

Alberta koledža
Informācijas tehnoloģijas
Programmēšana

DMITRIJS JAUNSLAVIETIS

Kvalifikācijas darbs

TĪMEKĻA VIETNES IZSTRĀDE
GRĀMATU IZDEVĒJIEM

Rīga - 2021

ANOTĀCIJA

Kvalifikācijas darba tēma ir “Tīmekļa vietnes izstrāde grāmatu izdevējiem”. Darba mērķis ir izpētīt grāmatu izdevēju nozares nepieciešamību pēc tīmekļa vietnēm un, pirmajā izstrādes ciklā, izstrādāt interneta viedokļa administrēšanas paneļa grafisku saskarni un divus moduļus.

Lai sasniegtu darba mērķi, tika izvirzīti šādi uzdevumi:

- izpētīt digitalizācijas iespējas grāmatu izdevēju nozarē Latvijā;
- salīdzināt iespējamās e-komercijas risinājumu veidus;
- izpētīt un salīdzināt iespējamās izstrādes metodes;
- izpētīt un salīdzināt iespējamās izstrādes tehnoloģijas;
- izstrādāt programmatūras inženierijas dokumentāciju pirmajam izstrādes ciklam;
- izstrādāt administrēšanas paneļa grafisko saskarni un moduļus “Kategorijas”, “Produkti”

Kvalifikācijas darba izstrādē tika izmantotas sekojošas pētījuma metodes: aptauja, intervija, monogrāfiskā metode un analīzes metode.

Darbs sastāv no ievada, 7 pamatnodaļām, secinājumiem, priekšlikumiem, literatūras un izmantoto informācijas avotu saraksta un prezentācijas daļas.

Pirmajā nodaļā ir raksturota problēma un aprakstīta pētīšanas metožu izvēle un to apraksts.

Otrajā nodaļā ir izpētīti iespējamie risinājumi tirgū.

Trešajā nodaļā ir web izstrādes tehnoloģiju apraksts un salīdzinājums.

Ceturtajā nodaļā - izmantojamās programmēšanas valodas un bibliotēkas.

Piektajā nodaļā - programmatūras inženierijas teorētiskais apraksts.

Sestajā nodaļā tiek apkopoti grāmatu izdevēju nozares pētījuma rezultāti.

Septītajā nodaļā ir iekļauti programmatūras izstrādes dokumentu komplekti (prasību specifikācija, un izstrādes gaitas apraksts).

Darba gaitā gūtais secinājums ir tas, ka prasību specifikācijas izstrādes un projektēšanas dziļums ir atkarīgs no projekta izmēra, dažreiz tā var aizņemt vairāk laika, nekā tas ir nepieciešams.

Darba apjoms ir 60 lpp, tas satur 36 attēlus, 3 tabulas un 4 pielikumus. Darba izstrādē izmantoti 34 literatūras avoti.

Atslēgvārdi: grāmatu izdevēji, Laravel, React, programmatūras inženierija, interneta veikals

ANOTATION

Qualification thesis is “Development of a Website for a Book Publishers”. The goals of this document are to study the need of a website in the book publishing industry and to develop a graphical interface and two modules for an online web administration panel, in the first development cycle.

To achieve the goals of the work, the following tasks were set:

- to study the possibilities of digitization in the book publishing industry in Latvia;
- compare possible types of e-commerce solutions;
- to study and compare possible development methods;
- to study and compare possible development technologies;
- develop software engineering documentation for the first development cycle;
- develop a graphical interface and modules for "Categories" and "Products" in the administration panel

The following research methods were used in the development of the qualification paper: survey, interview, monographic method and analysis method.

The work consists of an introduction, 7 main chapters, conclusions, suggestions, a list of literature and information sources used and a presentation.

The first chapter describes the problem and describes the choice of research methods and their description.

The second chapter explores possible market solutions.

The third chapter describes and compares web development technologies.

In the fourth chapter - the programming languages and libraries to be used.

In the fifth chapter - a theoretical description of software engineering.

The sixth chapter summarizes the results of a study of the book publishing industry.

The seventh chapter includes sets of software development documents (requirements specification, and description of the development process).

The main conclusion is that the depth of development and design of the requirements specification depends on the size of the project, sometimes it may take longer than necessary.

The volume of the work is 77 pages, it contains 48 figures, 6 tables and 34 sources of information were used in the development of the paper.

Keywords: book publishers, Laravel, React, software engineering, online store

SATURA RĀDĪTĀJS

IEVADS	6
1. PROBLĒMAS AKTUALITĀTES PĒTĪJUMA METODES	8
1.1. Intervija ar izdevniecības pārstāvi	8
1.2. Grāmatu izdevēju aptauja par tīmekļa risinājumu izmantošanu uzņēmumā	8
2. IESPĒJAMIE RISINĀJUMI TIRGŪ	9
3. WEB IZSTRĀDES TEHNOLOĢIJAS	11
4. IZMANTOJAMĀS PROGRAMMĒŠANAS VALODAS UN BIBLIOTĒKAS	14
4.1. Servera puse	14
4.2. Klienta pusē – administrācijas panelis	15
4.3. Klienta pusē – interneta veikala vietne	16
4.4. Izstrādes vide	17
4.5. Datubāze	18
4.6. Izmantojamie rīki	19
5. PROGRAMMATŪRAS INŽENIERIJA - TEORĒTISKĀ DAĻA	20
5.1. Programmatūras dzīves cikls	21
5.2. Programmatūras izstrādes metodes	21
5.2.1. Ūdenskrituma modelis	22
5.2.2. Agile modelis	23
5.2.3. Prototipēšanas modelis	24
5.3. Prasību dokuments	24
5.4. Prasību definēšanas pieejas	25
5.5. Sistēmas prasību specifikācija	26
5.6. Programmatūras projektēšana	29
5.7. Testēšanas dokumentācija	29
5.8. Lietotāja ceļvedis	30
6. NOZARES PĒTĪJUMA REZULTĀTS	31
7. PROGRAMMATŪRAS INŽENIERIJA - PRAKTISKĀ DAĻA	33

7.1. Prasību definīcija tīmekļa risinājumam.....	33
7.2. Prasību specifikācija	34
7.2.1 Administrācijas panelis.....	34
7.3. Projektējums	39
7.3.1 Izmantojamās tehnoloģijas.....	39
7.3.2 Lietotāja saskarnes modelēšana	39
7.3.3 Datu dekompozīcija	42
7.3.4 Algoritmu apraksts.....	43
7.4. Izstrāde.....	46
7.4.1 Izstrādes vides izveidošana.....	46
7.4.2 API konfigurēšana	47
7.4.3 Lietotāja saskarnes veidošana.....	47
7.4.4 Autorizācijas realizācija.....	49
7.4.5 Kategoriju koka izvadīšana.....	50
7.4.6 Ierobežojumi	51
SECINĀJUMI	52
PRIEKŠLIKUMI	53
LITERATŪRAS UN INFORMĀCIJAS AVOTU SARAKSTS.....	54
PREZENTĀCIJAS DAĻA	57
PIELIKUMI	61

IEVADS

Mūsdienās jebkurš uzņēmums vai organizācija vairāk nav iedomājami bez tīmekļa risinājumiem, tie var būt lielāka vai mazāka izmēra interneta veikali, kas ir vērsti uz dažādu preču pārdošanu vai noteiktas kategorijas precēm, pakalpojumu sniegšanas tīmekļa risinājumi, kā arī mārketinga funkciju veicošas tīmekļa lapas “vizītkartes”.

Kvalifikācijas darba tēma ir grāmatu izdevējiem paredzēta tīmekļa vietne, šāda ideja autoram radās, jo bija sadarbības pieredze ar grāmatu izdevēju sīa “AVOTS”. Sadarbība izpaudās uzņēmuma interneta veikala un bloga tīmekļa vietnes daļu izstrādē, kā rezultātā tika iegūts priekšstats par nozari, problēmām nozarē un tīmekļa risinājumu trūkumu šajā nozarē Latvijā.

Grāmatu popularitātes krituma, dēļ citu informācijas avotu un izklaides iespēju attīstības (lielā merā interneta), samērā nelieliem grāmatu izdevējiem, kuriem bieži vien nav arī grāmatnīcas, ir ierobežots budžets. Ņemot vērā aizvien pieaugošo attīstību e-komercijas risinājumu jomā, pieaug arī izstrādes un uzturēšanas izmaksas, taču, lai arī tā augošo digitalizācijas tempu, pieprasījumu pēc risinājumiem veicināja arī covid-19 pandēmijas radītie ierobežojumi klātienē tirdzniecībai, tādēļ risinājumi tiek izmantoti bieži vien nepiemēroti nišai, vai arī novecojuši un grūti uzturami saviem spēkiem.

Ņemot vērā situāciju, autors izvirza šādus kvalifikācijas darba ietvaros veicamus mērķus:

- izpētīt grāmatu izdevēju un tirgotāju nozari Latvijā, par galveno pētījuma objektu izvirzot tīmekļa un e-komercijas risinājumus nozarē;
- izpētīt kādas ir minimālās prasības e-komercijas produktam ņemot vērā nišas īpašības;
- izpētīt iespējamās mūsdienu risinājumus tirgū;
- izveidot iespējamā risinājuma projektējumu;
- sākt risinājuma izstrādi.

Lai sasniegtu izvirzītos mērķus, autors izvirza šādus uzdevumus:

- aptaujāt un intervēt nozares uzņēmumus;
- izpētīt pieejamākos e-komercijas risinājumus;
- apkopot ievākto informāciju, veikt analīzi;
- pamatojoties uz iegūto informāciju piemeklēt tīmekļa vietnes veidošanai piemērotākās tehnoloģijas un programmēšanas valodas;
- izveidot tīmekļa vietnes projektējuma dokumentāciju (prasību specifikāciju,

projektējumu, testēšanas dokumentu un lietotāja ceļvedi);

- sākt tīmekļa vietnes izstrādi, un prezentēt izstrādātos moduļus;

Kvalifikācijas darba mērķu sasniegšanai tiks izmantotas šādas pētījumu metodes:

- Intervija ar izdevniecības “AVOTS” pārstāvi – lai sagatavotu prasību specifikāciju.
- Aptauja grāmatu izdevējiem – lai izpētītu tēmas aktualitāti un galvenās prasības tīmekļa vietnei.
- Analīzes metode – lai varētu salīdzināt pieejamos risinājumus tirgū, jāsalīdzina to piedāvājums un atšķirības, izmantojot pieejamo informāciju izstrādātāju interneta vietnēs.
- Monogrāfiskā metode – izpētīt prasības mūsdienu e-komercijas risinājumiem specializētā literatūrā

Informācijas iegūšanai tika izmantoti:

- dažāda veida interneta resursi (Blogi, programmēšanas valodu un tīmekļa tehnoloģiju dokumentācija, e-komercijas risinājumu vietnes);
- literatūra (grāmatas kurās tiek aprakstīti tīmekļa izstrādes un e-komercijas produktu principi un risinājumi);
- mācību materiāli, kuri tika iegūti mācību procesa laikā koledžā un papildus izglītībasursos.

Kvalifikācijas darbs tika izstrādāts 2021. gada rudens semestrī, Alberta koledžā, Rīgā. Darba izstrādes periods 26.10.2021. – 06.01.2022., kas arī tiek uzskatīts par pētījuma periodu.

1. PROBLĒMAS AKTUALITĀTES PĒTĪJUMA METODES

Lai gūtu padziļinātu priekšstatu par situāciju un problēmām e-komercijas risinājumu izmantošanā Latvijas grāmatu izdevēju nozarē, tika izmantotas šādas trīs pētījuma metodes:

1. intervija ar izdevniecības SIA “AVOTS” pārstāvi;
2. elektroniska aptauja grāmatu izdevējiem, par e-komercijas risinājumiem uzņēmumā;
3. Latvijas grāmatu izdevēju interneta vietņu un veikalu apzināšana un iepazīšanās ar izmantotajiem risinājumiem.

1.1. Intervija ar izdevniecības pārstāvi

Informācijas par nozari iegūšanas nolūkā, autors izveidoja intervijas jautājumus, kurus uzdot nozares speciālistiem. Intervijas jautājumi bija sekojoši:

- cik grāmatu izdevēji ir Latvijā?
- cik un kuri ir lielākie izdevēji Latvijā?
- vai visi grāmatu izdevēji paši tirgo savas izdotās grāmatas?
- kādi ir realizācijas veidi? (Tirgo paši savas grāmatas, tirgo savas un citu izdevēju grāmatas, ir internetveikals, ir fizisks veikals (grāmatnīca), citi?)
- kādas ir asociācijas, biedrības - kā tās veicina tirdzniecību?
- vai ir vēl kādi komentāri, informācija, kura varētu būt noderīga pētot Latvijas grāmatu izdevēju nozares situāciju un problēmas?

Šos jautājumus autoram bija iespēja uzdot izdevniecības “Avots” valdes loceklim Jānim Lejam, izpilddirektorei Laumai Lejai un “Latvijas Grāmatizdevēju asociācijas” izpilddirektorei Marikai Celmai. Intervijas notika gan klātienē, gan elektroniski nosūtot uz e-pastu jautājumus.

1.2. Grāmatu izdevēju aptauja par tīmekļa risinājumu izmantošanu uzņēmumā

Lai paplašinātu izpratni par Latvijas grāmatu izdevēju nozares digitalizācijas un e-komercijas rīku izmantošanas līmeni, tika izveidota aptauja. Lai aptauja aptvertu pēc iespējas lielāku respondentu skaitu, tā tika nosūtīta izdevējiem ar izdevniecības “AVOTS” un Latvijas Grāmatizdevēju asociācijas palīdzību. Aptaujas norises laiks 2 nedēļas no 8. novembra līdz 22. novembrim 2021. gadā.

2. IESPĒJAMIE RISINĀJUMI TIRGŪ

Uz pētījuma brīdi pastāv vairāki e-komercijas risinājumu veidi, kuri spēj apmierināt lielāku un mazāku uzņēmumu vajadzības un īpatnības.

Pamatā var izšķirt 4 interneta veikalu izstrādes pieejas (Kseniia Kyslova (2021). eCommerce web development: What approach and tools are right for your online store?):

1. tīmekļa lapu konstruktori ar e-komercijas moduļiem; (WordPress + WooCommerce)
2. e-komerces platformas, kuras ir iepriekš izveidotas, un to ražotājs atbild par hostingu, uzturēšanu un atbalstu; (Shopify, BigCommerce, Mozello, u.c.)
3. e-komerces atvērtā koda platformas, konstruktori, kuri piedāvā lejupielādēt bāzes programmu, uz kuras pamata izstrādāt interneta veikalu; (PrestaShop, Opencart, u.c.)
4. individuāli izveidots e-veikals.

Tīmekļa lapu konstruktori ar pieslēgtu e-komercijas moduli, populārākais WordPress blogošanas vietnes konstruktors ar pievienotu spraudni WooCommerce, ir salīdzinoši lēts un ērts risinājums nelieliem uzņēmumiem ar mazu produktu spektru. E-komercijas modulis šajā gadījumā tiek pievienots gatavai tīmekļa lapai, kurš ļauj izveidot produkta pirkšanas formu klienta pusē un pievienot pirkumu pārvaldīšanas moduli lapas satura pārvaldīšanas sistēmai.

Šāds e-komercijas risinājums ir ērts risinājums gadījumos, kad jau ir izveidota tīmekļa vietne, it īpaši ja moduļa ražotājs sadarbojās ar tīmekļa vietnes izstrādes platformu. Tā kā šādi risinājumi parasti tiek bieži atjaunināti un ir tehniskais atbalsts, tad retāk jāuztraucās par nekorektu darbību, vai papildus speciālista nepieciešamību.

Kā mīnusu var atzīmēt izmaksas un ierobežojumi plaši pielāgot individuālām uzņēmuma vajadzībām. Piemēram WooCommerce modulim izmaksas tiek aprēķinātas pēc daudziem kritērijiem, dārgākais ir daļas maksa par katru veikto pirkumu (Kathrun Marr (2018). WooCommerce Pricing: How Much Does it Cost to Run a Store?). Pielāgošana ierobežojās ar ražotāja izveidoto funkcionalitāti. Šādām platformām ir ierobežota sadarbība, vai pielāgošana individuāli izveidotām tīmekļa vietnēm var būt sarežģīta vai neiespējama.

E-komerces platformas, kuras ir iepriekš izveidotas, un to ražotājs atbild par hostingu, uzturēšanu un atbalstu ir piemērotas uzņēmumiem, kuri nevēlās maksāt par interneta veikala izveidošanu izstrādātājiem. Piemēram Shopify piedāvā gatavus interneta veikalus, kurus viegli atvērt un izmantot savām vajadzībām pašu spēkiem.

Galvenās šāda risinājuma priekšrocības ir tas, ka veikalu ir iespējams atvērt ātri paša spēkiem, tīmekļa vietni uztur un atbalsta ražotājs un ja tā ir pazīstama liela platforma, nav jāuztraucās par drošību un darbību.

Mīnusi ir izmaksas un interneta veikala pārvešana uz citu platformu. Izmaksas sastāv no daudzām komponentēm, parasti tā ir abonēšanas maksa un daļa no pārdošanas ienākumiem. Ja uzņēmums vēlēties pārnest interneta veikalu ar precēm un visiem veikala datiem uz citu platformu vai pat individuāli izveidotu tīmekļa lapu, var rasties ierobežojumi to izdarīt.

Atvērtā koda risinājumi tādi kā OpenCart vai Magento piedāvā gatavu atvērtā koda bāzes interneta veikalu, uz kura pamata var izstrādāt savu individuālu interneta veikalu izmantojot platformai izveidotus moduļus, izveidot savus moduļus, kā arī mainīt kodu un arhitektūru.

Pieejas galvenās priekšrocības ir izstrādes brīvība – ir iespēja izstrādāt individuālu interneta veikalu uz iepriekš izveidota interneta veikala bāzes kura ir notestēta un pielāgota tirgošanas vajadzībām.

Galvenais risinājuma mīnuss ir tāds, ka šādā gadījumā būs nepieciešams algot izstrādātāju, kurš ne tikai izstrādās uzņēmuma prasībām atbilstošu interneta veikalu, bet arī spēs veikt sistemātiskus ražotāja atjauninājumus, tas var radīt sarežģījumus, ja sākuma kods ir modificēts, taču atjauninājumi ir nepieciešami, jo šādas sistēmas parasti tiek biežāk pakļautas drošības riskiem.

Individuāli izveidots interneta veikals – dod uzņēmumam inovāciju brīvību un pielāgošanas iespējas tieši savām vajadzībām. Individuāli izstrādātas tīmekļa vietnēs nav nekā lieka, tādēļ ātrdarbība parasti ir labāka, kā arī ir brīvība tehnoloģiju izmantošanā.

Mīnusi šādai pieejai ir izmaksas un prasību definēšanas spēja. Izmaksas šādas pieejas gadījumā parasti ir atkarīgas no vēlāmās funkcionalitātes un var sasniegt vairākus desmitus tūkstošus eiro, taču labi izstrādāta tīmekļa vietne prasīs vienu ieguldījumu, un turpmāk tā piederēs uzņēmumam, protams papildus izmaksas var radīt apkalpošana, taču atkrīt papildus izmaksas no pārdošanas daudzuma, u.c. Liela problēma varētu būt slikti definētas prasības no klienta puses, taču šāda risinājuma gadījumā izstrādātāju komandā ir arī analitiķi, kuri palīdzēs definēt biznesa modeli, un palīdzēs klientam rast pareizo risinājumu.

Autora izvēlētais risinājums ir nozarei paredzēta individuāla projekta izveide, kas ir grāmatu izdevēji un tirgotāji, šis risinājums iekļaus sevī tikai nepieciešamākās funkcijas e-komercies risinājumam kurš būs pieejams arī mazu uzņēmumu budžetam.

