nlp.stanford.edu/software/CRF-NER.shtml
[47] https://cloud.google.com/vision/

7.5.2. Stanford Named Entity Recognizer (NER)

Stanford NER ir nosaukto vienību atpazinēja (*Named Entity Recognizer*) implementācija. NER veic vārdu secības klasifikāciju, ja tajā ir nosaukumi, piemēram, personvārdi, uzņēmumu nosaukumi, gēnu vai proteīnu nosaukumi. *Stanford* NER piedāvā vairākus modeļus, pēc kuriem vadīties, meklējot vārdu secību klases. Prototipa gadījumā tiek izmantots modelis ar 7 klasēm [49]:

- Atrašanās vieta (*Location*);
- Persona (*Person*);
- Organizācija (*Organization*);
- Valūta (*Money*);
- Procenti (*Percent*);
- Datums (*Date*);
- Laiks (*Time*);

Stanford NER ir noderīgs, jo nenosaka klasi visiem vārdiem, bet konkrētām vārdu secībām. Ja tekstā ir neiederīgi vārdi, tad NER tos neklasificē. Klasificējamiem vārdiem jāveido izprotama, korekta secība. *Stanford* NER ir balstīts uz *Java* programmēšanas valodu, bet tam tiek piedāvātas saskarnes dažādās programmēšanas valodās [49].

Prototipa gadījumā tiek izmantota *Stanford* NER PHP saskarne PHP-*Stanford*-NLP[48], kas veic NER marķēšanu. PHP-*Stanford*-NLP kā ieejas datus saņem teksta virkni un kā rezultātu izdod masīvu, kura elements ir vārds, kam atzīmēta kāda no 7 klasēm. Gadījumos, kur vārds neatbilst nevienai klasei, tiek piešķirta vērtība "0". Pēc NER darbības tiek caurskatīti visi masīva elementi un atlasīti unikālie klašu ieraksti. No teksta izgūtās klases tiek izmantotas vienai vai vairākām klasēm atbilstošu moduļu atlasei. Ja neviens modulis neatbilst konkrētajām klasēm, tad tiek izmantoti visi pieejamie moduļi.

```
E:\Dropbox\domains\ADI\base-classes\base-page.php:72:string '
        John Doe was a theoretical physicist born in March 14, 1879.
        He was born in Germany and he is a shareholder of Google with 19% of shares.
        John had 9$ in his pocket when he arrived at 9:30' (length=215)

E:\Dropbox\domains\ADI\page-models\front.php:95:
array (size=7)
  0 => string 'PERSON' (length=6)
  1 => string 'DATE' (length=4)
  2 => string 'LOCATION' (length=8)
  3 => string 'ORGANIZATION' (length=12)
  4 => string 'PERCENT' (length=7)
  5 => string 'MONEY' (length=5)
  6 => string 'TIME' (length=4)
```

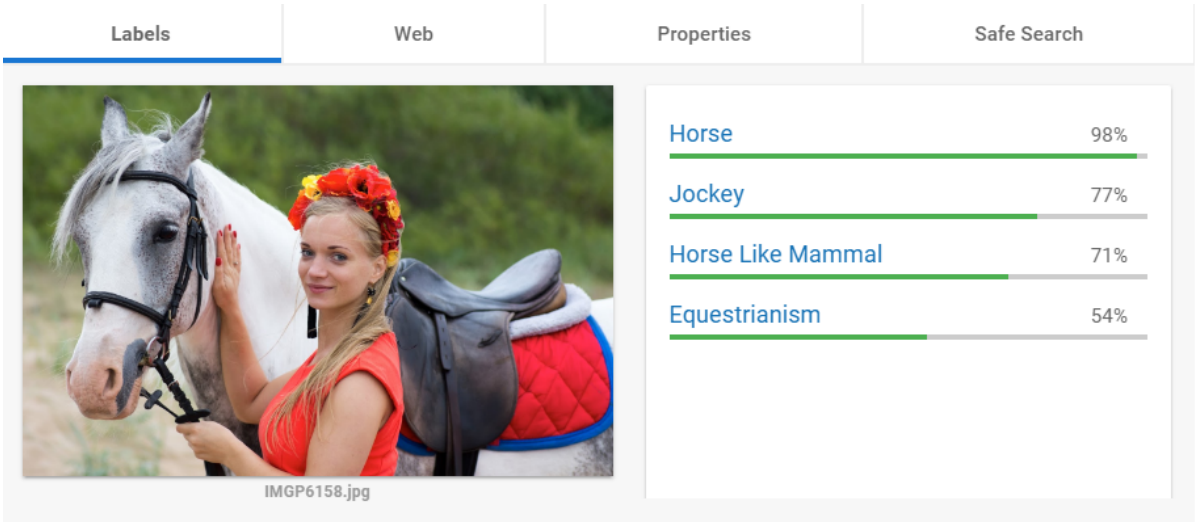
7.9. att. **NER teksta klasificēšana**

[48] https://github.com/agentile/PHP-Stanford-NLP

Attēlā 7.9. redzams neliels piemērs, kur PHP-*Stanford*-NLP tiek padots teksts ar dažādiem teksta elementiem. Tekstā ir piemēri visām 7 klasēm, rezultāts ir masīvs, kurš satur visas atrastās klases.

7.5.3. Google Cloud Vision API

Google Cloud Vision API[49] ir rīks, kurš izmanto spēcīgus mašīnmācīšanās modeļus, piedāvājot REST API. *Vision* API spēj ātri klasificēt attēlus dažādās kategorijās, noteikt attēlā atrodamus individuālus objektus, sejas un drukātus vārdus. *Vision* API plašu klāstu attēlos bieži redzamu objektu. Laikam ritot, *Vision* API tiek papildināts ar jauniem konceptiem un atpazīšanas precizitāte tiek uzlabota. *Vision* API izmanto *Google Image Search*[50], lai meklētu, piemēram, slavenības, uzņēmumu logo, ziņas vai pasākumus. Vēl viena *Vision* API iespēja ir teksta izgūšana no attēliem: ja attēlā ir redzams teksts, tad *Vision* API izmanto optisko simbolu atpazīšanu (OCR), lai atpazītu tekstu un valodu. [51].