3. WEB IZSTRĀDES TEHNOLOĢIJAS

Veidojot individuālu, modernu tīmekļa vietni, galvenās sastāvdaļas ir lietotāja saskarne, datubāze un veids, kā šīs daļas sadarbojās savā starpā. Divas populārākās web izstrādes pieejas ir datu pārraide izmantojot HTTP metodes (tiek saukts arī par tīmekļa pakalpojumiem WebServices) vai izmantojot API pieejas punktus, precīzāk REST vai CRUD API. Lai arī REST api arī izmanto HTTP protokolu datu pārraidei uz un no servera, lietotņu izstrādes pieejas ir atšķirīgas.

HTTP protokols ir pamats tam kā notiek jebkādu datu apmaiņa starp tīmekli un lietotāju, tas ir klienta-servera protokols, pieprasījumus veido klients, parasti interneta pārlūkprogramma. Šis ir aplikācijas slāņa protokols, kurš var tikt nosūtīts izmantojot teorētiski jebkādu transporta slāņa protokolu. Protokola evolūcijas gaitā, tas ir kļuvis ļoti paplašināms, un tādēļ tiek izmantots ne tikai hiperteksta dokumentu pārraidei, bet arī attēlu, video un citu datu pārraidei. (MDN Web Docs (2021). An overview of HTTP.)

Izstrādājot Tīmekļa lietotnes izmantojot HTTP metodes, tiek izmantotas HTTP standarta metodes, protokolam tās ir vairākas, dažas no tām (PROTOCOL SUPPORT LIBRARY (2021). Hypertext Transfer Protocol (HTTP)):

- GET – pieprasa datus no servera;
- HEAD – pieprasa datus no servera bez ķermeņa elementa;
- POST – pievienotos pieprasījumam datus nosūta serverim;
- PUT – pievienotos pieprasījumam datus nosūta serverim, ja šādi dati eksistē, tad pārraksta;
- DELETE – pieprasījums dzēst kādu serverī esošu resursu;
- PATCH – daļēji modificē serverī esošu resursu, datus;

HTTP metožu izstrādes pieejas pamatprincips ir sekojošs:

Klients, jeb interneta pārlūks atkarībā no lietotāja darbībām, piemēram klikšķis uz hipersaites, izveido pieprasījuma ziņu serverim. Pieprasījuma ziņa satur:

- Pieprasījuma līniju – satur pieprasījuma metodi (GET, POST, u.c.), pieprasījuma URI, un HTTP protokola versiju;
- Pieprasījuma galveni – tā satur dažādu informāciju, piemēram valodas kodējumu, simbolu kodējumu, klienta informāciju, pārlūka versiju, satura tipu u.c.;
- Ziņojuma ķermenis – satur datus un resursus, kurus nosūta klients.

Atbildē uz pieprasījuma ziņojumu klients saņem Http atbildes ziņojumu, kurā ir iekļauts:

- atbildes josla – Http versija, atbildes kods, statuss;
- atbildes galvene – tā satur dažādu informāciju, piemēram – atbildes datumu, servera tipu, pēdējo izmaiņu datumu, datu tipu, datu apjomu, u.c.;
- atbildes ķermeni – dati, kurus atgrieza serveris atbildē uz pieprasījumu.

Tātad, veidojot tīmekļa aplikācijas saziņu izmantojot standarta HTTP metodes – klienta pusē tiek izvietotas hipersaites ar noteiktu HTTP metodi un URI adresi, šis pieprasījums nosūtās uz serveri, un no servera tiek iegūts fails atbilstoši ceļam URI vai arī palaista programma atbilstoša šim ceļam, ņemot vērā metodi un nosūtītos resursus pieprasījumā.

Otra un modernāka pieeja izstrādājot tīmekļa lietotnes ir API. Lai gan API datu pārraide notiek izmantojot HTTP protokolu, arhitektūra ir savādāka. API ir veids kā lietotnes sazinās savā starpā. Lietotnē izņemot tās funkcionalitāti tiek izveidots piekļuves punkts un noteikumi ar kuriem tā sazinās ar citām lietotnēm (RapidAPI (2019). API vs Web Service: What's the Difference?). Šis piekļuves punkts ir vienīgais ceļš kā sazināties ar lietotni un saņemt no lietotnes nepieciešamo atbildi. Parasti katrai nepieciešamai funkcijai ir savs piekļuves punkts, metode un arī parametri, ko šis punkts pieņem.

Tīmekļa tehnoloģijās visbiežāk tiek izmantots RESTful jeb REST api. Lietotne, kura izmanto saziņai REST API, izmanto HTTP protokolu, lai saņemtu pieprasījumus, atbildē lietotne nosūta uz servera sagatavotu objektu ar klienta pieprasītiem datiem, pēc lietotnes iebūvētiem noteikumiem. Izmantojot šādu pieeju dati parasti tiek atgriezti JSON formātā nevis XML, kā klasiskajā HTTP pieprasījumā.

Galvenie REST API principi ir:

- Bezstāvokļa protokols (Stateless) – klients nosūta sagatavotu ziņojumu serverim – un serveris to apstrādā un no sūta sagatavotu atbildi. Serverim nav nepieciešams uzturēt jebkāda veida stāvokli.
- Klienta-servera arhitektūra – arhitektūra nosaka, ka klients nav piesaistīts serverim – tas uzlabo pārnēsājamību un mērogojamību.
- Piemērotība kešošanai – resursi var būt kešojami gan servera gan klienta pusē ātrdarbības uzlabošanai.
- Vienveidīga saskarne – pieprasījumi vienam un tam pašam resursam ir vienādi, REST arhitektūra paredz, ka resurss nevar būt liels, taču tam jā satur visa nepieciešamā informācija, ko pieprasa klients.
- Slāņu struktūra – REST API pieprasījumi var iziet cauri vairākiem pieprasījumu slāņiem, ne klients ne serveris nezina, cik starp tiem ir slāņi.

Taču REST api ir tikai izstrādes princips, jeb arhitektūra, metode, kura izmanto šīs

arhitektūras principus saucās CRUD API.

CRUD API ir akronīms vārdiem CREATE, READ, UPDATE un DELETE, jeb izveidot, nolasīt, atjaunot un dzēst. Šīs ir CRUD API metodes, kurām ir saskatāma līdzība ar standarta HTTP metodēm:

- CREATE – POST;
- READ – GET;
- UPDATE – PUT;
- DELETE – DELETE.

Tas ir abstrakcijas līmenis, kurš ievieš kārtību izstrādājot api, piemēram ir definēta būtība grāmata, darbības, kuras mēs varam veikt ar grāmatu, definējam piekļuves punktus (Eban Scott (2020). How does CRUD relate to a REST API?):

- CREATE - /api/book
- READ - /api/book/{id}
- UPDATE - /api/book
- DELETE - /api/book/{id}

Id šajā gadījumā norāda uz konkrētu grāmatas ierakstu.

Lai arī šī ir ļoti vienkārša pieeja, tā nosedz tikai standarta nepieciešamās vienkāršākās lietotnes darbības, bieži vien pieprasījumi ir daudz sarežģītāki, piemēram tādi, kas prasa vairāku tabulu apvienošanu konkrētos gadījumos, bet ne vienmēr, šādos gadījumos, jauna metodes veido pats izstrādātājs, un attiecīgi katrai metodei nozīmē savu piekļuves punktu un ieejas datus (James Higginbotham (2018). Taking REST beyond CRUD).

Api pieeja autoram šķiet, ļoti elastīga un paplašināma, API pieprasījumi var sadalīt mazās daļās lielākus objektus (piemēram produktu sarakstu) un saņemt to tikai pēc noteikta pieprasījuma pa daļām. Tāpat vilinoša šķiet ideja, saņemt datus JSON objekta formā, kura struktūru autors pats varēs viegli koriģēt izstrādes gaitā. Daudz mazu piekļuves punktu, būs arī ļoti piemēroti izstrādājot klienta vietnes daļu izmantojot javascript bibliotēku React, kura labi sadarbojās ar pakotni axios – pieprasījumu veikšanai.

4. IZMANTOJAMĀS PROGRAMMĒŠANAS VALODAS UN BIBLIOTĒKAS

4.1. Servera puse

Lai izstrādātu uz REST API arhitektūras balstītu tīmekļa lietotni, ir nepieciešams izvēlēties servera puses ietvaru, kas tam ir piemērots. Mūsdienās šo satvaru klāsts ir ļoti plašs, to galvenā atšķirība bieži vien ir izmantojamā programmēšanas valoda un vairākums no tiem ir balstīts uz MVC (Model view controller) izstrādes modeli.

Izstrādājot lietotnes, bieži tiek pieminētas izstrādes arhitektūras jeb paterni. Servera puses lietotnēm populārākais ir MVC, jeb Model view controller patrons, taču ir arī citi piemēram MVP (Model View Presenter), MVVM (Model View View-Model), un citi.

Populārākais no tiem ir Model View Controller patrons. Šajā paternā lietotne ir sadalīta abstraktās trijās daļās:

- Modelis (Model) – modelis atspoguļo datus kurus ir nepieciešams parādīt skatā (view). Modelis sastāv no klasēm un kolekcijām, kuras apraksta biznesa loģiku, un noteikumus pēc kuriem dati tiek manipulēti;
- Skats (View) – skats ir lietotnes daļa, kura parāda datus, kuri ir saņemti no kontroliera (controller). Parasti HTML vai XML.
- Kontrolieris (Controller) – kontrolieris apstrādā ienākošos pieprasījumus, tas apstrādā datus kuri ir iegūti no modeļa un padod tos atpakaļ skatam.

Šis izstrādes modelis arī tiks pielietots izstrādājot projektu dēļ tā popularitātes un autora zināšanām par to, kuras tika iegūtas studiju laikā.

Populārākie ietvari, kuri ir pielāgoti CRUD API veidošanai ir:

- Express.js – šis ietvars izmanto javascript programmēšanas valodu servera pusē. Šis ietvars ir ļoti populārs dēļ augstas Node.js (servera javascript lietotnes) popularitātes, maza izmēra, elastīguma un ātruma, tas atbalsta arī REST API izveidi, taču lielākais mīnuss ir vāji definēts izstrādes veids, kas var kļūt par klupšanas akmeni iesācējiem (Aman Goel (2021). 10 Best Web Development Frameworks).
- Spring – šis ietvars izmanto Java programmēšanas valodu servera pusē. Ietvars seko MVC paterna izstrādes principiem un ir plaši izmantots, taču tas ir sarežģīts apgūšanā un prasa ļoti labas Java valodas zināšanas un pieredzi (Aman Goel (2021). 10 Best Web Development Frameworks).
- Laravel – šis ietvars izmanto PHP programmēšanas valodu, kura ir pirmā un

populārākā servera puses lietotņu izstrādes valoda. Laravel ir salīdzinoši jauns ietvars, kurš seko MVC paterna izstrādes principiem un ir piemērots API izstrādei tā pamatā. Šis ietvars ir ļoti piemērots iesācējiem, dēļ popularitātes, tam ir liela kopiena, kas ļoti atvieglo problēmu risināšanu, un jau gatavu risinājumu klāsts. Tomēr ir novērots, ka lielos projektos, tas ir lēnāks par konkurentiem piemēram Express.js un Django (Aman Goel (2021). 10 Best Web Development Frameworks.

Autors izstrādājot projektu pieņēma divus iespējamus variantus ietvara izvēlē – Express.js un Laravel (PHP). Express.js vilināja ar Javascript valodu un Node.js servera lietotnes esamību, kas iespējams būtu labāk piemērots Javascript klienta puses satvara React.js apvienojumā, taču dēļ pieredzes darbā, zemas sarežģītības, komplektā API funkcionalitātes un pilnīgas MVC paterna esamības Laravel atšķirībā no Express.js, kurā nav strikti izstrādes principi, tika izvēlēts Laravel ietvars.

4.2. Klienta pusē – administrācijas panelis

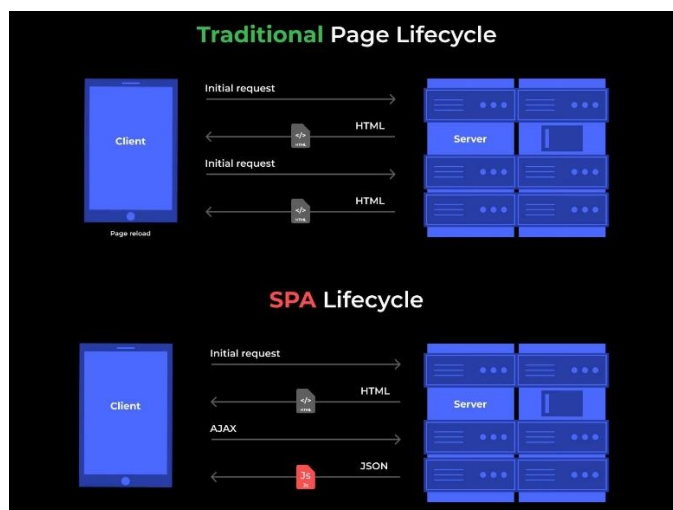
Tīmekļa vietnes, it īpaši interneta veikali, nav iedomājamas bez administrēšanas daļas, jeb satura pārvaldības daļas. Administrācijas daļa būtībā ir formas, tai ir jānodrošina satura papildināšana, datu izvadīšana un ātru saskarni. Mūsdienās populārs veids, kā izstrādāt sarežģītas, bet ātras tīmekļa saskarnes ir SPA (Single Page Application), jeb vienas lapas lietotnes principu.

Klienta pusē, parasti interneta pārlūks, attēlo informāciju izmantojot HTML dokumentu, kurš ir stilizēts izmantojot CSS stilu tabulas. Klasiskajā pieejā, kad lietotājs pieprasa kādu resursu, jeb tīmekļa lapu – pieprasījumam no servera klientam tiek nosūtīts hiperteksta fails XML formātā, jeb HTML. Lai lapas būtu dinamiskas, serveris atkarībā no pieprasījuma, veic izmaiņas, izveido izmainītu HTML lapu un nosūta atbildē klientam, apskatāms 4.1. att. augšējā shēma.

Taču vienas lapas lietotnē HTML fails var saturēt tikai vienu rindiņu, kas kalpo par piekļuves punktu Javascript lietotnei, kura ģenerē HTML koka struktūru atkarībā no pieprasījuma (Artem Cervichnik Svitlana Varaskina (2021). Single-Page Applications vs Multi-Page Applications: The Battle of the Web Apps). Atkarībā no lietotāja darbībām, javascript izmantojot AJAX asinhronos pieprasījumus, var sūtīt pieprasījumus uz serveri, lapas lietošanas laikā. Atkarībā no pieprasījuma no servera dati tiek saņemti JSON formātā, kurš satur strukturētu objektu, kurš arī izmaina lapas saturu un izskatu, apakšējā shēma 4.1. att. attēlā.

Galvenās priekšrocības SPA pieejai ir ātrums – lapas saturs tiek atjaunots tikai tas, kurš

tiek pieprasīts – piemēram interneta veikalam nevajag ielādēt visus produktus uzreiz, bet piemēram lādēt klāt kamēr lietotājs tin lapu lejā.



4.1.att. Tradicionālā klienta puses vietnes izstrāde un vienas lapas lietotnes izstrādes pieeja (Artem Cervichnik Svitlana Varaskina (2021). Single-Page Applications vs Multi-Page Applications: The Battle of the Web Apps)

Lai arī šāda veida lapu var izstrādāt izmantojot Javascript bez ietvariem, mūsdienās ir izstrādāts daudzi augstieftīvi ietvari, kas atvieglo un paātrina izstrādi.

Populārākie ietvari ir:

- Angular – šo ietvaru ir izstrādājis “Google”. Tas ir visapjomīgākais no šiem trijiem ietvariem, un prasa vairāk pieredzes, nekā citi. Šajā satvarā ir daudz iebūvēti moduļi, un ir mazāka nepieciešamība pēc papildus moduļiem.
- Vue – šis ietvars ir atvērtā koda ietvars, un to izstrādā kopiena. Vue tiek atzīts, kā vienkāršākais apguvei, tas ir arī visjaunākais ietvars, un tiek uzskatīts par daudzsološu nākotnē.
- React – šo ietvaru ir izstrādājis un izmanto Facebook. Tam ir mazāk iebūvētu moduļu, taču viss nepieciešamais ir pieejams papildus. Lai gan tas ir lēnāks un mazāks kā pārējie, šobrīd tas ir populārākais ietvars, un ir piemērots gan lieliem gan maziem projektiem. Tiek modernizēts un regulāri atjaunots.

Angular šim projektam šķiet par apjomīgu, izvēle projektā izmantojamai tehnoloģijai krīt uz Vue vai React, taču dēļ pieredzes darbā šim projektam autors izmantos React.js un tam pieejamās pakotnes – axios (http pieprasījumiem), react-router navigācijai lapā un react-bootstrap saskarnes vizuālo elementu veidošanai.

4.3. Klienta pusē – interneta veikala vietne

Lai arī vienas lapas izstrādes pieejas ātrums un ērtums ir ļoti vilinošs, tās nav labi

piemērotas labai meklētājprogrammu optimizācijai, jeb SEO (Bartosz Staryga (2021). SPA SEO: Mission Impossible?).

Šī problēma rodas dēļ tā, ka piemēram Google meklētājprogrammas indeksēšanas robots, nokļūstot vietnē, apstrādā visas pieejamās saites un ceļus kuri ir pieejami HTML dokumentā. Taču kā tika minēts iepriekšējā nodaļā, SPA lietotnēs HTML fails var būt tikai kā ieejas punkts Javascript lietotnei un tas var saturēt tikai vienu rindiņu. Šādos gadījumos, robots palaiž javascript un saņem saites, taču tas notiek lēni un izpētīt visus ceļus bieži vien nav iespējams, tādēļ šādām lapām ir zems SEO reitings.

Iespējamie risinājumi ir:

- Atteikties no SPA pieejas labākam SEO – šāda pieeja uzlabos lapas optimizāciju meklētājprogrammām, taču zudīs arī SPA aplikāciju priekšrocības – ātrums un interaktivitāte, kā arī izstrāde izmantojot SPA pieeju bieži vien ir ātrāka un strukturētāka.
- Izmantot SPA lietotņu renderēšanu uz servera – šis ir jauns risinājums, kurš paredz to, ka Javascript lietotnes renderēšana notiek nevis klienta pusē, bet uz servera – klients saņem uzģenerētu HTML kodu – šī pieeja atrisina ar SEO saistītas problēmas, taču tas samazina lietotnes darbības ātrumu un pieprasa javascript servera ietvara izmantošanu.
- Izstrādāt SEO svarīgas daļas HTML, taču nesvarīgas priekš SEO un sarežģītas saskarnes un formas – izmantojot React komponentes. Šo pieeju izvēlās autors ņemot vērā izvēlēto servera puses ietvaru Laravel, kurš nav Javascript bāzēts ietvars un kuram ir iebūvēta skatus izstrādes tehnoloģija blade. Blade sūta klientam HTML failu, kurš ir piemērots SEO, taču izstrāde Laravel ietvarā, līdzinās SPA lietotņu izstrādes veidam sadalot lapu komponentēs.

4.4. Izstrādes vide

Izvēloties piemērotu izstrādes vidi, tika apskatīti trīs varianti – uzstādīt Laravel un React projektu atsevišķi un izmantot tajos iebūvētas lokālā servera vides lietotnes php artisan serve un npm run, un mariaDB datubāzi izmantojot XAMPP pakotni, Uzstādīt Laravel uz Apache servera pakotnē XAMPP un izmantot React NPM iebūvētu lokālo izstrādes vidi vai arī uzstādīt vidi kā virtuālu konteineri izmantojot Docker vidi.

Pirmais variants, uzstādīt mariaDB datubāzi izmantojot XAMPP pakotni un izmantot PHP artisan serve (laravel) un npm run (react), lai palaistu ietvaros iebūvētās izstrādes vides. Šāda pieeja ir vispopulārākā, tas samazina nepieciešamo konfigurāciju daudzumu, un ir ātri pārnesams uz citu mašīnu, taču ir nepieciešams atsevišķi uzstādīt datubāzi.

Otrais variants ir uzstādīt visu servera puses daļu - Laravel un datubāzi izmantojot izstrādes servera vidi XAMPP. Šī pieeja ir populāra, taču prasa papildus konfigurāciju –

virtuālajiem hostiem. Bieži vien rodas problēmas ar portiem, ja kāda lietotne to jau aizņem, tāpat arī var rasties problēmas ar atļaujām failu piekļuvē, un ir sarežģīti pārnest uz citu mašīnu.

Trešais variants ir Docker virtuālie izstrādes konteineri. Izstrādes vides pārvaldnieks Docker ir lietojumprogramma, ar kuras palīdzību ir iespējams izveidot jebkādu izstrādes vidi izmantojot konteinerus (Docker (2021). Use containers to Build, Share and Run your applications). Konteineris ir virtuālā mašīna, kura ir konfigurēta konkrētas izstrādes vajadzībām. Konteinerim ir sava operētājsistēma un tikai nepieciešamās lietojumprogrammas un pakotnes, dēļ šādas pieejas izstrādes vide ir ļoti ātrdarbīga. Tā kā sakonfigurēta izstrādes vide glabājās konteinerī, to ir vienkārši pārnest uz jebkuru citu datoru, vienīgā prasība ir uzstādīta lietojumprogramma Docker.

Projekta izstrādei tiks izmantots Docker-compose rīks, šis rīks ir viens no Docker lietojumprogrammas rīkiem, kurš ļauj sakonfigurēt izstrādes vidi, kura sastāv no vairākiem konteineriem. Šim nolūkam kalpo YAML konfigurācijas fails, kurā tiek konfigurēti lietotnes pakalpojumi.

Docker-compose ir paredzēts visām vidēm – produkcijas, izstrādes un testēšanas. (Docker (2021). Overview of Docker Compose)

Izstrādes vides konfigurēšanai ir iespējamas divas pieejas:

1. Izveidot konteineri pašam izmantojot Docker konteineru konfigurācijas failu (docker-compose.yml).
2. Izmantot jau iepriekš sakonfigurētu konteineri, kuru var atrast Docker konteineru datubāzē.

Projektā tiek izmantots PHP valodas satvars Laravel 8, veicot izpēti, tika atrasts izstrādātāja “Bitnami” sakonfigurēts Laravel 8 vides konteineris (Docker hub (2021). Bitnami/Laravel image). Šāda izstrādes vides attēla izmantošana, ne tikai atvieglo konfigurāciju, bet arī samazina drošības riskus, tiek veikti uzlabojumi un kļūdu labojumi un šāds veids dod iespēju gūt atbalstu no attēla izstrādātāja vai lietotājiem.

Šādā veidā ievērojami vienkāršāk ir uzstādīt ne tikai visu izstrādes vidi, bet arī satvaru Laravel 8, jo šajā attēlā viss ir sakonfigurēts un iekļauts.

4.5. Datubāze

Tā kā projekta izstrādei tiks izmantots Laravel servera puses satvars, svarīgi izpētīt dokumentāciju, lai uzzināt datubāzu atbalstu.

Laravel 8 oficiālajā dokumentācijā ir norādītas 4 datubāzes, kuras Laravel satvars atbalsta (Laravel documentation (2021). Database: Getting Started):

- MySQL 5.7+;
- PostgreSQL 9.6+;
- SQLite 3.8.8+;
- SQL Server 2017+.

Tā kā izstrādes vide tiks īstenota izmantojot gatavu Bitnami/Laravel docker konteineri, tajā jau ir sakonfigurēta datubāze mariaDB.

MariaDB ir atvērta koda datubāze, kuru ir izstrādājuši MySQL izstrādātāji un tā ir pilnībā savietojama ar MySQL (MariaDB (2021). Documentation).

Tāpat kā MySQL, MariaDB ir iespējams pārvaldīt izmantojot grafisku rīku phpMyAdmin, tas ir labs veids kā pārbaudīt vai datubāzē veidojās pareiza struktūra, un atklūdot nepieciešamības gadījumā, kā arī veikt datubāzes rezerves kopijas.

4.6. Izmantojamie rīki

Autors projekta izstrādei izmantos:

- Visual studio code – bezmaksas koda redaktors, ar lielu papildus moduļu bāzi, kuri palīdz paplašināt redaktora funkcionalitāti, strādājot ar dažādām tehnoloģijām un programmēšanas valodām;
- Ubuntu Linux 20.04 – linux bāzēta operētājsistēma;
- MySQL Workbench – MySQL datubāzes pārvaldīšanas rīks;
- Chrome un Firefox – populārākie interneta pārlūki ar iebūvētiem izstrādātāja rīkiem, kuri palīdz izstrādē, atklūdošanā un dod iespēju pārbaudīt lietotnes darbību uz dažāda izmēra ekrāniem.

5. PROGRAMMATŪRAS INŽENIERIJA - TEORĒTISKĀ DAĻA

Pirms interneta veikala programmēšanas daļas, svarīgi veikt programmatūras inženieriju.

Programmatūras inženierija tiek definēta dažādi, piemēram (Matthew Martin (2021). What is Software Engineering? Definition, Basics, Characteristics) rakstā ir pieminētas šādas pazīstamas definīcijas:

- IEEE 610.12.-1990 standartā tā tiek definēta kā sistematizēta, disciplinēta un aprēķināma pieeja programmatūras izstrādei, darbībai un apkalpošanai;
- Frīdrihs (Fricis) L. Bauers (vācu datorzinātņu pionieris) (J. A. N. Lee (2017). Friedrich (Fritz) L. Bauer) – standarta inženierijas principu izmantošana, kas palīdz izveidot ekonomisku programmatūru, kura ir uzticama un darbojās lietderīgi reālos apstākļos;
- Barrijs Boems (Barry Boehm – amerikāņu programmatūras inženieris) – pielietot praksē zinātņi veidojot kreatīvu dizainu un izstrādājot datorprogrammatūru, kuras sastāvdaļa ir dokumentācija, kura ir nepieciešama programmatūras izstrādei, lietošanai un uzturēšanai.

(Computer Science Department, University of Cape Town (2011). The software crisis Chapter 2. Process and Model.) Liela nepieciešamība pēc programmatūras inženierijas radās 1960. gados, kad daudzi programmatūras izstrādes projekti cieta neveiksmi. Neveiksmes iemesls bija nepareiza izstrādes pieeja, kura tajā laikā bija izveidojusies un plaši izplatīta – rakstīt kodu un labot (“code-and-fix”). Šāda pieeja izskatās aptuveni šādi – programmētāji domā, ka ir sapratuši prasības, notiek programmatūras daļas programmēšana un tad labošana. Šāda pieeja tiek izmantota atkal un atkal kamēr programmatūra netiks pabeigta.

Kā tik noskaidrots, šāda pieeja ir vienkārša taču bieži vien der tikai ļoti maziem projektiem, un var radīt problēmas izstrādājot lielākus projektus. Galvenās problēmas, kuras rodās izmantojot šādu pieeju ir:

- nav iespējams noteikt izstrādes termiņu un budžetu;
- nav novērtēti iespējamie riski un dizaina nepilnības;
- sistēmas nekad netiek pabeigtas;
- tiek kavēti termiņi;
- sistēma var darboties, bet ar kļūdām un neefektīvi;
- šādā sistēmā grūti ieviest izmaiņas un jauninājumus;
- sistēma nepilda tai paredzētos uzdevumus;
- u.c.

5.1. Programmatūras dzīves cikls

Lai izvairītos no augstāk minētajiem šķēršļiem izstrādes laikā, ir nepieciešama programmatūras izstrādes plānošana, kura aptver visu programmatūras dzīves ciklu (Software Testing Help (2021). SDLC (Software Development Life Cycle) Phases, Process, Models.):

- Prasību ievākšana, specifikācija un analīze – šajā fāzē no klienta tiek ievākta visa nepieciešamā informācija par vēlamu produktu. Projektu vadītājs un biznesa analītiķis ievāc informāciju no klienta un analizē – veicot vairākas produkta testēšanas pieejas. Tas ir nepieciešams, lai pēc iespējas labāk saprastu produkta nepieciešamību, tā funkcionalitāti un izvairīties no nepilnībām un aplamībām, pirms virzīt produktu izstrādē.

- Projektējums – šajā fāzē izstrādātāju komanda definē izstrādes pieejas, izmantojamās tehnoloģijas, vizuālo dizainu, funkcijas, moduļus, arhitektūru, datu modeļus, u.c. prasību specifikācijā definētā produkta izstrādei nepieciešamo (Harness Author (2021). Understanding The Phases Of The Software Development Life Cycle.).

- Izstrāde un implementācija – projektējums tiek “pārvērsts” pirmkodā, tiek izstrādāta programmatūra atbilstoši dizainā pieņemtajiem lēmumiem. Izstrādes gaitā neizbēgami rodas sarežģījumi, lēmumu pieņemšanai bieži vien tiek norīkots galvenais izstrādātāis, kurš pieņem lēmumus un vada izstrādes gaitu (Harness Author (2021). Understanding The Phases Of The Software Development Life Cycle.).

- Testēšana – šajā fāzē notiek izstrādāto moduļu un programmatūras testēšana – kura ir nepieciešama, lai izvairītos no kļūdām, pārliecinātos par kvalitāti, ātrdarbību un atbilstību klienta prasībām pirms ieviešanas.

- Programmatūras ieviešana – šajā fāzē programmatūra tiek palaista produkcijā. Visas iesaistītās puses tiek iepazīstinātas ar produkta izstrādes gaitu, ar to, kā tas darbojās. Šajā posmā ir izstrādāts lietotāja ceļvedis.

- Novērošana un apkalpošana – šajā fāzē tiek novēroti produkta parametri – slodze, ātrdarbība. Ja nepieciešams tiek veiktas korekcijas, lai uzlabotu darbību, un novērstas novērotās kļūdas.

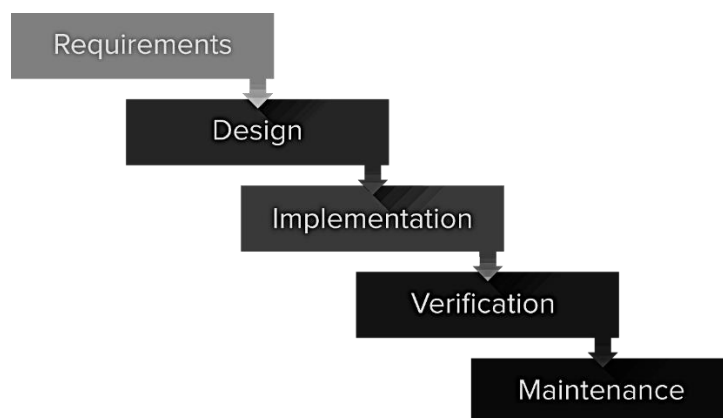
5.2. Programmatūras izstrādes metodes

Iepriekšējā nodaļā minētie programmatūras dzīves cikla posmi ir fundamentāli, taču produkti mēdz būt dažādi pēc lieluma, specifikas, klienta vēlmēm un spējas definēt prasības, tādēļ ir izstrādāti dažādi izstrādes dzīves cikla modeļi. Lai gan tie šķiet atšķirīgi, izstrādes pamata fāzes saglabājās visos modeļos.

5.2.1 Ūdenskrituma modelis

Visvecākais izstrādes modelis, tiek saukts arī par lineāru sekvenciālu modeli. Šī modeļa pamatā ir izstrādes posmi, kur katrs nākamais posms ir atkarīgs no iepriekšējā posma rezultāta un tiek uzsākts tikai tad, kad iepriekšējais ir pabeigts (Harness Author (2021). Understanding The Phases Of The Software Development Life Cycle.).

Pamatā ir bāzes programmatūras dzīves cikla fāzes, kuras lineāri seko viena otrai, apskatāms 5.1. att. Šādam modelim trūkums ir tā atkarības no iepriekšējiem posmiem, progress notiek lineāri vienā virzienā, šāda pieeja rada problēmas labot iepriekšējos posmos radītās kļūdas un prasa ļoti precīzi definētas prasības un piemērotu projektējumu.



5.1. att. Ūdenskrituma izstrādes metode (Harness Author (2021). Understanding The Phases Of The Software Development Life Cycle. Waterfall Model)

Laika gaitā, lai samazinātu šī modeļa nepilnības, tik izstrādāti dažādi tā paveidi – piemēram Ūdenskrituma metode ar pārsedzošām fāzēm vai Sašimī ūdenkrituma modelis, V-modelis un inkrementa modelis.

Ūdenskrituma metode ar pārsedzošām fāzēm, kā varētu secināt no nosaukuma, atšķirās no klasiskā modeļa ar to, ka fāzes pārsedzās – piemēram strādājot pie projektējuma, jau sākās izstrāde, to nepabeidzot (COMPLEX TESTER (2012). SPIRAL, PROTOTYPE, FISH BONE, SASHIMI.) Šāda pieeja tiek izmantota pārsvarā izstrādes laika samazināšanai – jo dažādi speciālisti strādā vienlaicīgi taču var rasties nepieciešamība pārtaisīt. (Sofia (2020). Software development process using waterfall method.)

Ūdenskrituma V-modelis ir izstrādes modelis, kurš var atrisināt problēmu ar laika trūkumu testēšanai. Testēšana būtībā notiek katrā izstrādes dzīves cikla fāzē.

Inkrementa modelī izstrādes process notiek atkārtoti atkal un atkal (Java T Point (2021). Incremental Model.). Vispirms tiek izstrādāta prasību specifikācija, tad balstoties uz to tiek izstrādāta pirmā moduļa versija izejot visus izstrādes posmus. Nākamajā izstrādes ciklā modulis

atkārtoti iziet cauri visām izstrādes fāzēm papildinot tā funkcionalitāti vai uzlabojot salīdzinājumā ar iepriekšējo versiju, attēls

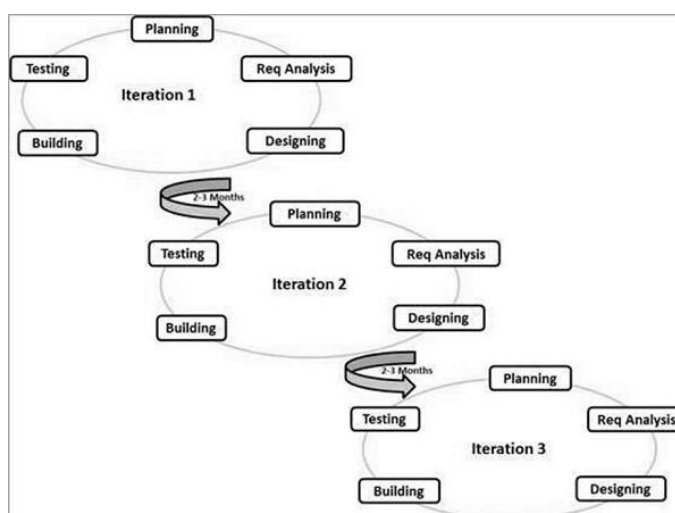
Šāda modeļa priekšrocība ir lielāka uzmanība izstrādes laikā, modulis tiek daudzkārt testēts un izstrādāts līdz pilnībai, taču izstrādes izmaksas ir augstas un ir nepieciešama ļoti laba plānošana.

5.2.2 Agile modelis

Agile izstrādes modelis ir adaptīvās izstrādes veids, kurš salīdzinājumā ar ūdenskrituma modeli ir ļoti dinamisks. Atšķirībā no Ūdenskrituma modeļa, kurš balstās uz smalki izstrādātu dokumentāciju pirms izstrādes un tādā veidā risku paredzēšanu, Agile modelī izstrāde mazāk balstās uz dokumentāciju, bet vairāk uz atgriezenisko saikni ar klientu. Programmatūra tiek sadalīta moduļos, katrs modulis iziet programmatūras dzīves ciklu, pēc kura tiek testēts, labots un saskaņots ar klientu. Pēc katra cikla modulis var tikt papildināts, un izstrāde var pāriet uz nākamajiem moduļiem, kuri tāpat, kā iepriekšējie tiek izstrādāti īsos sprintos (TutorialsPoint (2021). SDLC - Agile Model.).

Agile izstrādes raksturīgākās iezīmes:

- Klientam tiek radīti gatavu moduļu demonstratīvās versijas – tādā veidā tiek labāk uztevertas klienta prasības produktam;
- Klients piedalās izstrādē visos izstrādes posmos;
- Biežas izmaiņas – Agile ir dinamiska izstrādes metode, tādēļ biežas izmaiņas var rasties un tiek ieviestas ātri;



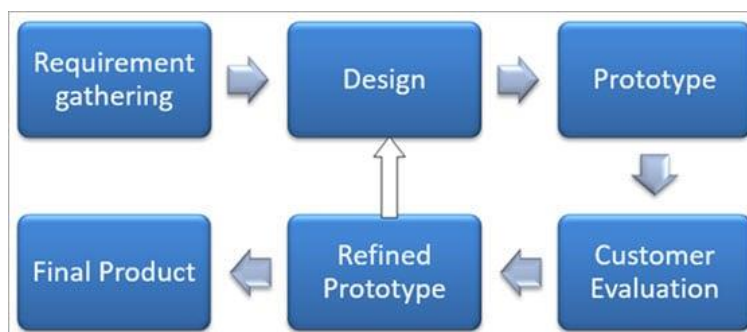
5.2. att. Agile modeļa darbības diagramma (TutorialsPoint (2021). SDLC - Agile Model.)

Lai arī šādai metodei ir daudz priekšrocības, piemēram ātrums, viegla pielāgošanās izmaiņām, maz dokumentācijas un iespēja strādāt ar klientu – tādā veidā būt pārliecinātam par to, ka izstrāda iet pareizajā klientam nepieciešamajā virzienā, tomēr ir arī trūkumi:

- risks izstrādāt programmatūru ar sliktu uzturēšanas, paplašināšanas vai zemu ilgtspēju;
- svarīgi lai klients zin, kas viņam ir nepieciešams, savādāk izstrāde var nonākt nepareizā virzienā;
- dēļ dokumentācijas trūkuma, var rasties problēma nodot sistēmu jauniem izstrādātājiem;
- var rasties daudz individuālas atkarības, dēļ nenoteiktas dokumentācijas;
- projekta veiksmi bieži vien nosaka izstrādes vadītāja spēja kontrolēt izstrādes gaitu, jo šajā modelī tas ir noteicošais faktors;
- vairāk laika tiek patērēts lai pārprogrammētu kādu moduli, nevis lai uzlabotu kopējo projektējumu (dizainu) (Computer Science Department, University of Cape Town (2011). The software crisis Chapter 2. Process and Model.).

5.2.3 Prototipēšanas modelis

Trešais modelis ir prototipēšanas modelis. Šāda veida izstrādē, balstoties uz klienta prasībām ātri tiek uztaisīts funkcionējošs modelis, kuru dod vērtēt klientam. Balstoties uz klienta vērtējumu, tiek izstrādāts jauns prototips vēl un vēl, kāmēr klients akceptē to par derīgu. Kad prototips ir apstiprināts, tas tiek izmantots kā prasību specifikācija un projektējums izstrādājama programmatūrai (Software Testing Help (2021). SDLC (Software Development Life Cycle) Phases, Process, Models.), process apskatāms attēlā 5.3. att.



5.3. att. Protitipēšanas izstrādes modelis (Software Testing Help (2021). SDLC (Software Development Life Cycle) Phases, Process, Models.)

Pēc prototipa saskaņošanas izstrāde norit izmantojot ūdenskrituma izstrādes modeli.

5.3. Prasību dokuments

Kā jau par to liecina nosaukums, prasību specifikācija ir prasību pret izstrādājamo produktu dokumentācija un specifikācija vairākos abstrakcijas līmeņos – klientam, izstrādātājiem, lietotājiem.