7.10. att. **Vision API attēla klasificēšana [51]**

Prototipa gadījumā no attēla tiek izgūti redzamie objekti (7.10. attēls), pēc kuriem tiek koriģēta atbilstošu moduļu izvēle. No attēla izgūst arī atslēgas vārdus un tā pamatkrāsas, kuras izmantot moduļu stila izveidē. *Vision* API piedāvā noderīgu funkciju klāstu, bez prototipā izmantotajiem atribūtiem iespējams izmantot cilvēku seju un pausto emociju, tekstu, uzņēmumu logo, orientieru un citu elementu meklēšanu. Risinājuma kontekstā šīs iespējas netiek plašāk apskatītas, bet, attīstot sistēmu, tās var būtiski uzlabot ģenerēto stilu.

Prototips izmanto *GoogleCloudVision*PHP[51] bibliotēku *Google Vision* API īstenošanai. Bibliotēka piedāvā ērtas funkcijas, lai izpildītu *Google Vision* API pieprasījumus. Satura

[49] https://cloud.google.com/vision/
[50] https://images.google.com/
[51] https://github.com/thangman22/google-cloud-vision-php

analīzes procesā prototips analizē katru attēlu ar *Vision* API, iegūstot JSON failu ar rezultātiem. Rezultāti tiek apkopoti, un, pēc tiem vadoties, tiek pieņemti dizaina lēmumi.

7.6. Stila ģenerēšana

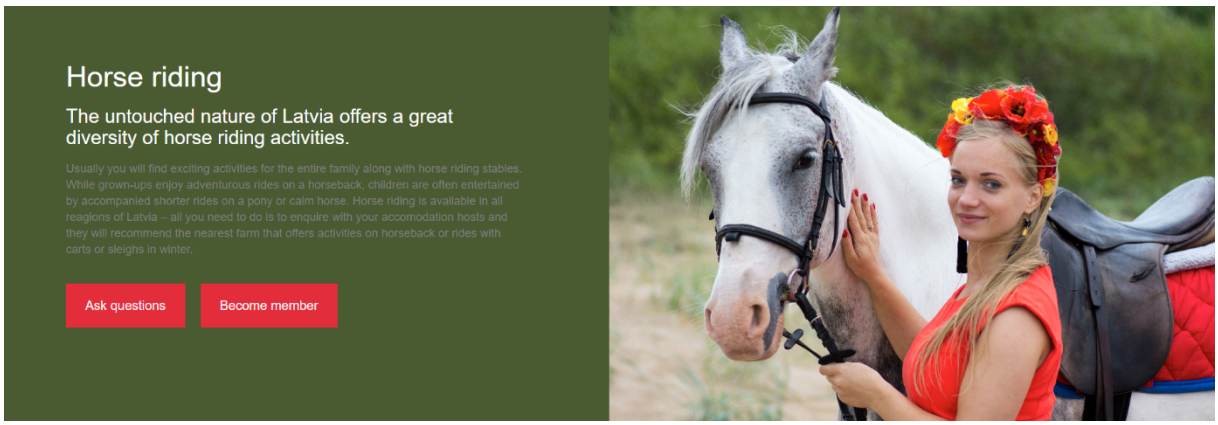
Tīmekļa vietnes stils tiek ģenerēts no vairākiem atsevišķiem elementiem. Vietnes pamata stils definē galveno, normalizē elementus un ir pamats tālākai stila izveidei. Tīmekļa vietnes stilam tiek pievienots *Bootstrap* stils (7.1.2.), kurš nodrošina režģa sistēmu un citu *Bootstrap* elementu stilu. Katra moduļa pamata stils apraksta moduļa izkārtojumu, savstarpējās elementu attiecības un izmērus. Modulis un tā stils ir veidoti neatkarīgi un pašpietiekami, moduli var ievietot dažādās lapas vietās un tam tiks saglabāts sākotnējais stils.

Lai ģenerētais stils būtu pēc iespējas dažāds, vairākkārt ģenerējot stilu, viens un tas pats bloks izskatītos atšķirīgi un stils atbilstu saturam, vairākiem mainīgajiem tiek piešķirtas dažādas vērtības, kuras ietekmē galarezultātu. Prototipa izveidē tiek definēti šādi elementi, kuru variācijas būtiski ietekmē iznākumu:

- Krāsa;
- Fonts;
- Apmales;
- Mala un polsterējums;
- Izmēri.

Elementi var tikt definēti katram blokam atsevišķi, piemēram, krāsa, vai visai lapai kopā, piemēram, fonts. Sistēmā definētais pamata stils tiek lietots tad, ja kāds no elementiem netiek mainīts. Ja, piemēram, malas un polsterējumi nemainās, tad izmanto iepriekš definētas vērtības. Krāsu palete tiek izgūta no satura attēliem (7.5.3.), ja saturā nav attēlu, tad tiek izmantotas iepriekš definētās krāsas.

Krāsas tiek definētas augšējā līmeņa blokos. Veidojot izkārtojumu un stilu tīmekļa vietnes rakstam, krāsa tiek definēta raksta modulim un visiem tā apakšmoduļiem. Vienam modulim tiek piešķirtas vairākas krāsas, kuras tiek izmantotas dažādu elementu stiliem. Pamata krāsa var tikt izmantota elementa fonam, akcentu krāsā var būt pogas, apmales vai citi elementi. Fontus, apmales, malas un polsterējumus definē visai lapai un pēc nepieciešamības katram blokam atsevišķi. Šo elementu izveides pamatā ir nejaušības princips, kur tiek izvēlētas vērtības noteiktos intervālos, vai iepriekš definētas vērtības, piemēram, fonti izvēlēti no fontu bibliotēkas. Izmēru definēšana.



7.11. att. **Stila elementu pielietojums**

Stila elementu vērtības tiek definētas ar PHP un ar Twig palīdzību tiek padotas konkrētajiem moduļiem. Veicot satura analīzi un potenciālo moduļu atlasīšanu tiek definētas stila elementu vērtības, kuras tiek padotas attiecīgajiem moduļiem, kuri tās izmantot elementu īpašību aprakstīšanai. Attēlā 7.11. redzams ģenerētais modulis ar tā stilu. Fona un pogu krāsas tika izgūtas no blakus esošā attēla, malas, polsterējums un fonts tika ģenerēti no sistēmā definētajām vērtībām.