Prasību specifikāciju dokuments parasti ietver (Vladimir Sechko, Nadejda Alkhaldi (2021). Your guide to writing a software requirements specification (SRS) document.):

- Produkta vērtības aprakstu – kādu labumu dos programmatūra, vai programmatūra atrisinās problēmu, vai programmatūra paaugstinās tirgus vērtību?
- Biznesa modeļa aprakstu – paskaidro klienta biznesa modeli, kuru atbalstīs iespējamais risinājums. Šajā sadaļā iekļauj organizācijas aprakstu, procesu diagrammas, biznesa funkcijas utt.
- Motivācija – paskaidro, kādēļ klients vēlās izstrādāt šādu programmatūru. Šim punktam jāpalīdz pieņemt lēmumus izstrādes gaitā.
- Funkcionālās un nefunkcionālās prasības – hierarhisks dažādu prasību izklāsts.
- Sistēmas īpašības – iekļauj sevī tādus punktus kā – paplašināmība, uzticamība, drošība, pieejamība.
- Lietotāji – definē potenciālos programmatūras lietotājus – iekļaujot aprakstā vecumu, dzimumu, profesiju, iemaņas, utt.
- Lietošanas gadījumi – apraksta diagrammas veidā soļus, kuri ir nepieciešami kāda rezultāta sasniegšanai.
- Pieņēmumi un ierobežojumi – izceļ projektējuma ierobežojumus – kuri ir svarīgi izstrādes gaitā, un produkta darbībā.
- Pieņemšanas kritēriji – apraksta parametrus pēc kuriem vadīsies apstiprinot gatavu produktu.

5.4. Prasību definēšanas pieejas

Prasību definēšana ir pirmais posms, pirms projekta uzsākšanas. Ir svarīgi, lai abām iesaistītajām pusēm, klientam un izstrādātājiem, būtu skaidrs kāpēc vispār ir nepieciešams šāds produkts, kādas funkcijas tās pildīs klienta biznesā un kā to var sasniegt no programmatūras viedokļa.

Ievācot prasības svarīgi ir pēc iespējas precīzāk definēt nepieciešamo un atteikties liekā, tādēļ šādas informācijas ievākšanai ir iespējamās dažādas pieejas.

Intervijas – intervijai jābūt iepriekš sagatavotai, lai varētu precīzāk noteikt funkcionālās un nefunkcionālās prasības produktam, kā arī ir svarīgi, lai atbildes būtu pilnīgas un klients tādā veidā varētu sniegt plašāku priekšstatu izstrādātājam par vēlamu produktu un biznesa modeli. Intervijās svarīgi ir iekļaut arī potenciālos lietotnes lietotājus.

Aptaujas – aptaujas var aptvērt lielāku potenciālo lietotāju skaitu, taču tas arī rada risku, ka aptauja var nesasniegt pietiekamu respondentu skaitu. Labi sagatavotas aptaujas bieži atklāj

vairāk informācijas par vēlamu produktu, nekā to var pastāstīt klienta projektu vadītājs.

Lietotāju izpēte – tā var būt pasīva vai aktīva. Lietotājus var novērot un tādā veidā izvērtēt un secināt nepieciešamo vai arī novērot un uzdotot jautājumus.

Dokumentu analīze – Efektīvs veids, kā ievākt prasības – var pētīt esošas sistēmas dokumentāciju, piefiksējot trūkumus un uzlabojot esošās ērtās funkcijas, tāpat arī var ievākt informāciju par kļūdu ziņojumiem un sūdzībām.

Prāta vētra un radošās darbnīcas – vērtīgi ir apvienot iesaistītos cilvēkus – lietotājus, klientu, izstrādātājus un veidot prāta vētras vai arī izrunāt konfliktējošas daļas, kopīgiem spēkiem un ar dažādiem viedokļiem nonākot līdz risinājumam.

Izmantošanas gadījumi un scenāriji (lietotāju stāsti) – izmantošanas gadījumos tiek izvirzīti iespējami lietošanas gadījumi noteiktai lietojumprogrammas daļai, taču scenārijos tiek aprakstīts kā sistēma reaģē un izpilda procesus nepieciešamus biznesa modelim. Parasti izpildītājs ir lietotājs un tiek aprakstīta darbība un rezultāts.

Prototipēšana – ātra prototipu izstrāde un prezentēšana klientam ir ļoti populāra mūsdienu izstrādē, tā ļauj klientam uzskatāmi parādīt iespējamo rezultātu un saņemt jau skaidrāku priekšstatu par nepieciešamo, jo bieži vien klients var iztēloties produktu savādāk, vai nemācēt raksturot savas ieceres.

(Jama Software (2021). 11 Requirements Gathering Techniques for Agile Product Teams)

Autors darba izstrādei izvmantos interviju ar nozares speciālistu un esošas sistēmas izpēti.

Prasību definēšana sākās ar viedokļu savākšanu par nepieciešamajiem datiem, funkcionālajām prasībām, nefunkcionālajām prasībām un idejām, kuras var palīdzēt izveidot modeli izstrādājamajai lietotnei. Šim uzdevumam tiks izmantota ideju burbuļu diagramma, kur tiks salikti visi viedokļi.

Nākamais posms ir novērtēt viedokļus, un salikt prioritātes, šim uzdevumam tiks izveidota tabula ar prasībām un prioritātēm, kā arī papildus viedokļa parametri un prasības.

Nākamais posms ir izveidot vienkāršotu modeli, kurš apvienos viedokļus un funkcionālās prasības. Šim uzdevumam tiks izmantota abstrakta diagramma ar scenārijiem.

Pēdējā definēšanas posmā tiks izveidota diagramma kurā tiks attēlots sistēmas apkārtnes modelis, ietverot sistēmas sastāvdaļas, kuras realizēs prasību izpildi.

5.5. Sistēmas prasību specifikācija

Sistēmas prasību specifikācijas posms ļauj precīzāk noteikt funkcionālo un

nefunkcionālo prasību realizēšanas iespējas un datu modelēšana no sistēmas perspektīvas. Lai arī šajā posmā specifikācija notiek no sistēmas viedokļa, prasībām ir jābūt aprakstītām saprotamā valodā gan pasūtītājam, gan izstrādātājiem, tādēļ tiek izmantotas tabulas un shēmas.

Tabulas un shēmas, kuras izmanto aprakstot funkcionālās un nefunkcionālās prasības ir jāunificē viena dokumenta ietvaros, lai dokuments būtu saprotams, un dokumenta izstrādes laikā, ja pie tā strādā vairāki cilvēki, nerastos atšķirības.

Datu modelēšanu var veikt dažādos abstrakcijas līmeņos, šajā posmā piemērotākais būs E-R (entity-relation) modelis vai arī konceptuālais datu modelis.

E-R modelis attēlo datu sistēmas loģiku no datu perspektīvas, modelis sastāv no tādām galvenajām komponentēm (Geeksforgeeks (2021). Introduction of ER Model) kā – būtība, atribūts, attiecība un mantošana (Anita Stivriņa (2020). Programmatūras prasību specifikācija).

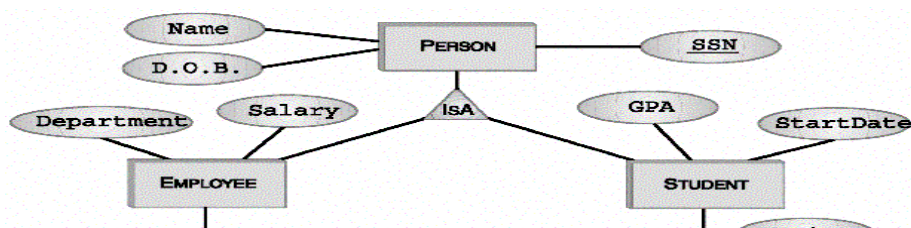
Būtība var būt objekts kuram ir piemēram objekta nozīme fiziskajā pasaulē piemēram – grāmata, māja, darbinieks, vai arī konceptuāla būtība, tāda kā – katalogs, grozs, u.c.

Atribūts ir būtības īpašība vai daļa, piemēram izmērs, kārtas numurs, krāsa, cena.

Attiecība – tāpat kā datubāzu sistēmās norāda uz attiecības veidu starp būtībām, piemēram viens pret vienu, vai viens pret vairākiem utt.

Mantošana – norāda būtības, kuras manto viena otras būtību daļēji – piemēram students var būt – bakalaura, maģistra vai koledžas students.

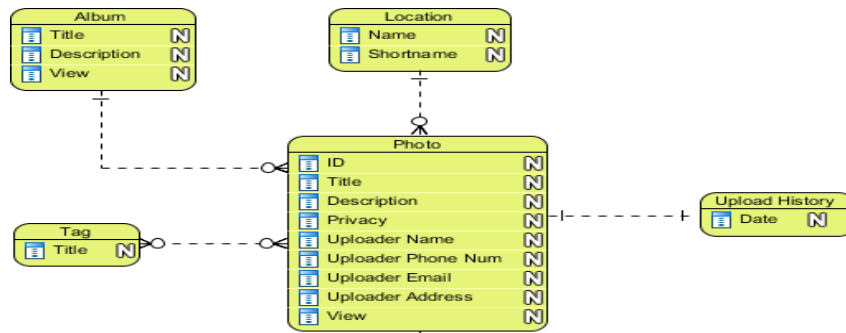
E-R modelis ir apskatāms 5.4. att.



5.4. att. Vienkāršs E-R modelis (Juniata College(2020). Entity Type Hierarchies)

Konceptuālais datu modelis tiek uzskatīts par visabstraktāko no visiem, to bieži izmanto prasību specifikācijas dokumentos dēļ tā, ka tas ir viegli saprotams gan no biznesa loģikas viedokļa, gan arī izstrādājot datu bāzes.

Katra būtība ir tabula, kurā ir tās atribūti un starp tabulām tiek norādītas būtību attiecības. Atšķirībā no datu fiziskā modeļa, lai arī vizuāli tie šķiet līdzīgi, ir tā, ka netiek norādīti ne datu tipi, ne atslēgas, bieži vien modelis var tikt veidots arī bez atribūtiem. Apskatāms 5.5. att. attēlā (Sparx Systems (2020). Conceptual Data Model).



5.5. att. Konceptuāls datu modelis (Visual-paradigm (2021). Conceptual, Logical and Physical Data Model)

Nefunkcionālās prasības ir prasības lietojumprogrammas kopējai darbībai un apkārtnei, tās raksturo piemēram ātrdarbību, drošību, pieejamību, stāstus. Šīs prasības definē sistēmas apkārtni un nosaka tās ierobežojumus.

Funkcionālās prasības apraksta konkrētu funkciju, kuru ir jāizpilda lietojumprogrammai piemēram ir jābūt iespējai pievienot preci, ir jābūt iespējais kategorizēt preci, u.c.

Gan funkcionālās, gan nefunkcionālās prasības ir jādefinē tā, lai tās būtu skaidras, nepārprotamas un pārbaudāmas. Testēšanas fāzē testus veic izmantojot šo iepriekšējo funkciju definīciju, un tām ir jāizpildās, lai prasība tiktu uzskatīta par izpildītu.

Nefunkcionālās prasības parasti definē tabulā – kurā norāda tās nosaukumu un mērvienību, kurā tā tiks mērīta un kā to var pārbaudīt.

Funkcionālās prasību definēšanai ir dažādas pieejas, parasti tās ir funkciju kartiņas, kurā tiek sīki izklāstīta funkcija – ievaddati, izvaddatim apraksts, stāvokļu apraksts, avots, norādes, blakus efekti, u.c., apskatāms 5.6. att.

Jaunas preces pievienošana pirkumam	
Funkcija	Jaunas preces pievienošana pirkumam
Apraksts	Pievieno preci aktuālajam pirkuma čekam
Ieeja	Preces numurs, preces daudzums
Avots	Tastatūra
Izeja	Vaicājums
Norādījums	Ekrāns, pievienot nākamo preci, izveidot čeku
Prasības	Jābūt ievadītam preces numuram no saraksta un daudzumam
Pirms-stāvoklis	Dati nolasīti
Pēc-stāvoklis	Ja nav ievadīts preces numurs – kļūdas paziņojums un atgriežās pie preču saraksta; Ja preces numurs ievadīts pareizi – pieprasa preces daudzumu; Ja preces daudzums nav ievadīts – pieprasa ievadīt daudzumu (neiziet no cikla); Ja preces daudzums ievadīts lielāks par 0 – divas opcijas – Pievienot preci, Izveidot čeku.
Blakus efekti	NAV

5.6. att. Funkcijas apraksts strukturētā valodā. Funkcijas karte (autora veidots ekrānšāviņš)

5.6. Programmatūras projektēšana

Programmatūras projektēšana un projektējuma dokuments apraksta to kā prasības tiks realizētas programmatūrā, šis dokuments ir paredzēts izstrādātājiem, tādāļ tā veidošanā tiek izmantota tehniskāka valoda un pieejas, kā attēlot to, kas ir jāizstrādā.

Projektējuma dokumentā iekļauj:

- Ievadu – ievadā apraksta dokumenta nolūku, definīcijas un saīsinājumus;
- Saistību ar citiem dokumentiem – prasību specifikāciju piemēram;
- Projektēšana – Arhitektūras projektēšana, abstraktā specificēšana, interfeisa projektēšana, komponentu projektēšana, datu struktūru projektēšana algoritmu projektēšana

- Detalizētais projektējums

Šī projekta ietvaros tiks izmantotas tādas projektēšanas metodes kā:

- Lietotāja saskarnes projektēšana izmantojot tiešsaistē pieejamos grafiskos rīkus tādus kā draw.io.

- Datu fiziskā modeļa izveidošana.
- Algoritmu attēlošana blokshēmās.

5.7. Testēšanas dokumentācija

Testēšanas dokumentāciju izstrādā testējot jau izstrādātos moduļus. Testēšana palīdz gan izstrādātājiem, gan klientam pārliecināties par to, ka prasības produktam, kuras ir definētas prasību specifikācijā, izpildās. Tāpat testēšanas fāze ļauj tikt vaļā no programmas kļūdām pirms programma tiks palaista aprītē. Kvalitatīva testēšana samazina papildus izmaksu risku klientam, jo vēlākās stadijās tas var prasīt laiku un ievērojamus finansiālus zaudējumus.

Testēšanas veidi:

- Vienībtestēšana – šis veids paredz, ka testēta tiek viena lietotnes vienība. Vienība var būt ļoti maza – komponente, funkcija vai klase.

- Integrācijas testēšana – izstrādātāji kombinē dažādus programmatūras moduļus kopā un testē. Šāds tests ļauj pārbaudīt to, vai datuplūsma ir pareiza un ir iespējama nākamā fāze – sistēmas testēšana.

- Sistēmas testēšana – šajā fāzē tiek pārbaudīta visa sistēma, šajā posmā izstrādātāji novērtē vai funkcionālās prasības izpildās. Papildus šajā posmā ir iespējams novērtēt sistēmas parametrus – slodzi, ātrdarbību.

- Lietotāju atzinuma testēšana – šajā fāzē sistēma tiek nodota testēšanai klientam un lietotājiem, lai gūtu apstiprinājumu un izmaiņu pieprasījumus, ja tādi ir nepieciešami (PerformanceLab (2021). Importance of Software Testing in SDLC).

Testēšanai ir arī daudz dažādi modeļi, piemēram šim projektam autors izmantos ,melnās kastes testēšanas metodi – testi tiek balstīti zinot tikai programmas specifikāciju, analizēti tiek tikai ievaddati un izvaddati.

5.8. Lietotāja ceļvedis

Lietotāja ceļvedis, kā to var noprast pēc nosaukuma, ir paredzēts iepazīstināt lietotājus ar produktu. Lietotāju ceļveži parasti ir orientēti uz divām lielām kategorijām:

- gala lietotājs;
- sistēmas administrātori.

Gala lietotāju dokumentācija palīdz lietotājam iepazīties ar sistēmu un tās funkcijām. Ceļvedim ir jābūt sastādītam atbilstoši gala lietotāju prasmēm, un jābūt iekļautām visām tam nepieciešamajām produkta funkcijām.

Ceļvedi var sadalīt sekojošās nodaļās:

- Ātra darba uzsākšana (Quick start guide) – paredzēts, lai lietotājs gūst vispārējo priekšstatu par sistēmu, un ir iekļautas pamata darbību apraksts darbam.
- Pilnais ceļvedis – iekļauj visu nepieciešamo darbam ar sistēmu – uzstādīšana, sistēmas prasības, katras iespējamās lietotnes funkcijas izsmelšu aprakstu ar attēliem, ieteikumiem un piemēriem.
- Problēmu risināšanas gids – apraksta iespējamās kļūdas un to iespējamās cēloņus un risinājumus.

No lietotāja ceļveža ir ļoti atkarīga lietotnes lietojamība, tā ir jāveido vienkārša saprašanai un loģiski strukturēta. Lietotāja ceļvedis var ļoti ietekmēt lietotāju apmierinātību ar produktu un kopējo produkta veiksmīgumu. (Altexsoft (2020). Technical Documentation in Software Development: Types, Best Practices, and Tools)

6. NOZARES PĒTĪJUMA REZULTĀTS

Grāmatu izdevēju nozares padziļinātai izpētei tika izmantota aptauja un intervija ar nozares pārstāvjiem, kā arī esošā internetveikala www.veikals.avotsabc.lv izpēte un darbs pie tā uzturēšanas.

Diemžēl aptaujas respondentu skaits ir pārāk mazs, lai izprastu uz šo brīdi esošo digitālo situāciju nozarē. Respondentu skaits ir 4, tādēļ aptaujas rezultātu dati netiks ņemti vērā.

Taču intervijā, kura iekļāva gan sagatavotus jautājumus gan saruna ar nozares pārstāvjiem, tika noskaidrots, ka:

- Pēc Centrālās statistikas pārvaldes datiem, 5811 NACE kodam atbilst ap 120 grāmatu izdevēji ik gadu, taču šo skaitu veido arī dažādas organizācijas, kas izdod kaut vai vienu grāmatu gadā un grāmatu izdošana nav viņu pamatnodarbošanās. Grāmatu izdošana kā pamatnodarbošanās bija ap 50 uzņēmumiem, covid krīzes laikā šis skaits ir samazinājies.

- Absolūts tirgus līderis izdevniecības nozarē ir izdevniecība “Zvaigzne ABC”, grāmatu tirdzniecības nozarē – “Zvaigzne ABC” un “Jānis Roze”. Nākamie pēc vidējās tirgus daļas datiem (2009.-2016. gads) ir izdevniecības:

- Jāņa Sēta ~1,4 milj.
- Lielvārds, RaKa ~ 1-1.4 milj.
- Egmont Latvia ~1,2 milj.
- Jumava Group ~0.7 milj.
- Kontinents ~0.5 milj.
- Apgāds Jānis Roze ~ 0.4 milj.
- Dienas Grāmata ~0.35 milj.
- Pētergailis ~0.3 milj.

- Grāmatu izdevēji pārsvarā tirgo savos interneta veikalos kam tādi ir vai uz vietas no birojiem, bet pārsvarā realizē lielajos grāmatnīcu tīklos (Jānis roze, zvaigzne, valter un rapa, globuss) vai grāmatu pārdošanas platformās www.buki.lv; virja.lv u.c.

- Latvijas Grāmatizdevēju asociācija aizstāv izdevēju intereses valsts institūcijās, organizē izdevēju dalību nacionālas un starptautiskas nozīmes nozaru izstādēs. Šī gada asociācijas veikums – panāca, ka no 2022. gada 1. janvāra gan drukātam grāmatām, gan elektroniskā formāta grāmatām (e-grāmatām un audiogrāmatām) tiks piemērota samazinātā 5% PVN likme. “sis ir viens no faktoriem kas ietekmē grāmatas cenu. 2021. gadā drukātām grāmatām PVN likme ir 12%, e-grāmatām 21%.

Kā tika noskaidrots sarunā ar izdevniecības “AVOTS” pārstāvjiem, lielākais kavēklis interneta risinājumu ieviešanā un attīstībā, tas ir nozarē strādājošo digitālās prasmes un

informācijas trūkums par iespējamajiem veidiem, digitalizēt uzņēmuma darbību. Nozarē trūkst jaunu speciālistu, kuri varētu modernizēt esošās sistēmas un izstrādāt jaunas e-komercijas pieejas un risinājumus.

Iepazīstoties ar izdevniecības “AVOTS” izveidoto risinājumu, izrādījās, ka interneta veikals tika izveidots pašu darbinieku spēkiem ar minimāliem finanšu ieguldījumiem izmantojot atvērtā koda e-komercijas tīmekļa vietņu izstrādes platformu Opencart.

Ieviešot šajā interneta veikalā tādu svarīgu moduļus, kā piegādes modulis Omniva un apmaksas modulis Swedbank banklink, autors iepazinās ar šo platformu, kā rezultātā tika atklāts, ka šis ir ērts risinājums standartizēta interneta veikala izstrādē, taču tāpat kā līdzīgos gatavos risinājumos rodās modernizēšanas un individuālas pielāgošanas ierobežojumi. Interneta veikala administrācijas daļa ir salīdzinoši grūti uztverama un nav specializēta konkrētas nozares vajadzībām.

Bieži rodas problēmas vajadzīgo risinājumu ieviešanai, jo modulim ir jābūt izstrādātam konkrētai platformai, un lai arī lielākie pakalpojumu sniedzēji tādi kā Swedbank vai Omniva ir izstrādājuši moduļus konkrētai platformai, tāpat pielāgojot rodās problēmas un ierobežojumi.

Opencart platformai nav pieejami plaši dizaina pielāgošanas rīki, tādēļ, dizains ir viens un pielāgošana var prasīt daudz pūļu vai arī būt neiespējama.

Lapas optimizācija ir visai ierobežota un ātrumu reitings piemēram google platformā ir bieži ļoti zems.

7. PROGRAMMATŪRAS INŽENIERIJA - PRAKTISKĀ DAĻA

Izstrādājamā programmatūra – Tīmekļa vietne grāmatu izdevējiem;

Izstrādājamie moduļi:

- Lietotāja saskarne – Administrācijas panelis;
- Modulis – Autorizācija;
- Modulis – Katageorijas;
- Modulis – Produkti (Grāmatas);

Iekļāutie dokumenti:

- Prasību definīcija tīmekļa risinājumam;
- Prasību specifikācija;
- Projektējums;
- Izstrāde;
- Testēšanas dokuments (1. pielikums);
- Lietotāja ceļvedis (2. pielikums).

7.1. Prasību definīcija tīmekļa risinājumam

Prasības programmatūras izstrādei tika ievāktas galvenokārt izmantojot intervijas ar nozarē strādājošajiem un esoša internetveikala pārvaldīšanas sistēmas izpēti un darbību novērošanu darbinieku darbam ar to. Nozares izpēte izmantojot aptauja nesniedza vērtīgus rezultātus, dēļ ļoti maza atbilžu skaita.

Ņemot vērā intervijas rezultātu tika noskaidrots, ka ļoti mazam daudzumam grāmatu izdevēju Latvijā ir savs interneta veikals, un kā autors uzzināja no veiktajām intervijām

Tīmekļa vietnes grāmatu izdevējiem galvenais mērķis ir izveidot atbilstošu nozares prasībām interneta vietni ar e-komercijas (interneta veikala) bāzes funkcionalitāti, kura būs pieejama maziem uzņēmumiem. Šādu tīmekļa vietni varēs viegli pielāgot uzņēmumam – definēt logo, nosaukumu, tekstus, u.c., ja nepieciešams pielāgot dizainu.

Tīmekļa vietnes e-komercijas modulim ir jābūt aprīkotam ar grāmatu tirgošanai nepieciešamajiem atribūtiem, vienkāršu iespēju pievienot preces un sadalīt tos kategorijās, veikt uzskaiti un iespēju meklēt. Ir jābūt iespējai apstrādāt pasūtījumus – klientam veikt pasūtījumus un saņemt iespēju izvēlēties ērtāku piegādes un apmaksas veidu, taču uzņēmumam redzēt pasūtījumus, mainīt pasūtījuma statusu un informēt klientu par pasūtījuma gaitu.

Izņemot interneta veikala funkciju, tīmekļa vietnei ir jāpilda uzņēmuma vizītkartes un saziņas ar klientu funkciju – uzņēmuma informācija, logo, jaunumi. Ļoti svarīgi lai tīmekļa

vietne ir labi pielāgota priekš SEO un Google Adwords.

Tīmekļa vietnes pārvaldīšanas sistēmai jābūt piemērotai darbam darbiniekam ar bāzes datora lietošanas zināšanām.

Sistēmai ir jābūt ātrdarbīgai un optimizētai lēniem un dārgiem interneta savienojumiem – piemēram mobīlajās ierīcēs.

Dizainam ir jābūt nepārsātinātam, ērtam un pielāgotam darbībai uz maziem ekrāniem (mazākais ierīces platums 320px)

Klienta datu drošībai ir jābūt programmatūras arhitektūras pamatā, kā arī ir jābūt aizsargātai tīmekļa vietnes satura veidošanas jeb Admin daļai.

Izstrādātajai programmatūrai ir jābūt viegli izplatāmai, un viegli uzstādāmai neatkarībā no operētājsistēmas.

Programmatūrā ir jābūt paredzētai iespējai to papildināt un modificēt. Programmatūras pirmkodam jābūt brīvi pieejamam, kā arī izstrādātam izmantojot brīvi pieejamas bibliotēkas un atvērtā koda moduļus un pakotnes.

Izstrādātajam ir jāsniedz tehniskais atbalsts programmatūras pielāgošanai un kļūdu labošanai, kā arī jāizveido lietotāja ceļvedis, atsevišķos gadījumos jāsniedz apmācība darbam ar programmatūru.

Sistēmas izstrāde tiks īstenota izmantojot Agile izstrādes metodoloģiju. Produkts tiks sadalīts moduļus un izstrāde – izstrādes ciklos. Katra cikla sākumā tiks izstrādāta prasību specifikācija izstrādājamajiem moduļiem – iekļaujot viedokļus un saliekot prioritātes ņemot vērā atkarības starp izstrādātajiem moduļiem un secību, kādā tie ir jāizstrādā.

7.2. Prasību specifikācija

7.2.1 Administrācijas panelis

Administrācijas panelis ir vietne, kura ir paredzēta interneta veikala darbībai un satura pārvaldīšanai tīmekļa vietnē.

Viedokļu burbuļu diagramma, apskatāma 7.1. att., abstrakti parāda prasības, kuras ir nepieciešamas administrācijas panelim



7.1. att. Viedokļu burbuļu diagramma (autora veidota diagramma)

Virspusējo viedokļi par administrācijas paneli kopumā, nosaka, kādas ir bāzes prasības administrācijas panelim, un tās ir:

- lietotāja saskarne – vienkāršs dizains, nekā lieka, ar intuitīvu pārskatāmu izvēlni, taču ir jāparedz mobīlā versija;
- izvēlnes – preces, kategorijas, klientu sadaļa, pasūtījumu sadaļa, statistikas sadaļa;
- kategoriju un apakškategoriju pievienošana;
- preču - pievienošana, kategorijas piešķiršana un uzskaitē;
- iespēja labot darbinieku pieļautās kļūdas;
- piekļuves ierobežošana ar paroli.

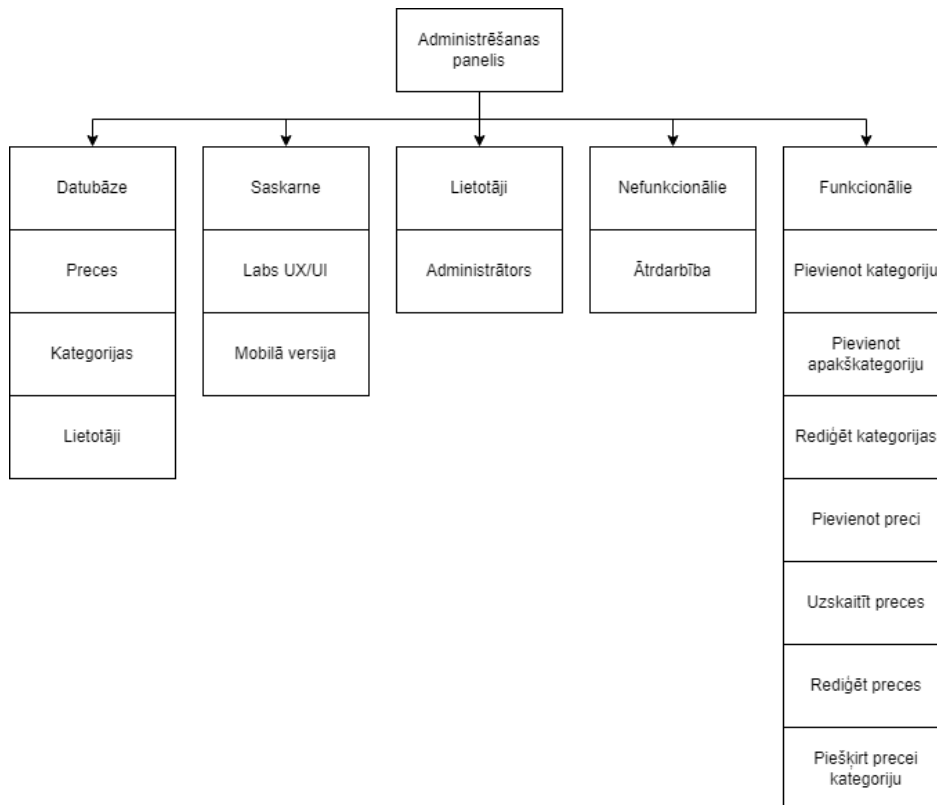
Pirmā izstrādes cikla laiks – 1 mēnesis, 01.12.2021. – 31.12.2021. Ņemot vērā izstrādes cikla termiņu, tabulā 3. pielikumā, ir uzskaitīti 1. izstrādes ciklā realizējamie viedokļi. Prioritāšu kolonnā tiks norādīta konkrētā viedokļa realizēšanas prioritāti šajā izstrādes ciklā.

Kā izriet no tabulā apkopotās informācijas – izstrādes pirmajā ciklā, prioritāri ir izstrādāt lietotāja saskarni, autorizācijas moduli, kategoriju pārvaldīšanas moduli un preču pārvaldīšanas moduli.

Vidējā prioritāte ir noteikta viedokļiem – saskarnes pielāgošana maziem ekrāniem – izstrāde tiks veikta ņemot vērā mazus ekrānus, taču precīza pielāgošana notiks vēlākos izstrādes ciklos. Kategoriju dzēšanas iespējas tik novērtētas datu modeļa plānošanas stadijā iespējams, ka kategorijas nevarēs dzēst.

Zema prioritāte tika noteikta viedokļiem un funkcijām, kuras ir atkarīgas no citu moduļu ieviešanas – šādas prasības ir statistikas, klientu bāzes un pasūtījumu daļas izstrāde.

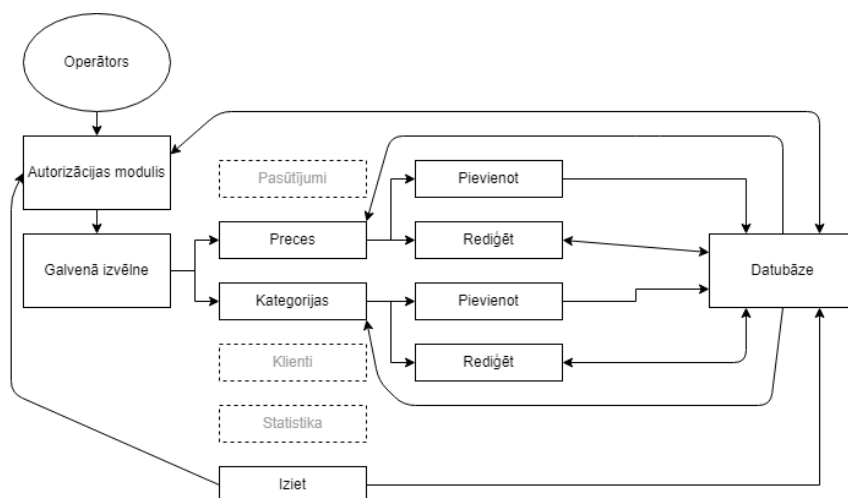
1. Izstrādes ciklā realizējamie viedokļi tika izskatīti sīkāk un tie ir apskatāmi strukturētā viedokļu diagrammā, 7.2. att. Šajā diagrammā viedokļi ir strukturētā veidā sadalīti atsevišķos atzaros, kuri parāda funkcionālās prasības, nefunkcionālās prasības, datubāzes prasības un lietotājus.



7.2. att. Strukturēta viedokļu diagramma (autora veidota diagramma)

Strukturētu viedokļu diagramma atklāj arī funkcionālās un nefunkcionālās prasības izstrādājamajiem moduļiem, 4. pielikumā, tabulā ir uzskaitītas funkcionālās un nefunkcionālās prasības.

Lai izprastu administrēšanas saskarnes darbības principu, tika izveidota diagramma, kurā ir attēlotas saiknes un atkarības starp izstrādājamajiem moduļiem. Vienkāršots sistēmas modelis ir apskatāms 7.3. att.



7.3. att. Vienkāršots sistēmas darbības modelis (autora veidota diagramma)

Kā redzams 7.3. att., lai piekļūtu sistēmai, operātoram ir jāveic autorizācija, kuras laikā notiek ierakstīšanās datu nosūtīšana datubāzei.

Ja administrātora dati datubāzē ir atrasti un ir korekti, operātors piekļūst galvenajai izvēlei.

Galvenajā izvēlnē navigācijas joslā ir pieejami 6 ceļi:

- pasūtījumi (tiks izstrādāts vēlākos izstrādes ciklos);
- preces;
- kategorijas;
- klienti (tiks izstrādāts vēlākos izstrādes ciklos);
- statistika (tiks izstrādāts vēlākos izstrādes ciklos);
- iziet - izrakstīties no sistēmas;

Izvēlne **Preces** – attēlo visu preču sarakstu – datus saņem no datubāzes.

Preces->Rediģēt – izvēlēto preci no saraksta ļauj rediģēt – saņem datus no datubāzes un sūta izmainītos.

Preces->Pievienot – jaunas preces pievienošanas forma – sūta datus uz datubāzi.

Izvēlne **Kategorijas** – attēlo visu kategoriju sarakstu – datus saņem no datubāzes.

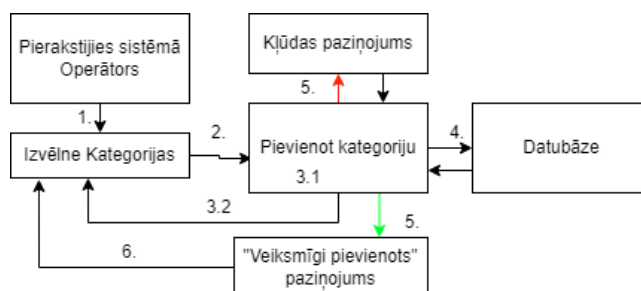
Kategorijas->Rediģēt - izvēlēto kategoriju no saraksta ļauj rediģēt – saņem datus no datubāzes un sūta izmainītos.

Kategorijas->Pievienot – jaunas kategorijas pievienošanas forma – sūta datus uz datubāzi.

Iziet – izrakstās no sistēmas – sūta pieprasījumu uz datubāzi un pāradresē uz pierakstīšanās skatu.

Lietotāju stāstu diagrammas ļauj iepazīties ar realizējamo funkciju izpildes cikliem, ir norādīts uzdevums, ceļš tā izpildei un iespējamo kļūdu apstrāde.

Attēls 7.4. att. parāda ceļus, ar kuriem saskarās pierakstījies sistēmā operātors, pievienojot jaunu kategoriju.

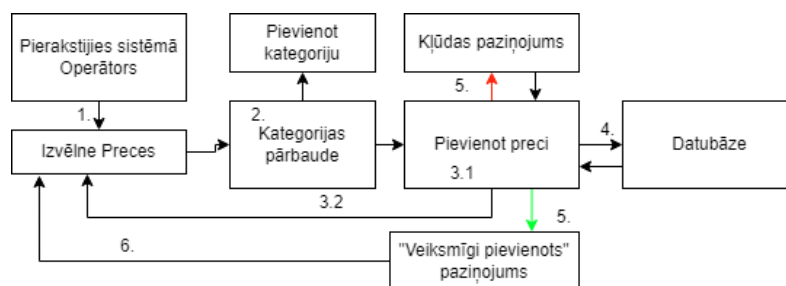


7.4. att. Jaunas kategorijas pievienošanas scenārijs (autora veidota diagramma)

1. pierakstīties operātors ar peles klikšķi pāriet uz izvēlni kategorijas;
2. nospiežot pogu “Pievienot” – pāriet uz jaunas kategorijas pievienošanas formu;
3. atverās kategorijas pievienošanas forma ar nepieciešamajiem datiem:
 - 3.1. aizpilda formu;
 - 3.2. nospiež pogu “Atpakaļ” – pāriet uz kategoriju izvēlni (sarakstu);
4. formas dati tiek nosūtīti uz datubāzi;
5. veiksmīgas saglabāšanas rezultātā tiek atgriezts paziņojums “Veiksmīgi pievienots”, neveiksmes gadījumā – kļūdas paziņojums un paliek atvērta forma – labojumiem;
6. pāriet uz izvēlņu kategoriju.

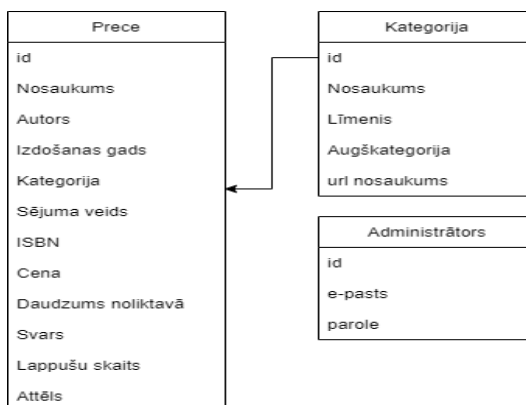
Kategorijas labošanai, ir piemērojams identisks scenārijs izņemot punktu 2. Lai labotu kategoriju, klients izvēlās attiecīgo kategoriju tabulā un uzklikšķina uz tās tabulā.

Attēlā 7.5. ir attēlots jaunas preces pievienošanas scenārijs, tas ir līdzīgs kategorijas pievienošanai, izņemot to, ka uzspiežot pievienot jaunu preci, tiek pārbaudīts, vai datubāzei ir pievienota kaut viena kategorija, pretējā gadījumā, lietotājs tiek novirzīts uz kategorijas pievienošanas logu ar informatīvu paziņojumu – “Nepieciešams pievienot vismaz vienu kategoriju”.



7.5. att. Preces pievienošanas scenārijs (autora veidota diagramma)

Datu modelis ir attēlots konceptuālā modeļa formā. Kā 7.6. att. redzams, kopumā ir 3 būtības – Prece, kategorija un Administrātori.



7.6. att. Konceptuālais datu modelis (autora veidota diagramma)

Kā redzams administrātoru būtībai nav saiknes ar pārējām divām, turpretī prece ir saistīta ar kategoriju, vai vairākām. Šajā datu modelī ir arī norādīti nepieciešamie būtību atribūti:

- Precei – ID, Nosaukums, Autors, Izdošanas gads, Kategorija, Sējuma veids (cietie/ mīkstie vāki), ISBN (unikāls grāmatu identifikators), Cena, Daudzums noliktavā, svars, Lappušu skaits un Attēls vai vairāki.
- Kategorijai – ID, Nosaukums, Līmenis, Augšskategorijas nosaukums un Url nosaukums.
- Administrātoram ir nepieciešams tikai epasts, parole un ID.

7.3. Projektējums

Projektējuma mērķis, balstoties uz prasību specifikāciju 7.2. nodaļā, definēt prasību realizēšanas iespējas.

Šajā dokumentā tiks veikta:

- Uzskaitītas izmantojamās tehnoloģijas;
- Lietotāja saskarnes modelēšana;
- Datu dekompozīcija;
- Algoritmu apraksts.