7.7. Prototipa izstrādes gaitā gūtie secinājumi un rezultāti

Prototipa izstrāde bija veiksmīga, un tās gaitā tika gūtas noderīgas atziņas, papildinot autora zināšanas par šāda veida sistēmu veidošanu ar ikdienā izmantojamām idejām un tehnoloģijām. Darba gaitā autors saskārās ar vairākiem izaicinājumiem, kuru atrisināšana sniedza vērtīgu pieredzi. Tika izdarīti šādi secinājumi:

1. Prototipa veiksmīgā izstrāde nozīmē, ka piedāvātais risinājums ir īstenojams un sākotnējā ideja bijusi praktiska.
2. Attēlu apstrāde un analīze ir sistēmai svarīga, bet tā sagādā zināmas grūtības un izaicinājumus. Process ir sarežģīts un rezultāti ir dažādi. Sejas ne vienmēr tiek atpazītas veiksmīgi, turklāt dažādi rīki piedāvā dažādus rezultātus. Prototipā izmantotais *jQuery Face Detection* bija neprecīzs un bieži nespēja attēlā atpazīt sejas. *Tracking.js* sniedza daudz labākus rezultātus, sejas tika atrastas biežāk un precīzāk.
3. Attēlu apstrāde ir lēna. Prototipa gadījumā attēli tiek apstrādāti katru reizi, kad lapa tiek ielādēta, tāpēc process ir lēns. Problēmas risinājums būtu apstrādāt katru attēlu tikai vienreiz un saglabāt tā rezultātus datubāzē. Arī apgriešanas rezultāti būtu jāsaglabā un atkārtoti jāizmanto gadījumos, kad attēls jau iepriekš ticis apgriezts nepieciešamā izmērā.

4. Dizaina dažādība un izkārtojuma variācijas ir atkarīgas no iespējamā izveidojamu un uzturamu modeļu skaita. Jo vairāk moduļu, jo vairāk kombinācijās tos var izvietot, rezultātā iegūstot jaunus vietnes izkārtojumus.
5. Sistēmas dizaina dažādība var būt nepārliecinoša. Prototipa izstrādē netika veidota liela moduļu bibliotēka, tāpēc prototipa ģenerētās tīmekļa vietnes izskatās līdzīgas. Šo problēmu var labot, ieviešot arvien vairāk elementu, kurus izmantot vietnes dizaina izveidē.
6. Moduļu izvēle tikai pēc nejaušības principa nav efektīva, jo var rasties situācijas ar vairākiem vienādiem blokiem pēc kārtas. Lai risinātu problēmu, tiek analizēts saturs ar mērķi uzlabot ģenerēšanas rezultātus.
7. Satura analīze ir lēns process, un tās rezultāti var būt neviennozīmīgi. Satura analīzes rezultātu ietekmē dažādi faktori, piemēram, rakstības valoda un analīzes mērķis.
8. Lai nodrošinātu saturam atbilstošu dizainu, jāizstrādā daudzi dažādiem satura tipiem piemēroti moduļi. Satura analīze uzrāda daudz atšķirīgu rezultātu. Lai tos varētu pēc iespējas vairāk izmantot, ir jāizstrādā satura dažādībai atbilstoši moduļi.
9. *Google Vision* API attēlu analīze ir ļoti efektīva. Prototipa izstrādē tika izmantota tikai neliela daļa rīka piedāvāto iespēju, pārējo funkciju izpēte un lietojums varētu būtiski ietekmēt iznākumu.

Prototipa izstrādes mērķis ir sasniegts. Tika pierādīta risinājuma atbilstība problēmai un tā iespējas problēmas atrisināšanā. Izstrādes gaitā tika novērotas gan risinājuma priekšrocības, gan nepilnības, kuras izlabojot risinājums tiktu pilnveidots un to varētu izmantot īsta produkta izstrādei.

REZULTĀTI

Darba ietvaros tika veikts pētījums ar mērķi izveidot tīmekļa vietnes stila ģenerēšanas sistēmas konceptu. Pētījuma teorētiskajā daļā tika veikta literatūras izpēte [3. nodaļa], lai apzinātu tehnoloģijas, rīkus un pieejas risinājuma izveidei. Literatūras izpētē tika apskatīti tādi jēdzieni kā procedurālā ģenerēšana un uz ierobežojumiem balstītas stila lapas. Tika izpētīti esošie risinājumi [4. nodaļa], precīzāk, "*The Grid*", "*Wix*" un "*RightClick.io*". Literatūras apskata un esošo risinājumu izpētes gaitā tika iegūtas idejas, rīki un pieejas risinājuma izstrādei.

Darba praktiskās daļas galvenais rezultāts ir izveidotais tīmekļa vietnes stila ģenerēšanas risinājums [5. nodaļa]. Risinājums ļauj ģenerēt tīmekļa vietnes izkārtojumu un dizainu, par pamatu izmantojot elementu bibliotēku un satura analīzi. Izmantojot elementu savstarpēju kombinēšanu un atribūtu izgūšanu no satura, tiek ģenerēts unikāls vietnes stils.

Risinājuma izstrādes gaitā tika apskatīti vairākas būtiskas sistēmas iezīmes un rīki [6. nodaļa], kas nodrošina tīmekļa vietnes stila ģenerēšanu. Tika apskatīti šādi jēdzieni:

- Tīmekļa vietnes izkārtojuma ģenerēšana, izmantojot moduļu bibliotēku;
- Tīmekļa vietnes stila ģenerēšana;
- Sistēmas modulāra uzbūve;
- Satura analīze;
- Attēlu apgriešana;
- Sejas atpazīšana.

Lai izvērtētu risinājuma praktiskās pielietošanas iespējas, tika izveidots tehniskais prototips [7. nodaļa], kurš īsteno minētās funkcijas. Prototips sastāv no vairākām atsevišķām komponentēm, kuras apvienojot, tiek ģenerēts tīmekļa vietnes stils. Tika izmēģināti vairāki rīki un pieejas, līdz sasniegts vēlamais rezultāts. Prototipa izstrāde tiek uzskatīta par veiksmīgu, un secināts, ka piedāvātais risinājums darbojas un ir īstenojams reālos dzīves apstākļos.