7.3.1 Izmantojamās tehnoloģijas

Administrācijas paneļa izstrādē tiks izmantotas šādas tehnoloģijas un programmēšanas valodas:

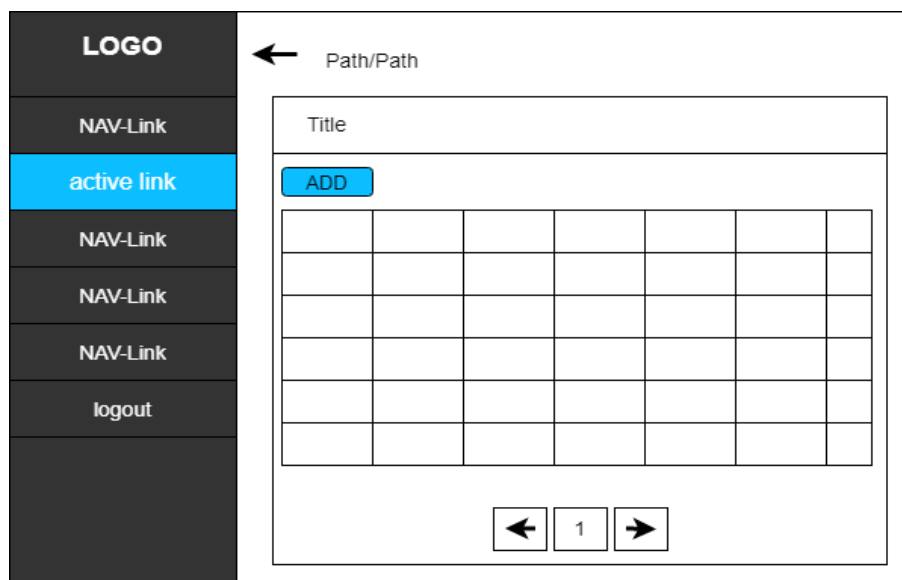
- **Izstrādes vide** – Docker
- **Servera pusē:**
 - PHP 7.4
 - Laravel 8
 - Laravel – modulis autorizācijai Sanctum
- **Klienta pusē:**
 - Typescript
 - CSS
 - React.js (bibliotēka)
 - ANTD design (UI bibliotēka)
- **Datubāze** – MariaDB

7.3.2 Lietotāja saskarnes modelēšana

Lietotāja saskarne ir salīdzinoši vienkārša, tās dizainā ir ņemtas vērā izvēlētās lietotāju

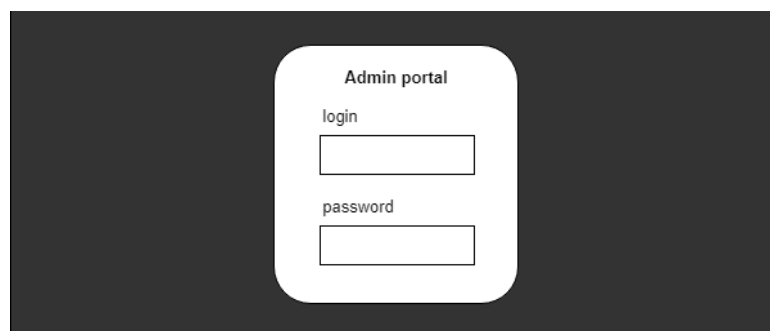
saskarņu veidošanas bibliotēkas pamata krāsās – tumši pelēka, balta, zilas pamata pogas un sarkanas dzēšanas pogas.

Attēlā 7.7. att., ir apskatāms lieliem (datora) ekrāniem paredzētais dizains, atvērums ar tabulu. Tabulās, uz doto izstrādes brīdi, tiks parādīts preču un kategoriju saraksts. Tāpat augšējā daļā ir parādīts atvēruma nosaukums, kurā patreiz atrodās lietotājs. Sānu navigācijas joslā ir izcelta sadaļa, kurā atrodās lietotājs.



7.7. att. Lietotāja saskarne, atvērums ar tabulu. (autora veidots)

Attēlā 7.8. att., ir apskatāma ierakstīšanās sistēmā forma. Patreiz, šajā izstrādes posmā, reģistrācijas forma nav paredzēta.



7.8. att. Ierakstīšanās sistēmā forma (autora veidots)

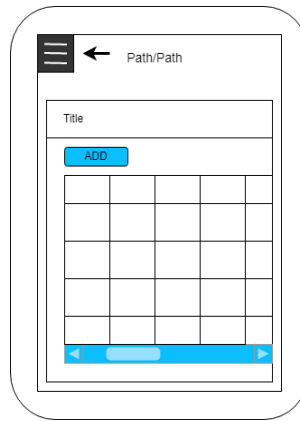
Attēlā 7.9. att., ir redzams formas atvērums lieliem ekrāniem. Ievades lauki ir vienāda platuma, rindā viens aiz otra. Ievades lauku nosaukumi atrodās virs ievades lauka. Saglabāšanas poga atrodās kreisajā apakšējā stūrī. Ja forma tiek izmantota kā rediģēšanas forma, tad papildus ir pieejama poga “Dzēst” – labajā apakšējā stūrī. Kategorijas rediģēšanas formai apakšā ir produktu saraksts, kuri pieder šai kategorijai.

7.9. att. Formas atvērums izskats (autora veidots)

Versija maziem ekrāniem, no 320px platumā, sānu navigācijas josla tiek paslēpta, atstājot pogu, kura atver sānu joslu, vai aizver. 7.10. att., ir redzams formas atvērums ar parādītu un paslēptu sānu navigācijas joslu. Ievades lauki formai izkārtoti viens zem otra, vidēja izmēra ekrāniem lauku daudzums rindā var būt atšķirīgs, atkarībā no platumā, saglabājot funkcionalitāti.

7.10. att. Formas atvērums uz maziem ekrāniem (autora veidots)

Atvērumos ar tabulām, tabulas ir ritināmas uz sāniem, atvērums apskatāms 7.11. att.



7.11. att. Tabulas atvērums uz maziem ekrāniem (autora veidots)

7.3.3 Datu dekompozīcija

Izstrādājamajos moduļos ir pavisam 3 datu vienumi:

1. Lietotājs;
2. Produkts;
3. Kategorija.

Tabulās – 1., 2., 3. tabula, ir uzskaitīti atribūti, kuri ir nepieciešami vienumiem, to datu tips un atslēgas, ja ir.

1. tabula

Administrātoru atribūtu tabula

Administrātors			
Atribūts	Nosaukums	Datu tips	Papildus informācija
id	id	bigInteger	PK, unique
e-pasta adrese	email	string	unique, required
parole	password	hash	unique, required

Veidojot datubāzi, jāizveido ieraksts tabulā Admin – lai pievienotu lietotāju.

2. tabula

Kategorijas atribūtu tabula

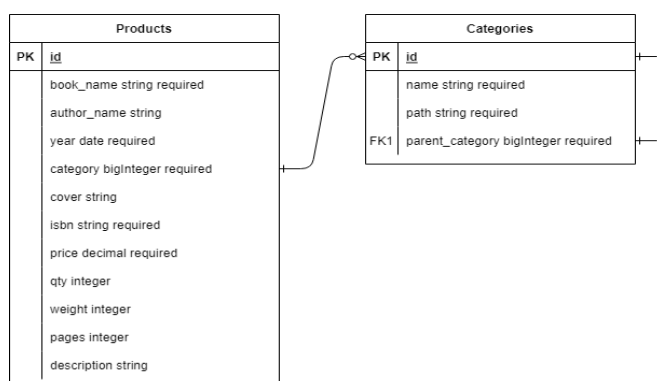
Kategorija			
Atribūts	Nosaukums	Datu tips	Papildus informācija
id	id	bigInteger	PK, unique
Nosaukums	name	string	required
Augš kategorija	parent_category	bigInteger	FK, required
Nosaukums priekš url	path	string	required

Veidojot datubāzi jāizveido pamatkategorijas ieraksts kategoriju tabulā, zem kuras veidot visas pārējās apakškategorijas.

3. tabula

Produkts			
Atribūts	Nosaukums	Datu tips	Papildus informācija
id	id	bigInteger	PK, unique
Nosaukums	book_name	string	required
Autors	author_name	string	
Izdošanas gads	year	date	required
Kategorija	category	bigInteger	FK, required
Sējuma veids	cover	string	
ISBN numurs	isbn	string	required
Cena	price	decimal	required
Daudzums	qty	integer	required
Svars	weight	integer	
Lappušu skaits	pages	integer	
Apraksts	description	string	

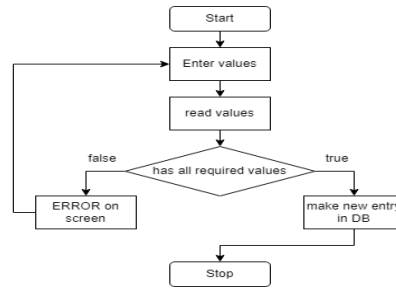
Administrātoru tabula ir atsevišķa tabula, kurai nav relācijas ar citām izstrādātāja veidotām tabulām. Kategoriju un produktu tabulu relācijas ir parādītas 7.12. att., kurā ir redzams, ka kategoriju tabulai ir relācija uz sevi, tādā veidā var noteikt apakškategorijas.



7.12. att. E-R diagramma produktiem un kategorijām (autora veidots)

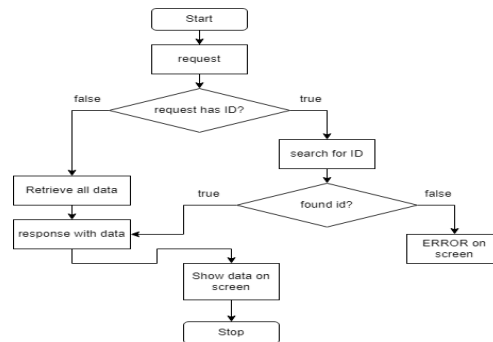
7.3.4 Algoritmu apraksts

Algoritms produktu vai kategoriju izveidošanai – 7.13. att.



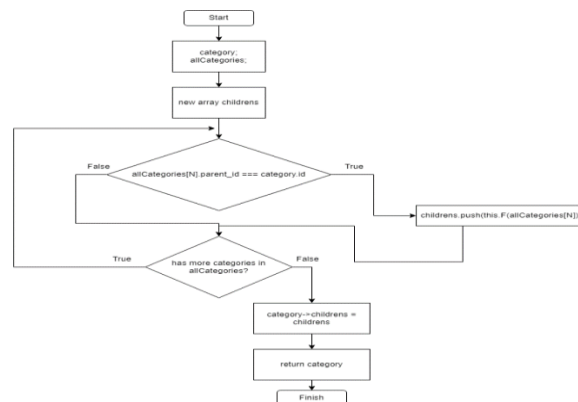
7.13. att. Algoritms produktu un kategoriju pievienošanai (autora veidots)

Algoritms kategoriju vai produktu izgūšanai no datubāzes, kurš ir apskatāms - 7.14. att. Pārbauda vai pieprasījumā ir ID numurs, ja nav, tad atgriež visus ierakstus, ja ir tad tikai ar konkrēto ID numuru datubāzē.



7.14. att. Algoritms kategoriju vai produktu izgūšanai no datubāzes (autora veidots)

Algoritms kategoriju koka izveidošanai, no datubāzē esošiem ierakstiem, apskatāms 7.15. att.

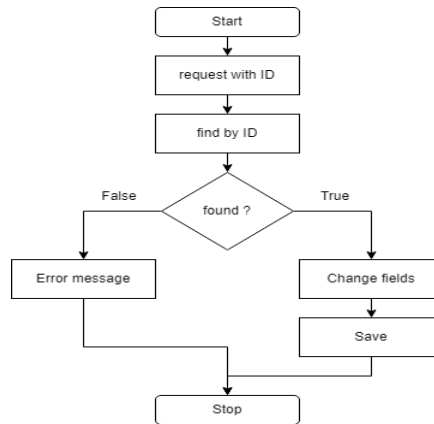


7.15. att. Rekursīvs algoritms kategoriju koka izveidošanai (autora veidots)

Kategoriju koka algoritma sākuma parametri ir kategorija un visu kategoriju masīvs. Funkcijas sākumā tiek izveidots masīvs kategorijas bērnu kategoriju glabāšanai. Algoritma beigās tiek atgriezts modificēts kategorijas objekts ar pievienotu bērnu masīvu. Algoritmā cikls iet cauri visu kategoriju masīvam pārbaudot vai kāda no kategorijām nav pārbaudāmās

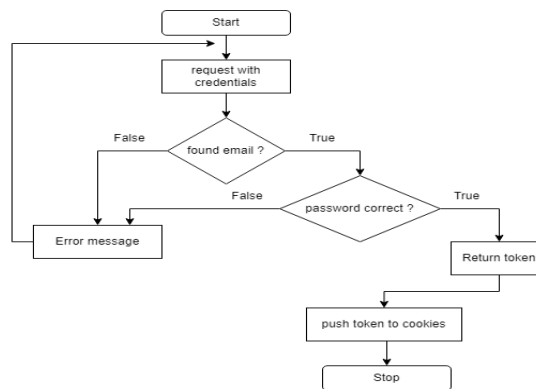
kategorijas apakškategorija. Ja ir, tad masīvam childrens tiek pievienots šīs pašas funkcijas, kurai parametrs jau ir šī apakškategorija, rezultāts.

Kategorijas vai produkta rediģēšanas algoritms apskatāms 7.16. att.



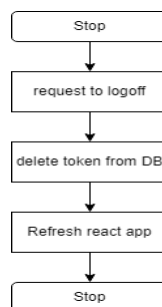
7.16. att. Produkta vai kategorijas rediģēšanas algoritms (autora veidots)

Ierakstīšanās sistēmā algoritms apskatāms 7.17. att.



7.17. att. Ierakstīšanās sistēmā algoritms (autora veidots)

Front-end pusē ir nepieciešams algoritms kurš seko tam, vai lietotājs ir ielogojies, šim nolūkam pie katras react aplikācijas atsvaidzināšanas notiek pārbaude, vai sīkfailos ir šis tokens, kuru saņem no DB ierakstīšanās brīdī. Izrakstīšanās brīdī tiek nosūtīts pieprasījums izrakstīties, tiek izdzēsts tokens no DB un pēc tam atsvaidzināta react lietotne, apskatāms 7.18. att.



7.18. att. Izrakstīšanās no sistēmas algoritms (autora veidots)

Galvenie algoritmi darba uzsākšanai ir definēti, korekcijas un papildus algoritmi tiks veidoti izstrādes gaitā.

Daudzi algoritmi, kuri ir saistīti ar ierakstu izgūšanu no datubāzes te netiek aprakstīti, jo ir iebūvēti ietvaros Laravel un React, kā arī īstenoti citās izmantojamās bibliotēkās, kā piemēram Sanctum – autorizācijas daļas pārvaldīšanai.

7.4. Izstrāde

7.4.1 Izstrādes vides izveidošana

Laravel projektam - tika izveidota projekta direktoriya un lejupielādēts Docker konfigurācijas fails izmantojot aprakstā norādīto komandu : `$ curl -LO https://raw.githubusercontent.com/bitnami/bitnami-docker-laravel/master/docker-compose.yml`

Kad konfigurācijas fails ir lejupielādēts, jāizpilda komanda: `$ docker-compose up`, Šī komanda palaiž lietojumprogrammu docker, kurš vadoties pēc konfigurācijas faila, uzstāda visu nepieciešamo Laravel 8 izstrādei.

Kā ir norādīts Bitnam/Laravel attēla dokumentācijā – vēlams konfigurācijas failā nomainīt datubāzes un aplikācijas piekļuves datu informāciju. Nomainot šos datus, jāveic laravel konfigurācija .env failā atbilstoši tām, attēlā 7.19. att. ir redzams docker konfigurācijas faila fragments ar autora izmainītiem piekļuves datiem, un 7.20. att. ir redzamas laravel konfigurācijas faila fragments.

```

mariadb:
  image: docker.io/bitnami/mariadb:10.2
  environment:
    # ALLOW_EMPTY_PASSWORD is recommended only for development.
    - MARIADB_ROOT_PASSWORD=root
    - ALLOW_EMPTY_PASSWORD=yes
    - MARIADB_USER=dmitrijs
    - MARIADB_DATABASE=laravel_eshop
    - MARIADB_PASSWORD=root
myapp:
  tty: true
  image: docker.io/bitnami/laravel:8
  environment:
    - DB_HOST=mariadb
    - DB_USERNAME=dmitrijs
    - DB_DATABASE=laravel_eshop
    - DB_PASSWORD=root
  
```

7.19. att. Laravel 8 bitnami docker konteineru konfigurācijas fails (autora veidots ekrānšāviņš)

```

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel_eshop
DB_USERNAME=dmitrijs
DB_PASSWORD=root
  
```

7.20. att. Laravel .env faila konfigurācijas fragments datubāzei (autora veidots ekrānšāviņš)

React projekts tika uzstādīts izmantojot `npx create-react-app` komandu iepriekš izveidotā mapē laravel projektā - /eshop-frontend/admin.

7.4.2 API konfigurēšana

Lai React lietotne varētu sūtīt pieprasījumus uz Laravel backend lietotni, ir nepieciešama papildus bibliotēka asinhronajiem HTTP pieprasījumiem, populārākā ir AXIOS.

Axios tika uzstādīts izmantojot YARN pakotņu uzstādītāju izmantojot komandu `yarn add axios`.

Lai atvieglotu pieprasījumu funkciju rakstīšanu tika izveidota palīgfunckcija, kurā norādītas pamata konfigurācijas, kuras ir nepieciešamas katrā pieprasījumā – apskatāms 7.21. att.

```

1 import axios from 'axios';
2 axios.defaults.withCredentials = true;
3 const http = axios.create({
4   baseURL: 'http://localhost:4000',
5   headers: {
6     "Accept": "application/json",
7     'Authorization': `Bearer ${sessionStorage.getItem('bearer_token')}`
8   }
9 });
10
11 window["http" as any] = http as any;
12
13 export { http };
14

```

7.21. att. Axios konfigurācija (autora veidots ekrānšāviņš)

Kā redzams tika uzstādīts bāzes URL un HTTP galveņu uzstādījumi – svarīgs ir Bearer tokens, kura esamība norād uz to, ka lietotājs ir autorizējies, bez šī tokena, nav iespējams saņemt informāciju no aizsargātiem URL ceļiem.

Izveidojot šādu palīgfailu, turpmāk jebkurā komponentē pieprasījumu ir iespējams veikt ar skriptu `http.get('/api/products')`, vai līdzīgu.

Pieprasījuma izpildei bija nepieciešama CORS konfigurācija, lai ļautu react lietotnei, kura būtībā ir cita vietne, piekļūt laravel API ceļiem.

7.4.3 Lietotāja saskarnes veidošana

Lietotāja saskarne tika veidota izmantojot React paredzētu saskarņu veidošanas bibliotēku ANTD design. Tā tika uzstādīta izmantojot Yarn pakotņu pārvaldnieku, izmantojot komandu `yarn add antd`.

Tāpat kā populārākajā lietotņu izstrādes bibliotēkā Bootstrap, ANTD ir gatavi nepieciešamie saskarnes elementi, piemēram – Form, Table, Button, Row, Col, u.c. Taču atšķirībā no Bootstrap, ANTD bibliotēka ir paredzēta izstrādei ar React, un tajā pieejamās komponentes ir aprīkotas ar papildus funkcionalitāti, savu API.

Vispirms tika izveidots lietotnes “skelets”, šim nolūkam `App.tsx` failā tika izveidota struktūras koks, apskatāms 7.22. att.

```

101 ? <Layout style={{ minHeight: "100vh" }}>
102 >   <Layout.Sider breakpoint="lg" collapsedWidth="0" theme="dark" className="sider">...
110 </Layout.Sider>
111   <Layout>
112     <PageHeader
113       onBack={history.goBack}
114       title={location}
115       className="header"
116     />
117 >   <Layout.Content style={{ margin: 20 }}>...
151 </Layout.Content>
152   <Layout.Footer className="footer">Copyright</Layout.Footer>
153 </Layout>
154 </Layout>

```

7.22. att. Lietotnes pamata skelets (autora veidots ekrānšāviņš)

Elements `<Layout>` ir konteineris lapai, kurā tiek izvietoti lapas galvenie elementi. `<Layout.Sider>` elements izveido sānu navigāciju, kā redzams attēlā, tam līdzīgi kā html tegam ir norādīti atribūti, šie atribūti ir noteikti bibliotēkā. `<PageHeader />` elements izveido lapas galveni. `<Layout.Footer>` elements izveido lapas kājēni.

`<Layout.Content>` sadaļā notiek skatu maiņa izmantojot `react-router-dom` pakotni. Šī pakotne ļauj izveidot ceļus konkrētiem komponentiem hipersaišu izskatā – apskatāms 7.23. att.

```

117   <Layout.Content style={{ margin: 20 }}>
118     <Card>
119       <Switch>
120         <Route path="/orders">
121           </Route>
122         <Route exact path="/categories/new">
123           <CategoriesForm />
124         </Route>
125         <Route exact path="/categories/edit/:id">
126           <CategoriesForm />
127         </Route>
128         <Route exact path="/categories">
129           <Categories config={config}/>
130         </Route>
131         <Route path="/items/new">
132           <ItemsForm />
133         </Route>
134         <Route path="/items/edit/:id">
135           <ItemsForm />
136         </Route>

```

7.23. att. React-router-dom komponentu navigācija.

Atkarībā no esošā URL ceļa, tiek parādīts noteiktais skats.

Antd bibliotēka ievērojami paātrina tabulu veidošanu, kā redzams 7.24. att., komponentei `<Table />` atribūtos tiek norādīti parametri, kā šai komponentei jādarbojās.

```

<Table
  columns={columns}
  dataSource={categories}
  rowKey={record => record.id}
  onRow={({record, rowIndex}) => {
    return {
      onClick: event => { history.push(`/categories/edit/${record.id}`) }, // click row
      onDoubleClick: event => { }, // double click row
      onContextMenu: event => { }, // right button click row
      onMouseEnter: event => { }, // mouse enter row
      onMouseLeave: event => { }, // mouse leave row
    };
  }}
  scroll={{ x: 500 }}
  loading={props.loading}
/>
)
}

```

7.24. att. <Table /> komponente (autora veidots ekrānšāviņš)

Galvenie atribūti – `columns` un `dataSource`. Atribūtam `columns` tiek padots masīvs ar objektiem, kur katrs objekts norāda tabulas kolonnas parametrus, bet `dataSource` atribūtam tiek padots masīvs ar datiem, kuri ir jāattēlo.

7.4.4 Autorizācijas realizācija

Autorizācijas funkcijas realizēšanai tika izmantota Laravel papildus bibliotēka Sanctum. Šī bibliotēka ļauj izveidot vienkāršu autorizāciju API pieprasījumiem. Bibliotēkas uzstādīšana notiek izmantojot komandu `composer install Laravel/Sanctum` un pēc pakotnes uzstādīšanas, jāveic migrācija, jo izveidojās speciālas tabulas, kuras glabā datus par ierakstītajiem lietotājiem.

Pamatā šī pakotne izveido bāzes uzstādījumus un kontrolierus vienai lietotāju grupai Users, taču tā kā projektā ir paredzētas vairākas lietotāju grupas, tika izveidots modelis un kontrolieris Administrācijas portāla autorizācijai – Admin un AdminAuthController. Par pamatu tika ņemti User un UserAuthController pakotnes izveidotie faili.

Lai norādītu, kuri API ceļi ir aizsargāti un kuri ir piejami bez autorizācijas, tiek izmantots Laravel bibliotēkas metode `middleware`. Kā redzams 7.25. att. visi ceļi, kuri atrodās `middleware` metodes parametros ir aizsargāti izmantojot `auth:sanctum` direktīvu. Vienīgais neaizsargātais ceļš ir `/login`, jo ir nepieciešama iespēja nosūtīt un saņemt datus autorizējoties.

```

Route::post('/login', [AdminAuthController::class, 'login']);

Route::group(['middleware' => ['auth:sanctum']], function () {
  Route::apiResources([
    'products' => ProductController::class,
    'category' => CategoryController::class,
  ]);
  Route::get('/categories-tree', [CategoryController::class, 'showTree']);
  Route::get('/products-by-category/{id}', [ProductController::class, 'getProductsByCategory']);
  Route::post('/logout', [AdminAuthController::class, 'logout']);
});

```

7.25. att. Middleware aizsargātie ceļi (autora veidots ekrānšāviņš)

Kad lietotājs ievada datus, autorizācijas formā un apstiprina nosūtīšanu, tiek nosūtīts

pieprasījums kurš satur epastu un paroli. Kā redzams 7.26. att. tiek meklēts lietotājs pēc epasta, ja lietotājs ir atrasts, ar speciālu hash funkciju tiek šifrēta parole un salīdzināta ar šifrēto paroli no datubāzes, ja tās sakrīt, tiek nosūtīts atbildē tokens, kuru izveido sanctum bibliotēka.

```

19 public function login(Request $request) {
20     $fields = $request->validate([
21         'email' => 'required|string',
22         'password' => 'required|string'
23     ]);
24
25     $admin = Admin::where('email', $fields['email'])->first();
26
27     if(!$admin || !Hash::check($fields['password'], $admin->password)) {
28         return response([
29             'message' => 'Kaut kas nav pareizi'
30         ], 401);
31     }
32
33     $token = $admin->createToken('eshopToken')->plainTextToken;
34
35     $response = [
36         'admin' => $admin,
37         'api_token' => $token,
38     ];
39
40     return response($response, 201);
41 }

```

7.26. att. Ierakstīšanās sistēmā funkcija (autora veidots ekrānšāviņš)

Kā redzams 7.27. att., React front-end pusē ierakstīšanās sistēmā pieprasījums tiek nosūtīts tikai tad, kad tiek veiksmīgi izpildīts csrf pieprasījums, tā ir laravel iebūvētā aizsardzības funkcija.

```

const loginRequest = async (values: any) => {
  await http.get('/sanctum/csrf-cookie').then(() => {
    http.post('/api/login', values, props.config).then((response: any) => {
      props.returnToken(response.data.api_token)
    }).catch(() => {
      message.error("Nepareizi ievadīti dati")
    })
  })
};

```

7.27. att. Pieprasījums ierakstīties sistēmā front-end pusē (autora veidots ekrānšāviņš)

Kad atbilde tiek saņemta, api_tokens no login komponentes tiek nosūtīts virskomponentei, kurā savukārt tas tiek piešķirts sessijas sīkdatnei. Kā redzams, 7.28. att. Tālāk react lietotnes skata stāvokli nosaka tas, vai ir vai nav bearer token sessijas sīkdatnē, loggedIn, ja ir tokens.

```

return (loggedIn
  ? <Layout style={{ minHeight: "100vh" }}>
    <Layout.Sider breakpoint="lg" collapsedWidth="0" theme="dark" className="sider">...
  </Layout.Sider>
  <Layout>...
</Layout>
  </Layout>
  : <LoginPage config={config} returnToken={login} />
);

```

7.28. att. Skats atkarībā no tā vai ir ielogojies vai nē.

7.4.5 Kategoriju koka izvadišana

Tā kā kategorijas varbūt ar apakškategorijām, bija nepieciešams izveidot funkciju, kura

ļauj saņemt strukturētus datus koka veidā. Šim nolūkam tik izveidota rekursīva funkcija, kura iet cauri kategoriju sarakstam, un ja apstrādājamai kategorijai ir atrasta apakškategorija tā tiek pievienota kategorijas objektam, children masīvā. Kā redzams 7.29. att., Ja tiek atrasta apakškategorija masīvam tiek pievienota vērtība ko atgriezīs rekursīvi palaista funkcija.

```
protected function treeMaker(Category $item, $categories)
{
    $children = array();
    foreach ($categories as $category) {
        if ($category->parent_category == $item->id) {
            array_push($children, $this->treeMaker($category, $categories));
        }
    }
    $item->children = $children;
    return $item;
}
```

7.29. att. Rekursīva funkcija kategoriju koka izveidošanai (autora veidots ekrānšāviņš)

Front-end daļā šo koka objektu apstrādā rekursīva komponenta funkcija, tā izveido strukturētu sarakstu, funkcija apskatāma 7.30. att.

```
const RecursiveTree = (props: any) => {
    const hasChildren = props.items.children.length > 0

    return (
        <>
        <li>{props.items.name}</li>
        {hasChildren && props.items.children.map((item: any) => (
            <ul key={item.id}>
            <RecursiveTree items={item}/>
            </ul>
        ))}
        </>
    )
}
```

7.30. att. Rekursīva komponentes funkcija (autora veidots ekrānšāviņš)

7.4.6 Ierobežojumi

Uzstādot datubāzi no jauna, ir nepieciešams manuāli ievadīt administrātorā lietotāja ierakstu un vienu kategoriju, kā root kategoriju. Šobrīd tas tiek realizēts izmantojot Laravel database seeder funkciju.

Ievadot jaunu produktu, sākumā tiek norādīts tā daudzums, pēc tam daudzumu var koriģēt papildinot precī par N vienībām. Ideja, ka preču daudzums varēs samazināties tikai ar papildus funkcionalitātes ieviešanu sistēmā, lai veidojās patiesa uzskaitē.

Produktu var dzēst izmantojot Laravel softDelete funkciju, kura atstāj ierakstu, bet pievieno dzēšanas datumu, tādā veidā norāda, ka šis ieraksts nav jāņem no datubāzes, tas ir domāts, tam, lai ierakstu varētu atjaunot un lai nākotnē saglabājās relācija uz pasūtījumiem.

Kategorijas augškategoriju mainīt nevar, nākotnē paredzēts izstrādāt loģiku pēc kuras to varēs darīt un arī varēs dzēst, pagaidām, tas nav realizēts.

SECINĀJUMI

1. Veicot prasību dokumenta izstrādi ir svarīgi izprast nozares īpatnības un biznesa loģiku, ļoti palīdz intervija ar klientu un esošo sistēmu izpētīšana.
2. Pēc iespējas sīkāk izstrādāta specifikācija, ļauj ātrāk izstrādāt produktu, taču tas var prasīt vairāk laika, nekā izstrāde pēc Agile metodes, kurā var tikt veikti labojumi izstrādes laikā konsultējoties ar klientu.
3. Sekojot līdz izstrādes tendencēm ir iespējams atvieglot un mobilizēt izstrādes vides, piemēram izmantojot lietotni Docker.
4. Mūsdienīgu satvaru un bibliotēku izmantošana, ļauj ne tikai paātrināt izstrādes gaitu samazinot koda daudzumu, bet arī garantē augstāku drošību un programmas ātrdarbību, jo populāri satvari tiek testēti un bieži atjaunināti.
5. Grāmatu izdevēju nozarē, kā izrādās, lielākais šķērslis digitalizācijā ir digitālās prasmes un zināšanu trūkums, trūkst jaunu speciālistu.
6. Ņemot vērā zemu respondentu daudzumu elektroniskai aptaujai, neskatoties uz labu izplatīšanas kanālu izvēli, autors uzskata, ka bija jāizdomā rezerves variants informācijas ieguvei par nozari.
7. Lai arī projektā noteiktās prasības ir minimālas priekš e-komercijas, tomēr izstrādes gaitā parādās jaunas prasības, kuru ieviešana aizņem papildus laiku.
8. Docker izstrādes vide, neskatoties uz norādīto atbalstu visām populārākajām operētājsistēmām, tomēr prasa papildus konfigurāciju un ātrāk darbojās uz Linux nekā uz Windows 10 operētājsistēmas.
9. Jāseko līdz front-end izstrādes tendencēm un pētījumiem, jo vienas lapas lietotnes izstrāde, kas mūsdienās ir tik populāra var radīt problēmas saistītas ar meklētājprogrammu optimizāciju – SEO.
10. Patērējot laiku izstrādājot universālas komponentes, var ievērojami atvieglot tālāko sistēmas izstrādi.
11. Ņemot vērā lielu daudzumu e-komercijas risinājumu izstrādes iespēju, arī bez programmēšanas, un to cik daudz laika var aizņemt jauna risinājuma izstrāde, iespējams, ka uzņēmumiem ar mazu budžetu, piemērotākais risinājums būtu gatavas platformas.
12. Rakstot programmas kodu autors vadījās pēc iespējamajiem testēšanas veidiem, kuri varētu būt beigās, tādēļ daudz iespējamās kļūdas un nenoteiktības, tika novērstas programmēšanas gaitā.

PRIEKŠLIKUMI

1. Veltīt vairāk laika nozares un klienta darbības novērošanai, tas palīdzēs labāk izprast vajadzības un iespējams ieteikt klientam, kā labāk izstrādāt produktu.
2. Iepazīties labāk ar pieejamiem projektašanas un inženierijas rīkiem tirgū, jo tie var palīdzēt veikt analīzi un izplānot izstrādi.
3. Radīt drošus risinājumus izplatīt grāmatas elektroniskā veidā, samazināt pirātisma iespēju.
4. Veikt ātrdarbības pārbaudi ar Docker un bez, jo šķiet, ka darbības ātrums ir par lēnu.
5. Izveidot izstrādātajiem moduļiem meklēšanas un filtrēšanas opcijas, jo pieaugot produktu daudzumam, nebūs iespēja tos atlasīt un atrast.
6. Padziļināt zināšanas Laravel, jo tajā ir ļoti daudz gatavu risinājumu, tieši interneta veikalu funkciju realizēšanā.
7. Iepazīties ar backend izstrādes satvariem, kuri izmanto programmēšanas valodu Javascript, jo iespējams, tie ļauj efektīvāk sasaistīts front-end Javascript satvarus ar servera pusi.
8. Iepazīties ar frontend vienas lapas aplikācijas izstrādes ietvariem Vue un Angular, jo iespējams, ka tie būtu ātrdarbīgāki nekā react.
9. Iepazīties ar vienas aplikācijas tīmekļa vietņu renderēšanas servera pusē tehnoloģijām, tas ļaus sasniegt pietiekamu meklētājprogrammu optimizāciju izmantojot vienu tehnoloģiju visai vietnei.

LITERATŪRAS UN INFORMĀCIJAS AVOTU SARAKSTS

1. Altexsoft (2020). Technical Documentation in Software Development: Types, Best Practices, and Tools. [skatīts 04.12.2021.]. Pieejams: <https://www.altexsoft.com/blog/business/technical-documentation-in-software-development-types-best-practices-and-tools/>
2. Aman Goel (2021). 10 Best Web Development Frameworks. [skatīts 02.12.2021.]. Pieejams: <https://hackr.io/blog/web-development-frameworks>
3. Anita Stivriņa (2020). Programmatūras prasību specifikācija. [skatīts 13.12.2021.]. Prezentācija, 2. Lekcija, studiju priekšmetā Programmatūras inženierija.
4. Artem Cervichnik Svitlana Varaskina (2021). Single-Page Applications vs Multi-Page Applications: The Battle of the Web Apps. [skatīts 02.12.2021.]. Pieejams: <https://themindstudios.com/blog/spa-vs-mpa/>
5. Bartosz Stryga (2021). SPA SEO: Mission Impossible?. [skatīts 02.12.2021.]. Pieejams: <https://www.magnolia-cms.com/blog/spa-seo-mission-impossible.html>
6. COMPLEX TESTER (2012). SPIRAL, PROTOTYPE, FISH BONE, SASHIMI. [skatīts 22.11.2021.]. Pieejams: <https://complextester.wordpress.com/2012/08/07/spiral-prototype-fish-bone-sashimi/>
7. Computer Science Department, University of Cape Town (2011). The software crisis Chapter 2. Process and Model. [skatīts 20.11.2021.]. Pieejams: https://www.cs.uct.ac.za/mit_notes/software/htmls/ch02s02.html
8. Difference between (2018). Difference Between API and Web Service. [skatīts 02.12.2021.]. Pieejams: <http://www.differencebetween.net/technology/internet/difference-between-api-and-web-service/#ixzz3e3WxplAv>
9. Docker (2021). Overview of Docker Compose. [skatīts 04.12.2021.]. Pieejams: <https://docs.docker.com/compose/>
10. Docker (2021). Use containers to Build, Share and Run your applications [skatīts 15.09.2021.]. Pieejams: <https://www.docker.com/resources/what-container>
11. Docker hub (2021). Bitnami/Laravel image [skatīts 16.09.2021.]. Pieejams: <https://hub.docker.com/r/bitnami/laravel>
12. Eban Scott (2020). How does CRUD relate to a REST API? [skatīts 02.12.2021.]. Pieejams: <https://codebots.com/crud/how-does-crud-relate-to-a-rest-api>
13. Geeksforgeeks (2021). Introduction of ER Model. [skatīts 14.12.2021.]. Pieejams: <https://www.geeksforgeeks.org/introduction-of-er-model/>
14. Harness Author (2021). Understanding The Phases Of The Software

Development Life Cycle. [skatīts 22.11.2021.] Pieejams: <https://harness.io/blog/software-development-life-cycle/>

15. J. A. N. Lee (2017). Friedrich (Fritz) L. Bauer. [skatīts 20.11.2021.]. Pieejams: <https://history.computer.org/pioneers/bauer.html>

16. Jama Software (2021). 11 Requirements Gathering Techniques for Agile Product Teams. [skatīts 04.12.2021.]. Pieejams: <https://www.jamasoftware.com/requirements-management-guide/requirements-gathering-and-management-processes/what-is-requirements-gathering>

17. James Higginbotham (2018). Taking REST beyond CRUD. [skatīts 02.12.2021.]. Pieejams: <https://tyk.io/blog/taking-rest-beyond-crud/>

18. Java T Point (2021). Incremental Model. [skatīts 22.11.2021.]. Pieejams: <https://www.javatpoint.com/software-engineering-incremental-model>

19. Juniata College(2020). Entity Type Hierarchies. [skatīts 14.12.2021.]. Pieejams: <http://jcsites.juniata.edu/faculty/rhodes/dbms/eermodel.htm>

20. Kathrun Marr (2018). WooCommerce Pricing: How Much Does it Cost to Run a Store?. [skatīts 13.11.2021.]. Pieejams: <https://woocommerce.com/posts/woocommerce-pricing/>

21. Kseniia Kyslova (2021). eCommerce web development: What approach and tools are right for your online store?. [skatīts 13.11.2021.]. Pieejams: <https://proxify.io/articles/ecommerce-web-development>

22. Laravel documentation (2021). Database: Getting Started. [skatīts 04.12.2021.]. Pieejams: <https://laravel.com/docs/8.x/database>

23. MariaDB (2021). Documentation. [skatīts 04.12.2021.]. Pieejams: <https://mariadb.org/documentation/>

24. Matthew Martin (2021). What is Software Engineering? Definition, Basics, Characteristics. [skatīts 20.11.2021.] Pieejams: <https://www.guru99.com/what-is-software-engineering.html>

25. MDN Web Docs (2021). An overview of HTTP. [skatīts 01.12.2021.]. Pieejams: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>

26. PerformanceLab (2021). Importance of Software Testing in SDLC. [skatīts 04.12.2021.]. Pieejams: <https://performancelabus.com/software-testing-importance-sdlc/>

27. PROTOCOL SUPPORT LIBRARY (2021). Hypertext Transfer Protocol (HTTP). [skatīts 01.12.2021.]. Pieejams: <https://www.extrahop.com/resources/protocols/http/>

28. RapidAPI (2019). API vs Web Service: What's the Difference? [skatīts

01.12.2021.]. Pieejams: <https://rapidapi.com/blog/api-vs-web-service/>

29. Sofiia (2020). Software development process using waterfall method. [skatīts 21.11.2021.]. Pieejams: <https://medium.com/@sofiia/software-development-process-using-waterfall-method-introductuin-3bd7a9c2f912>

30. Software Testing Help (2021). SDLC (Software Development Life Cycle) Phases, Process, Models. [skatīts 22.11.2021.]. Pieejams: <https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/>

31. Sparx Systems (2020). Conceptual Data Model. [skatīts 14.12.2021.]. Pieejams: https://sparxsystems.com/enterprise_architect_user_guide/15.2/model_domains/conceptual_data_model.html

32. Tutorialspoint (2021). SDLC - Agile Model. [skatīts 23.11.2021.]. Pieejams: https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm

33. Visual-paradigm (2021). Conceptual, Logical and Physical Data Model. [skatīts 14.12.2021.]. Pieejams: https://www.visual-paradigm.com/support/documents/vpuserguide/3563/3564/85378_conceptual,l.html

34. Vladimir Sechko, Nadejda Alkhaldi (2021). Your guide to writing a software requirements specification (SRS) document. [skatīts 20.11.2021.]. Pieejams: <https://itrexgroup.com/blog/guide-to-software-requirements-specification-document-benefits-expert-tips/#>

PREZENTĀCIJAS DAĻA



Alberta Koledža
Informāciju tehnoloģijas – «Programmēšana»

Dmitrijs Jaunslavietis

Kvalifikācijas darbs

**TĪMEKĻA VIETNES IZSTRĀDE GRĀMATU
IZDEVĒJIEM**

2022, Rīga

Problēmas apraksts



- Ierobežots budžets
- Popularitāte
- Novecojušas tehnoloģijas
- Zināšanu trūkums

Izdevniecība AVOTS 

1/3/2022

2

Mērķi un uzdevumi

- Izpētīt nozari tās digitālo pratību un e-komercijas īpatnības;
- Ievākt minimālās prasības e-komercijas risinājumam;
- Izpētīt e-komercijas pieejamos risinājumus;

Uzdevumi:

- Veikt aptaujas un intervijas
- Izveidot projekta dokumentāciju
- Sākt tīmekļa vietnes izstrādi



1/3/2022

3

Izmantotās pētniecības metodes

- Intervija ar nozares pārstāvjiem
- Aptauja grāmatu izdevējiem
- Monogrāfiskā metode – pieejamo dokumentāciju un informācijas avotu pētīšana
- Analīzes metode – esošo un pieejamo risinājumu.