Pētījuma gaitā tika rastas atbildes uz izvirzītajiem pētījuma jautājumiem un sasniegti izvirzītie mērķi:

- Veikta literatūras izpēte, iegūtas padziļinātas zināšanas par tīmekļa vietnes dizaina ģenerēšanu un noskaidrotas pamata tehnoloģijas un pieejamie rīki.
- Veikta esošo risinājumu izpēte un izvērtējums, apzinātas šo risinājumu nepilnības.
- Ierosināts risinājums tīmekļa vietnes stila ģenerēšanai.
- Aprakstītas risinājuma komponentes un kopējais risinājums.
- Izveidots risinājuma tehniskais prototips.
- Darba gaitā gūtas vairākas atziņas un izdarīti būtiski secinājumi par tīmekļa vietnes stila ģenerēšanu un ar to saistīto rīku izmantošanu.

Pētījuma ietvaros tika īstenota sākotnējā ideja un atrisināta izvirzītā problēma. Pētījuma sākumā netika apzināti visi potenciālie izaicinājumi, ar kuriem nācās saskarties risinājuma izveidē, tāpēc darba ietvaros tika apskatīti papildus rīki un koncepti. Pētījuma gaitā tika izpētīts ne vien autora risinājums, bet arī trešo pušu rīki un to integrēšanas iespējas vienotā sistēmā, kas bija vērtīgi pētījumam.

SECINĀJUMI

Darba mērķis ir atrisināt problēmu standarta tīmekļa vietņu izstrādes procesā: tas ir dārgs, nepieciešams daudz resursu un specifisku zināšanu. Darbā ierosinātā tīmekļa vietnes stila ģenerēšanas sistēma ir īstenojama un spēj nodrošināt problēmas risināšanai nepieciešamās darbības. Tiek piedāvāta sistēma, kura īsā laikā spēj ģenerēt tīmekļa vietnes dizainu. Process neprasa papildu resursus un specifiskas zināšanas, sistēmā tiek ievadīti dati un, balstoties uz tiem, tiek ģenerēts tīmekļa vietnes stils.

Pētījuma gaitā tika noskaidroti dažādi rīki un tehnoloģijas, ar kuru palīdzību iespējams īstenot sistēmas funkcijas. Sniegts ieskats minētās sistēmas īstenošanā un izstrādē. Darba gaitā tika secināts, ka nav definētu vadlīniju un pieejas šādas sistēmas izveidei. Pētītos rīkus var lietot un modificēt tīmekļa vietnes stila ģenerēšanas sistēmas veidošanai, bet īpaši šādu un līdzīgu sistēmu izveidei paredzēti rīki un tehnoloģijas pagaidām nav izstrādāti.

Darba gaitā tika apskatīti vairāki esošie risinājumi, kuri dažādos veidos ģenerē tīmekļa vietnes. Nav zināmi apskatīto rīku darbības principi un rezultāta atbilstība izvirzītajiem sistēmas kritērijiem. Ir atrodami sistēmas atsevišķu sastāvdaļu, rīku un tehnoloģiju apraksti, bet nav plašākas informācijas par tās kopējo uzbūvi un pamatprincipiem.

Izstrādājot risinājuma prototipu, tika secināts, ka ģenerētā dizaina daudzveidība un atbilstība saturam ir atkarīga no izstrādājamo un uzturamo elementu skaita un dažādības. Ieviešot arvien lielāku skaitu dažādu elementu, iespējams iegūt arvien atšķirīgākas un sarežģītākas izkārtojuma un stila kombinācijas. Prototipa gadījumā izstrādātais elementu skaits bija neliels, līdz ar to ģenerētie tīmekļa vietņu stili pēc uzbūves bija līdzīgi.

Svarīga sistēmas sastāvdaļa ir satura analīze un tās atgrieztie dati. Tika novērots, ka satura analīze, pieaugot attēlu un tekstu skaitam, kļūst arvien lēnāka. Satura analīzes rezultāti var būt neprecīzi un nepilnīgi. Veicot sejas atpazīšanu, tika novērots, ka atrasto seju precizitāti ietekmē dažādi faktori, piemēram, izmantotie rīki, attēla kvalitāte un tajā redzamais. Teksta analīzē tika novērots, ka rezultāts var būt tekstam neatbilstošs, jo to ietekmē dažādi faktori, piemēram, rakstības stils, valoda, teksta saturs. Lai nodrošinātu dažādiem satura veidiem, formātiem un citām niansēm atbilstošu tīmekļa vietnes izkārtojumu, jāizveido elementu bibliotēka, kura satur daudz un dažādus elementus.

Sākotnējās pētījuma sfēras un kopējais pētījuma apjoms izstrādes gaitā ir palielinājies. No vienas puses, ir gūtas papildus zināšanas, no otras, šaurāka pētījuma sfēra un konkrētu lietu padziļināta izpēte, iespējams, sniegtu jaunus rezultātus un atziņas. Pētījuma sākumā netika apzinātas visas problēmas, ar kurām nāktos saskarties, un visas risinājuma izveidei nepieciešamās sastāvdaļas.

Darba ietvaros tika atbildēts uz pētījuma jautājumiem un sasniegti visi pētījumā izvirzītie mērķi. Izveidotais risinājums atbilst visiem nosacījumiem, un ar prototipa izveidi tika pierādīts, ka šādu sistēmu iespējams īstenot reālos apstākļos. Izveidotais risinājums un tā prototips ir apjomīgs, tas izmanto daudzus trešo pušu risinājumus un nodrošina pamata funkcijas. Lai darbojošos prototipu varētu uzskatīt par pabeigtu produktu, jāiegulda vēl daudz darba un izpētes. Pētījumā tika apskatītas sistēmas pamata funkcijas, tomēr, lai tas būtu piedāvājams sistēmas lietotājiem, ir jāatrisina vairākas vēl neapskatītas problēmas un jānodrošina papildu darbības.

IZMANTOTĀ LITERATŪRA

[1] Procedural generation. [tiešsaiste] – [skatīts: 14.12.2016.]. Pieejams: https://en.wikipedia.org/wiki/Procedural_generation

[2] Cassowary. [tiešsaiste] – [skatīts: 15.12.2016.]. Pieejams: https://en.wikipedia.org/wiki/Cassowary_(software)

[3] Greg J. Badros , Kim Marriott, Alan Borning and Peter Stuckey "Constraint Cascading Style Sheets for the Web" 1999. [tiešsaiste]. Pieejams: http://constraints.cs.washington.edu/web/ccss-uwtr.pdf

[4] AI Design: Robot Apocalypse Warning Number 1 [tiešsaiste] – [skatīts: 24.12.2016.]. Pieejams: https://www.completewebresources.com/the-grid-ai-website-review/

[5] The Grid Help Center [tiešsaiste] – [skatīts: 24.12.2016.]. Pieejams: http://help.thegrid.io/

[6] Is The Grid a better web designer than you? [tiešsaiste] – [skatīts: 24.12.2016.]. Pieejams: http://www.webdesignerdepot.com/2014/12/is-the-grid-a-better-web-designer-than-you/

[7] Grid Style Sheets 2.0 [tiešsaiste] – [skatīts: 08.01.2017.]. Pieejams: http://gridstylesheets.org/

[8] Has Website AI Arrived? [tiešsaiste] – [skatīts: 12.01.2017.]. Pieejams: http://chrislema.com/website-ai-arrived/

[9] wix.com [tiešsaiste] – [skatīts: 15.01.2017.]. Pieejams: http://www.wix.com/

[10] Now You Can Use Artificial Intelligence to Build Your Website [tiešsaiste] – [skatīts: 15.01.2017.]. Pieejams: http://www.cmswire.com/web-cms/now-you-can-use-artificial-intelligence-to-build-your-website/

[11] The Future of Website Creation: Introducing Wix ADI [tiešsaiste] – [skatīts: 15.01.2017.]. Pieejams: http://www.wix.com/blog/2016/06/wix-artificial-design-intelligence/

[12] Wix Unveils New Innovation: Websites That Design Themselves Using AI [tiešsaiste] – [skatīts: 15.01.2017.]. Pieejams: https://smallbiztrends.com/2016/06/wix-artificial-design-intelligence.html

[13] Has Wix Killed Web Design? [tiešsaiste] – [skatīts: 15.01.2017.]. Pieejams: https://uxmag.com/articles/has-wix-killed-web-design

[14] PageRank [tiešsaiste] – [skatīts: 15.01.2017.]. Pieejams: https://en.wikipedia.org/wiki/PageRank

[15] Procedural Generation As The Future [tiešsaiste] – [skatīts: 17.01.2017.]. Pieejams: http://wccftech.com/article/procedural-generation-future

[16] Constraint CSS [tiešsaiste] – [skatīts: 18.01.2017.]. Pieejams: http://gridstylesheets.org/guides/ccss/

[17] Alan Borning, Bjorn Freeman-Benson, and Molly Wilson “Constraint Hierarchies” 1992. [tiešsaiste]. Pieejams: ftp://ftp.cs.washington.edu/pub/constraints/papers/constraint-hierarchies-lisp-symb-comp.pdf
[18] Greg J. Badros and Alan Borning “The Cassowary Linear Arithmetic Constraint Solving Algorithm: Interface and Implementation” 1998. [tiešsaiste]. Pieejams: http://constraints.cs.washington.edu/cassowary/cassowary-tr.pdf
[19] Cassowary JS [tiešsaiste] – [skatīts: 19.01.2017.]. Pieejams: https://github.com/slightlyoff/cassowary.js
[20] GSS engine [tiešsaiste] – [skatīts: 21.01.2017.]. Pieejams: https://github.com/gss/engine
[21] Wix.com Review [tiešsaiste] – [skatīts: 21.01.2017.]. Pieejams: https://superbwebsitebuilders.com/wix-review/
[22] 13 Key Things To Know Before You Use Wix.com [tiešsaiste] – [skatīts: 21.01.2017.]. Pieejams: http://www.websitebuilderexpert.com/wix-review
[23] The Grid Developer documentation [tiešsaiste] – [skatīts: 21.01.2017.]. Pieejams: http://developer.thegrid.io/#header-libraries
[24] Welcome to RightClick.io! [tiešsaiste] – [skatīts: 23.01.2017.]. Pieejams: http://blog.rightclick.io/
[25] 5 Useful Chatbots for Small Business Owners [tiešsaiste] – [skatīts: 23.01.2017.]. Pieejams: http://www.lifehack.org/525040/5-useful-chatbots-for-small-business-owners
[26] No Man's Sky Versus the Actual Universe [tiešsaiste] – [skatīts: 24.01.2017.]. Pieejams: http://www.kotaku.co.uk/2016/04/20/no-mans-sky-versus-the-actual-universe
[27] The Grid’s Eyes [tiešsaiste] – [skatīts: 29.03.2017.]. Pieejams: http://automata.cc/discovering-salient-regions/
[28] Redesigns: Learning with You [tiešsaiste] – [skatīts: 29.03.2017.]. Pieejams: https://blog.thegrid.io/redesigns-learning-with-you/
[29] The Composer [tiešsaiste] – [skatīts: 29.03.2017.]. Pieejams: http://design-systems.github.io/basics/
[30] HTML Layouts [tiešsaiste] – [skatīts: 16.04.2017.]. Pieejams: https://www.w3schools.com/html/html_layout.asp
[31] CSS Introduction [tiešsaiste] – [skatīts: 16.04.2017.]. Pieejams: https://www.w3schools.com/css/css_intro.asp
[32] HTML & CSS [tiešsaiste] – [skatīts: 16.04.2017.]. Pieejams: https://www.w3.org/standards/webdesign/htmlcss
[33] Understanding CSS Grid Systems from the Ground Up [tiešsaiste] – [skatīts: 16.04.2017.]. Pieejams: https://www.sitepoint.com/understanding-css-grid-systems/

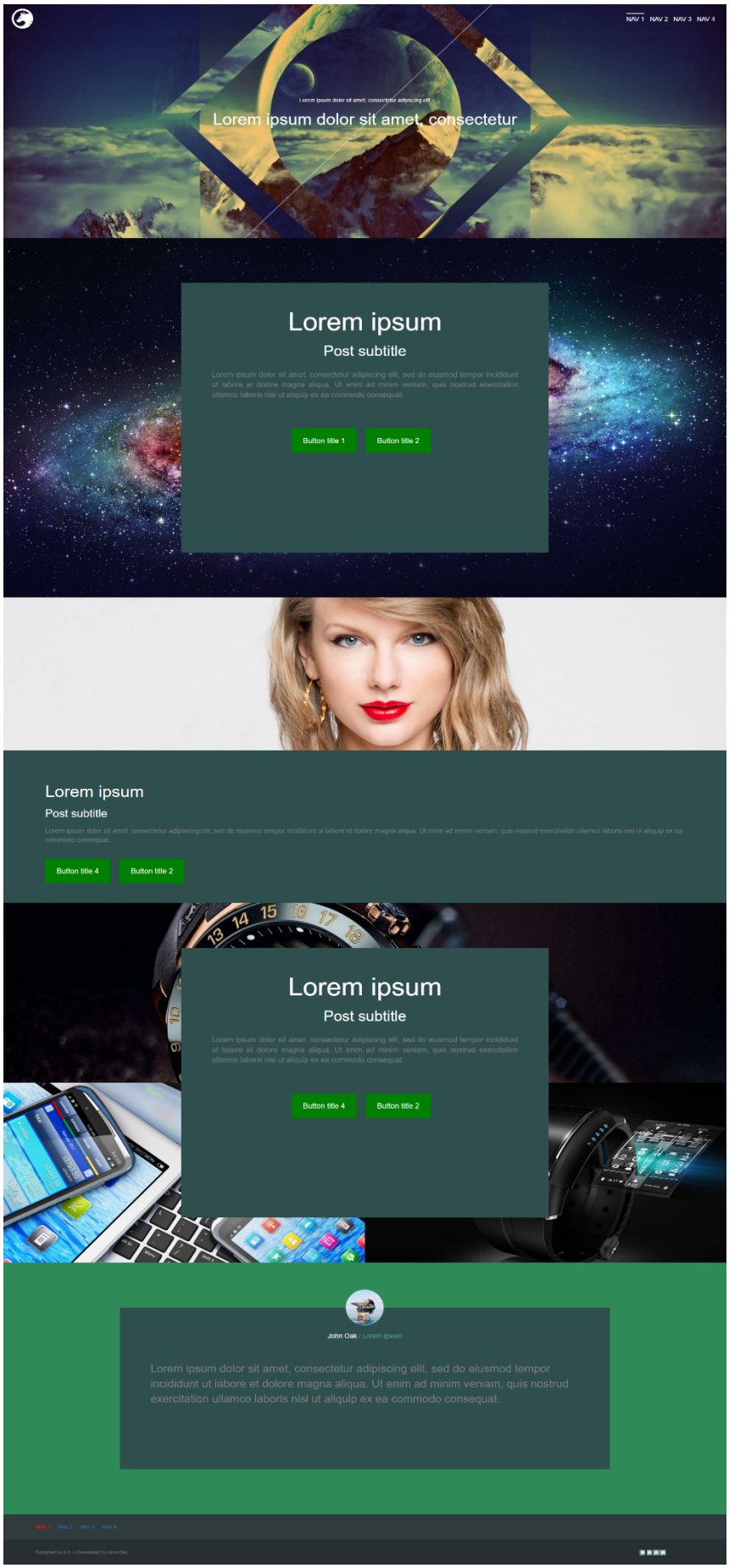
[34] Modularity in Websites [tiešsaiste] – [skatīts: 18.04.2017.]. Pieejams: https://cs.wellesley.edu/~cs110/reading/modularity.html
[35] Top 5 php template engines [tiešsaiste] – [skatīts: 18.04.2017.]. Pieejams: http://www.sitecrafting.com/blog/top-5-php-template-engines/
[36] How to Write a Project Scope Statement [tiešsaiste] – [skatīts: 30.04.2017.]. Pieejams: http://www.projectengineer.net/how-to-write-a-project-scope-statement/
[37] Introducing JSON [tiešsaiste] – [skatīts: 08.05.2017.]. Pieejams: http://www.json.org/
[38] JSON-LD is an official Web Standard [tiešsaiste] – [skatīts: 08.05.2017.]. Pieejams: http://www.dataversity.net/j-son-ld-official-web-standard/
[39] WP REST API [tiešsaiste] – [skatīts: 08.05.2017.]. Pieejams: http://v2.wp-api.org/
[40] An Introduction to CSS Pre-Processors [tiešsaiste] – [skatīts: 08.05.2017.]. Pieejams: https://htmlmag.com/article/an-introduction-to-css-preprocessors-sass-less-stylus
[41] Preprocessors [tiešsaiste] – [skatīts: 08.05.2017.]. Pieejams: http://learn.shayhowe.com/advanced-html-css/preprocessors/
[42] A Modular Approach to Web Development [tiešsaiste] – [skatīts: 09.05.2017.]. Pieejams: https://blog.fedecarg.com/2008/06/28/a-modular-approach-to-web-development/
[43] Twig [tiešsaiste] – [skatīts: 11.05.2017.]. Pieejams: https://twig.sensiolabs.org/
[44] Bootstrap | CSS [tiešsaiste] – [skatīts: 11.05.2017.]. Pieejams: http://getbootstrap.com/css/
[45] CSS with superpowers [tiešsaiste] – [skatīts: 11.05.2017.]. Pieejams: http://sass-lang.com/
[46] Content aware image cropping [tiešsaiste] – [skatīts: 11.05.2017.]. Pieejams: https://github.com/jwagner/smartcrop.js/
[47] jQuery Face Detection Plugin [tiešsaiste] – [skatīts: 12.05.2017.]. Pieejams: https://github.com/jaysalvat/jquery.facedetection
[48] tracking.js [tiešsaiste] – [skatīts: 12.05.2017.]. Pieejams: https://trackingjs.com/docs.html
[49] Stanford Named Entity Recognizer (NER) [tiešsaiste] – [skatīts: 14.05.2017.]. Pieejams: https://nlp.stanford.edu/software/CRF-NER.shtml
[50] Rapid Automatic Keyword Extraction (RAKE) [tiešsaiste] – [skatīts: 18.05.2017.]. Pieejams: https://hackage.haskell.org/package/rake
[51] CLOUD VISION API [tiešsaiste] – [skatīts: 19.05.2017.]. Pieejams: https://cloud.google.com/vision/
[52] Greg j. Badros and Alan borning “The Cassowary Linear Arithmetic Constraint Solving Algorithm” 2002. [tiešsaiste]. Pieejams: https://constraints.cs.washington.edu/solvers/cassowary-tochi.pdf

[53] PageSpeed Insights Rules [tiešsaiste] – [skatīts: 21.05.2017.]. Pieejams: https://developers.google.com/speed/docs/insights/rules

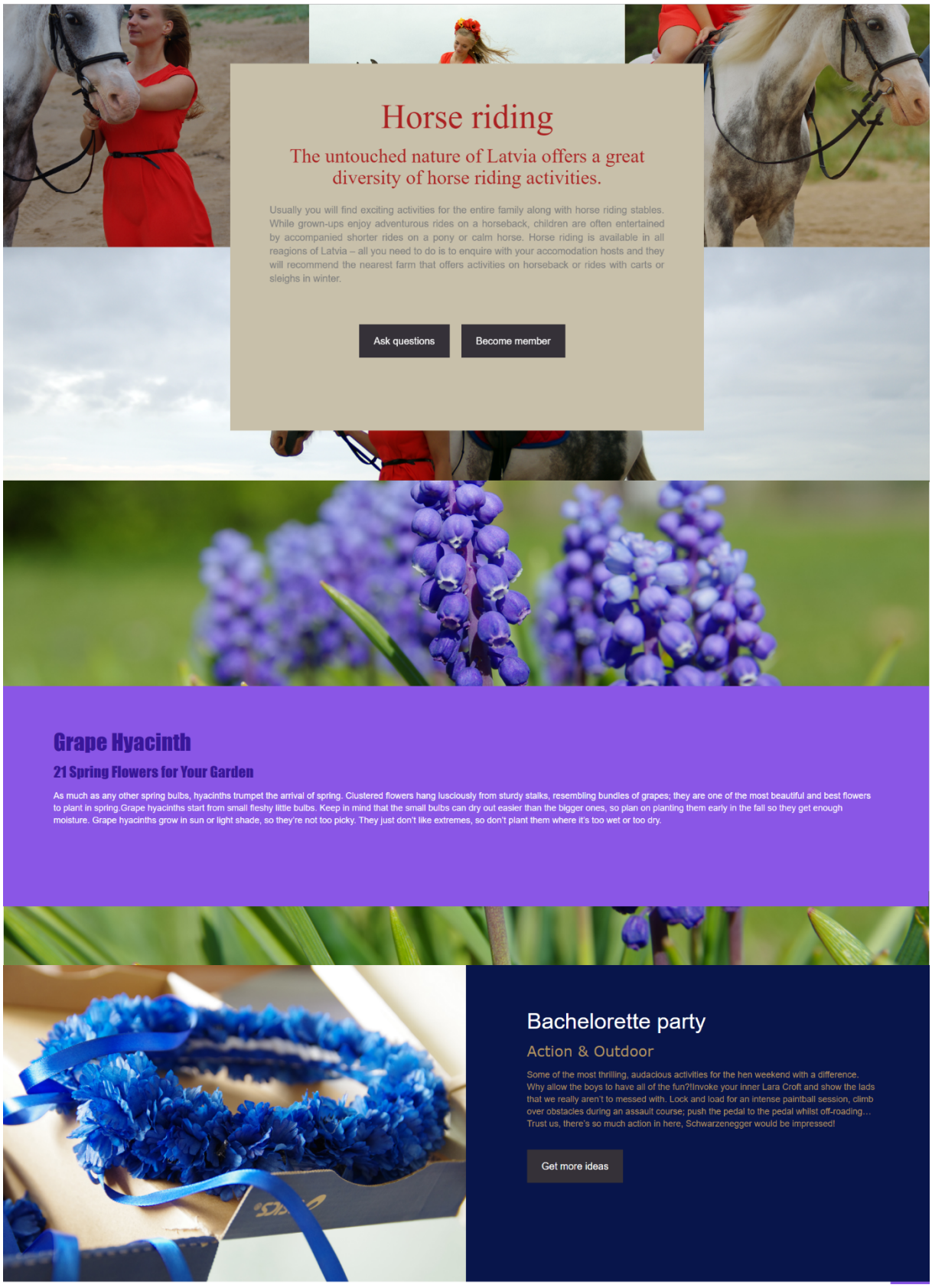
PIELIKUMI

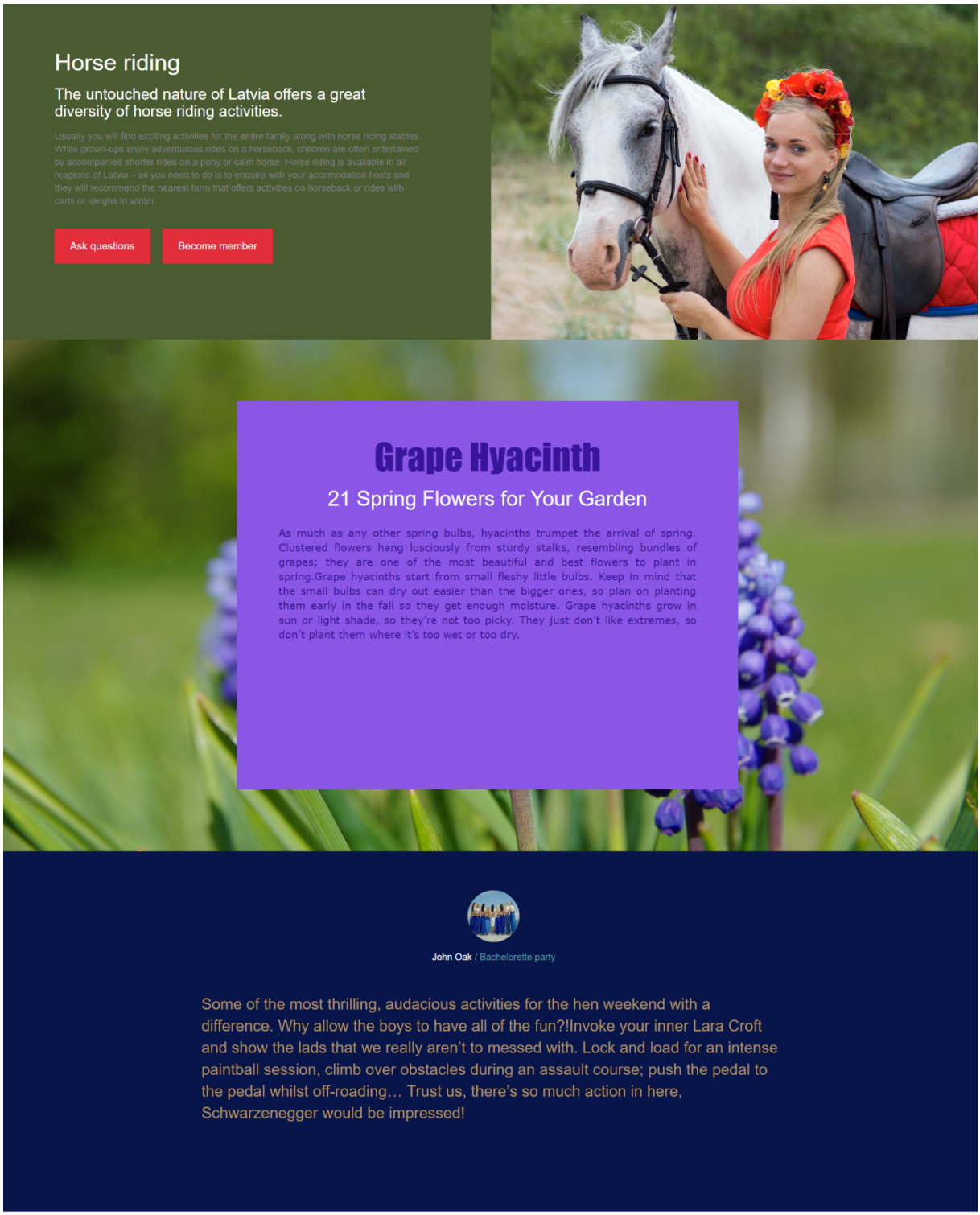
```
{
  "page": {
    "title": "Page title",
    "subtitle": "Page subtitle",
    "logo" : "/data/images/logo.png",
    "favicon" : "/data/images/favicon.png"
  },
  "header": {
    "title" : "Header title",
    "subtitle": "header subtitle",
    "image": "/data/images/image.jpg"
  },
  "navigation": {
    "links": [{
        "title": "Navigation title",
        "url": "https://mvp.lv/"
      }]
  },
  "posts": [{
      "sections": [{
          "title": "Post title",
          "subtitle": "Post subtitle",
          "content": "Post content",
          "date": "07-03-2017",
          "author": "Jānis Ozoliņš",
          "buttons": [{
              "title": "Button title",
              "url": "https://mvp.lv/"
            }],
          "title_image": {
            "title": "Image title",
            "url": "/data/images/image.jpg"
          },
          "image": {
            "title": "Image title",
            "url": "/data/images/image.jpg"
          }
      }]
  }],
  "footer": {
    "title": "Footer title",
    "subtitle": "Footer subtitle",
    "copyrights": "copyrights",
    "navigation": [{
        "title": "Navigation title",
        "url": "https://mvp.lv/"
      }]
  },
  "social": [{
      "title": "Social title",
      "icon": "/data/images/image.png",
      "url": "https://mvp.lv/"
    }]
}
```

2. pielikums. Ģenerētais tīmekļa vietnes stils



3. pielikums. Ģenerētie tīmekļa vietnes bloki




Horse riding
The untouched nature of Latvia offers a great diversity of horse riding activities.
Usually you will find exciting activities for the entire family along with horse riding stables. While grown-ups enjoy adventurous rides on a horseback, children are often entertained by accompanied shorter rides on a pony or calm horse. Horse riding is available in all reagions of Latvia – all you need to do is to enquire with your accomodation hosts and they will recommend the nearest farm that offers activities on horseback or rides with carts or sleighs in winter.
Ask questions
Become member
Grape Hyacinth
21 Spring Flowers for Your Garden
As much as any other spring bulbs, hyacinths trumpet the arrival of spring. Clustered flowers hang lusciously from sturdy stalks, resembling bundles of grapes; they are one of the most beautiful and best flowers to plant in spring.Grape hyacinths start from small fleshy little bulbs. Keep in mind that the small bulbs can dry out easier than the bigger ones, so plan on planting them early in the fall so they get enough moisture. Grape hyacinths grow in sun or light shade, so they're not too picky. They just don't like extremes, so don't plant them where it's too wet or too dry.
John Oak / Bachelorette party
Some of the most thrilling, audacious activities for the hen weekend with a difference. Why allow the boys to have all of the fun?!Invoke your inner Lara Croft and show the lads that we really aren't to messed with. Lock and load for an intense paintball session, climb over obstacles during an assault course; push the pedal to the pedal whilst off-roading… Trust us, there's so much action in here, Schwarzenegger would be impressed!

```
<article class="row article-0">
    <div class="row article-row" style="height: {{ post.image_height }}px;">
        {% set imageCount = 0 %}

        {% if post.image_first %}
            {% for key, section in post.data.sections %}
                {% if section.image and post.image_template[imageCount] is defined %}
                    {% include 'partials/article/image/image-'~ post.image_template[imageCount] ~'.twig' with {
                        'image': section.image
                    } %}
                    {% set imageCount = imageCount + 1 %}
                {% endif %}
            {% endfor %}
        {% endif %}

        {% for key, section in post.data.sections %}
            {% if section.title or section.subtitle or section.content %}
                {% include 'partials/article/section/section-'~ post.section_template ~'.twig' with {
                    'section': section
                } %}
            {% endif %}
        {% endfor %}

        {% if not post.image_first %}
            {% for key, section in post.data.sections %}
                {% if section.image and post.image_template[imageCount] is defined %}
                    {% include 'partials/article/image/image-'~ post.image_template[imageCount] ~'.twig' with {
                        'image': section.image
                    } %}
                    {% set imageCount = imageCount + 1 %}
                {% endif %}
            {% endfor %}
        {% endif %}
    </div>
</article>
```

DOKUMENTĀRĀ LAPA

Maģistra darbs "Automātiska tīmekļa vietņu stila ģenerēšana" izstrādāts LU Datorikas fakultātē.

Darba teksta galīgā versija izgatavota 22.05.2017**.**

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

Autors: _____ (Jānis Ozoliņš)

(Autora paraksts un datums)

Ar savu parakstu apliecinu, ka esmu lasījis augstāk minēto maģistra darbu un atzīstu to par **p i e m ē r o t u / n e p i e m ē r o t u** (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu maģistrantūrā.

Darba vadītājs: _____ (Dr.dat. Uldis Bojārs)

(Vadītāja paraksts un datums)

Darbs iesniegts **maģistratūras sekretariātā** _____.

(Iesniegšanas datums)

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.

Studiju metodiķe: _____.

(Metodiķes paraksts)

Recenzents: Docents, Dr.dat. Sergejs Kozlovičs

(Akad.amats, zin.grāds, vārds, uzvārds)

Darbs aizstāvēts maģistra gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____

(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____

(Sekretāra paraksts)