1/3/2022

4

Teorētisko aspektu raksturojums

- Speciālistu trūkums nozarē un zems respondences līmenis
- Agile un prototipēšanas metode
- Minimālās prasības nozarei
- Mūsdienīgas tehnoloģijas – MVC Laravel un SPA React.js
- REST API pieeja
- Docker kā izstrādes vide



1/3/2022

5

Praktiskā darba raksturojums

- Pirmais «Sprints»

Administrācijas panelis (front-end)

The screenshot displays an administrative interface for managing categories. It features a dark sidebar with navigation options: 'Admin', 'Pārvaldītājs', 'Kategorijas', 'Grāmatas', 'Klases', 'Statistika', and 'Iziet'. The main content area is titled 'Kategorijas' and includes a 'Pievienēt kategoriju' button. Below this is a table listing categories:

Pieder pie	Kategorija	URL ceļš	Precu daudzums kategorijā
/	Visas grāmatas	visas-grāmatas	0

Below the table, there is a 'Kategoriju karte' section showing a tree view with 'Visas grāmatas' as a child item. Navigation controls (back, forward, and a highlighted '1') are visible at the bottom right of the table area.



1/3/2022

6

Praktiskā darba raksturojums

- Autorizācija api-token (Laravel Sanctum)
- Rekursīva koku izvadīšana backend PHP un frontend JavaScript (TypeScript)
- Produktu (grāmatu soft Delete)
- Docker operētājsistēmu savietojamība



1/3/2022

7

Secinājumi

1. Veicot prasību dokumenta izstrādi ir svarīgi izprast nozari;
2. Digitālās prasmes trūkums, trūkst jaunu speciālistu;
3. Agile un prototipēšanas apvienošana;
4. Zems budžets neļauj izstrādāt individuālus risinājumus;
5. Nav universālu rīku, kuri der visam, to kompetenta apvienošana ļauj sasniegt vēlamo rezultātu.



1/3/2022

8

Priekšlikumi

1. Nepieciešams vairāk iepazīties ar programmatūras izstrādes palīgrikiem;
2. Veikt Docker vides ātrdarbības salīdzināšanu;
3. Veltīt laiku Laravel iebūvētiem e-komercijas risinājumiem;
4. Iepazīties ar SPA lietotņu servera puses renderēšanu;
5. Piemeklēt nozarei piemērotu prasību definēšanas pieeju.



1/3/2022

9



Paldies par uzmanību!

PIELIKUMI

Testēšanas dokuments

Šis dokuments atspoguļo tīmekļa lietotnes “Administrācijas panelis” testēšanas gaitu un rezultātus. Testējamās vienības izstrādātājs ir Dmitrijs Jaunslavietis.

Lietotne ir paredzēta izstrādājamā interneta veikala kategoriju veidošanai un produktu pievienošanai, kā arī uzskaitēi.

Lietotne ir izstrādāta izmantojot operētājsistēmu Linux Ubuntu 20.04. Lietotne ir paredzēta lietošanai mūsdienās aktuālajos tīmekļa pārlūkos. Lietotnes izstrādē tika izmantotas programmēšanas valodas Javascript, CSS, PHP un ietvari Laravel, React.js. Lietotnes vide darbojās izmantojot docker konteineru sistēmu.

Testēšanai izmantojamā pieeja ir Melnās kastes metode, kuras nolūks pārbaudīt vai izpildās funkcionālās un nefunkcionālās prasības.

Testēšanas rezultāti

Testēšana norisinājās balstoties uz testēšanas nosacījumiem tabulā.

tabula

Testēšanas nosacījumu tabula

Prasība	Testa mērķis	Testēšanas veids
Pievienot jaunu preci	Pārbaudīt vai forma nepieņem laukam neparedzētus datus	<ul style="list-style-type: none"> • Ciparu ievades laukā ievadīt simbolus un burtus • Saglabāt produktu, bez obligātiem laukiem
Pievienot jaunu kategoriju	Pārbaudīt vai forma nepieņem laukam neparedzētus datus	<ul style="list-style-type: none"> • Ciparu ievades laukā ievadīt simbolus un burtus • Saglabāt kategoriju, bez obligātiem laukiem
Labot preci	Pārbaudīt vai forma nepieņem laukam neparedzētus datus	<ul style="list-style-type: none"> • Ciparu ievades laukā ievadīt simbolus un burtus • Saglabāt kategoriju, bez obligātiem laukiem

1. pielikuma turpinājums
tabulas turpinājums

Labot kategoriju	Pārbaudīt vai forma nepieņem laukam neparedzētus datus	<ul style="list-style-type: none"> • Ciparu ievades laukā ievadīt simbolus un burtus • Saglabāt kategoriju, bez obligātiem laukiem
Pievienot precei kategoriju	Pārbaudīt vai var mainīt preces kategoriju	<ul style="list-style-type: none"> • Izmainīt izveidotas preces kategoriju
Aizsargāt piekļuvi ar paroli un lietotāja piekļuves vārdu	Pārbaudīt vai ir iespēja piekļuve datiem bez autorizācijas, vai ar nepareiziem datiem.	<ul style="list-style-type: none"> • Ievadīt URL no citiem administrācijas paneļa skatiem. • Ievadīt dažādus piekļuves datus
Izrakstīties no sistēmas	Pārbaudīt vai lietotājs tiešām izrakstās no sistēmas.	<ul style="list-style-type: none"> • Izrakstīties no sistēmas un veikt refresh un manuāli ievadīt URL no citiem administrācijas paneļa skatiem.
Ātrdarbība	Pārbaudīt cik daudz dati atnāk no datubāzes	<ul style="list-style-type: none"> • Ievadīt 100 produktus un pārbaudīt vai atnāk tikai paginācijai attiecīgie dati.
Pieejamība uz dažāda izmēra ekrāniem	Pārbaudīt vai uz dažāda izmēra ekrāniem ir pieejama visa informācija un ievadlauki	<ul style="list-style-type: none"> • Veikt visu skatu novērošanu pie dažādām izšķirtspējām no 320px līdz 1980px platumam.

Pievienot jaunu preci

Pievienojot preci laukos kuri pieņem jebkākus simbolus un ciparus (teksta ievadlauks) tika ievadīti kirilicas burti un speciālie simboli < > ? & % # \$;

Ievadlaukos ar atzīmi “skaitlis” tika ievadīti negatīvi skaitļi, burtus ievadīt neļauj.

Lauks daudzums ļauj ievadīt burtus.

Rezultāts – Teksta lauki pieņem jebkākus simbolus, ievadlauki ar atzīmi skaitlis parāda kļūdu, ja ir ievadīts negatīvs skaitlis, lauks “daudzums” ļauj ievadīt burtus (jālabo), taču nesaglabā serverī, jo backendā notiek validācija.

Saglabāt produktu bez obligātajiem laukiem.

Produkta obligātie lauki – nosaukums, isbn numurs, daudzums noliktavā un cena.

1. pielikuma turpinājums

Saglabāt bez šiem laukiem neļauj front-end validācija – parādās uzraksts, ka lauks ir obligāts.

Bez pārējiem laukiem saglabāšana notiek korekti.

Pievienot jaunu kategoriju

Saglabāt jaunu kategoriju, bez obligātiem laukiem.

Obligāti lauki – nosaukums, ceļš, augškategorija.

Nosaukums – Validācija nostrādā backendā, dati nesaglabājās

Augškategorijai noklusētā izvēle ir root kategorija, nav iespējams ievadīt tukšu.

Ceļš – saglabā bez datiem laukā – Jālabo

Labot precī un labot kategoriju

Rezultāti identiski pievienošanas formām, jo tiek izmantoti vieni un tie paši komponenti.

Izmainīt preces kategoriju

Pārbaudīt vai ir iespējams nomainīt preces kategoriju.

Preces kategoriju var izmainīt – pārbaudīts gan datubāzē, gan preču sarakstā.

Autorizācijas apiešana

Ievadīt URL no citiem administrācijas paneļa skatiem.

Neierakstoties sistēmā pārbaudīti visi uz šo brīdi iespējamie ceļi:

localhost:3000/items, localhost:3000/categories, localhost:3000/items/new, localhost:3000/items/edit/1, localhost:3000/categories/new, localhost:3000/categories/edit/1

Autorizācijas skats netika apiets.

Autorizētam lietotājam izrakstoties no sistēmas pārliecināties, ka tiešām ir izrakstījies.

Identiski ar iepriekšējo testu – kad lietotājs ir izrakstījies no sistēmas, ierakstīšanās skatu apiet nav sanācis izmantojot augstāk minētos URL.

Pārbaudīt, vai tiek ņemts vērā gan epasts gan attiecīga parole.

Izveidots vēl viens lietotājs. Sistēmā patreiz ir:

test@test.lv ar paroli test123 un test2@test2.lv ar paroli 123test.

Testa uzdevums pārbaudīt vai attiecīgais lietotājs var ielogoties tikai ar savu paroli, vai epasts ir saistīts ar paroli. Pirms tam pārbaudīts, vai attiecīgais lietotājs spēj ielogoties ar savu paroli.

Testu rezultāti:

test@test.lv 123test – KĻŪDA,

1. pielikuma turpinājums

test2@test.lv test123 – KĻŪDA,
test@test.lv qwerty – KĻŪDA,
test2@test2.lv qwerty – KĻŪDA,
example@example.com test123 – KĻŪDA,
example@example.com 123test – KĻŪDA.

Ātrdarbības pārbaude

Patreiz nav izveidota dalīta datu atgriešana, tādēļ tests nav iespējams.

Pieejamība uz dažāda izmēra ekrāniem

Pārbaudītie ekrānu platumi:

1920px, 1600px, 1366px, 1280px, 768px, 520px, 375px un 320px.

1920px, 1600px, 1366px – viss izskatās ļoti labi, visi formas lauki ir vietās un uzraksti nepārklājās.

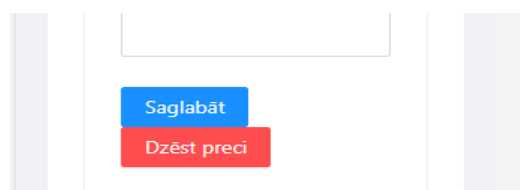
1280px – Īpaši gariem formu nosaukumi tiek pārnesti jaunā rindā, dēļ kā lauki nav vienā rindā, skatīt 7.32. att. – jālabo!

768px, 520px – Sānu navigācija slēpjas un parādās, artefakti nav novēroti.

375px, 320px – Jāpārdomā pārskatāmāka tabula, pie 320px – poga “Dzēst preci” pāriet jaunā rindā, skatīt 7.33. att. – jālabo!

Pārbaudīts izskatas arī pie gadījuma platumiem – nav novēroti jauni artefakti izņemot, tos kuri aprakstīti augstāk.

7.32. att. Artefakts pie 1280px platumā (autora veidots ekrānšāviņš)



7.33. att. Artefakts pie 320px platumā (autora veidots ekrānšāviņš)

Lietotāja ceļvedis

Tīmekļa lietotnes “Administrācijas portāls” moduļi “Kategorijas” un “Grāmatas” ir paredzēti topošās tīmekļa vietnes grāmatu izdevējiem, interneta veikala sadaļas kategoriju un produktu pārvaldīšanai.

Modulis kategorijas atbild par jaunu kategoriju pievienošanu, kategoriju struktūras veidošanu, kategoriju rediģēšanu un pārlūkošanu.

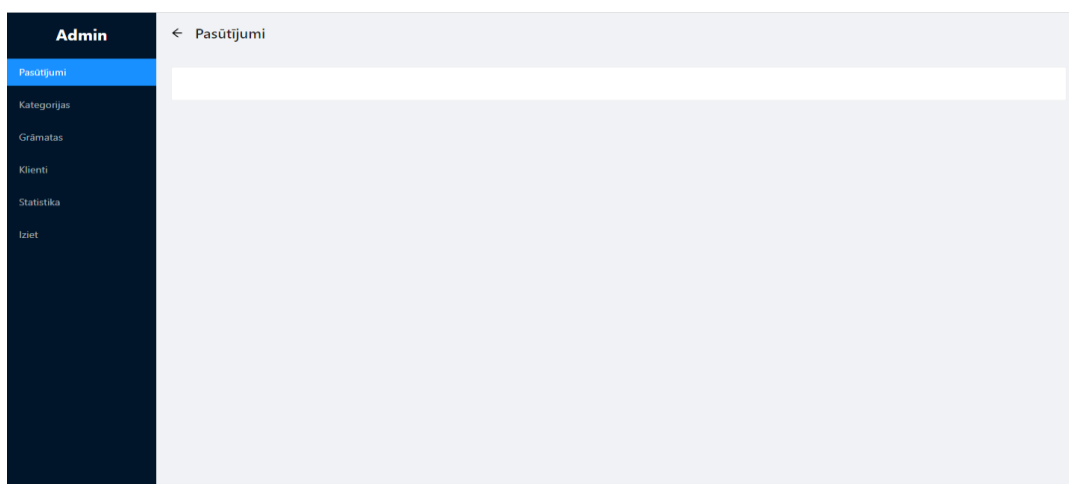
Modulis grāmatas atbild par jaunu grāmatu pievienošanu, kategorizēšanu, pārvaldīšanu, pārlūkošanu, dzēšanu un rediģēšanu.

Darba uzsākšana

Lai pievienotos sistēmai, tīmekļa pārlūkā ievadiet jūsu interneta veikala adresi un aizslīpšvītras admin – piem. <https://veikals.lv/admin>. (Patreiz localhost:3000).

Jūsu pārlūkā atvērsies autorizācijas logs, kurā ievadiet jūsu lietotāja epasta adresi un paroli. (test@test.lv, parole:test123).

Ja autorizācija ir veiksmīga jūsu pārlūkā atvērsies administrācijas panelis, apskatāms 1.att.



1. att. Administrācijas panelis autorizēts (autora veidots ekrānšāviņš)

Lai izrakstītos no sistēmas uzspiediet pogu “Iziet” sānu navigācijas joslā.

Kategoriju pārvaldīšana

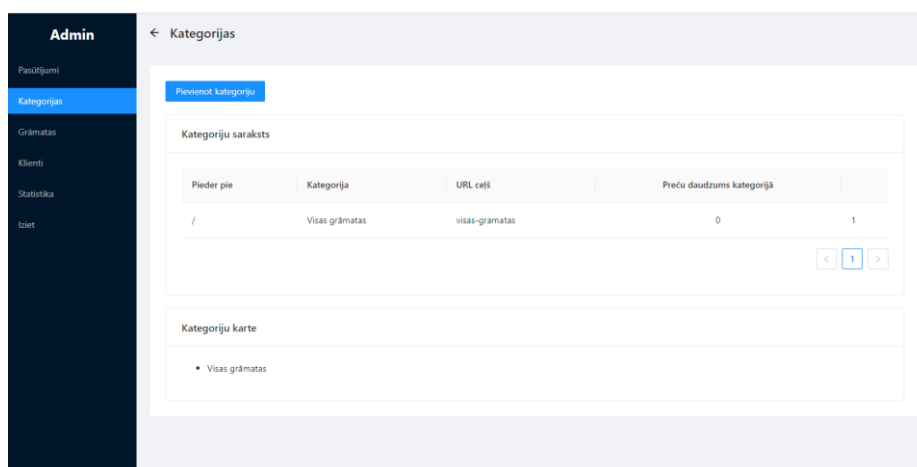
Lai apskatīt, pievienot vai rediģēt kategorijas pārejiet uz kategoriju sadaļu, attēls 2. att, sānu navigācijas joslā nospiežot pogu “Kategorijas”.

Atvērsies kategoriju sadaļu, kurā jūs redzēsiet kategoriju sarakstu tabulā un esošo kategoriju koku zemāk.

2. pielikuma turpinājums

Lai pievienotu kategoriju nospiediet pogu zilā krāsā ar uzrakstu “Pievienot kategoriju”

Lai rediģēt esošas kategorijas, uzklikšķiniet uz nepieciešamās kategorijas tabulas rindas.



2. att. Kategoriju atvērums (autora veidots ekrānšāviņš)

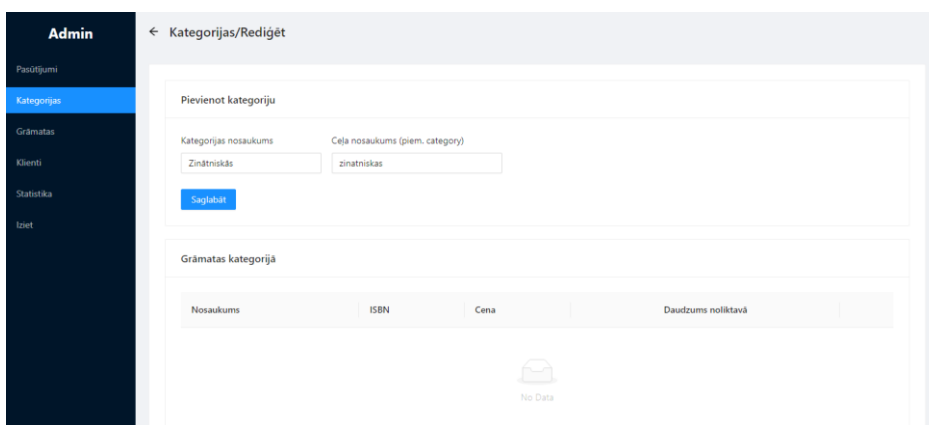
Sadaļā pievienot jaunu kategoriju, ir nepieciešams izvēlēties kategoriju, kura būs augšskategorija, ierakstīt kategorijas nosaukumu un ceļa nosaukumu.

Kategorijas un ceļa nosaukumam nav jābūt unikālam. Ceļa nosaukumā, lūdzu ievadiet tekstu bez garumzīmēm, atstarpēm un specifiskiem simboliem, vēlam, lai tas būtu tuvs nosaukumam, tas ir nepieciešams, labākai meklētājprogrammu optimizācijai interneta veikalā.

Kad vēlamās izmaiņas ir veiktas spiediet saglabāt. Veiksmīgas saglabāšanas gadījumā, jūs tiksiet pārdresēts uz sarakstu.

Rediģēšana līdzīgi kā saglabāšana ļauj ierakstīt/izmainīt nosaukumu un ceļu, taču neļauj mainīt augšskategoriju, tas ir laicīgs ierobežojums šai programmas versijai.

Atverot rediģēšanas formu, zem tās būs produktu saraksts, kurš atbilst rediģējamai kategorijai. Forma un saraksta sadaļā redzama 3. att.



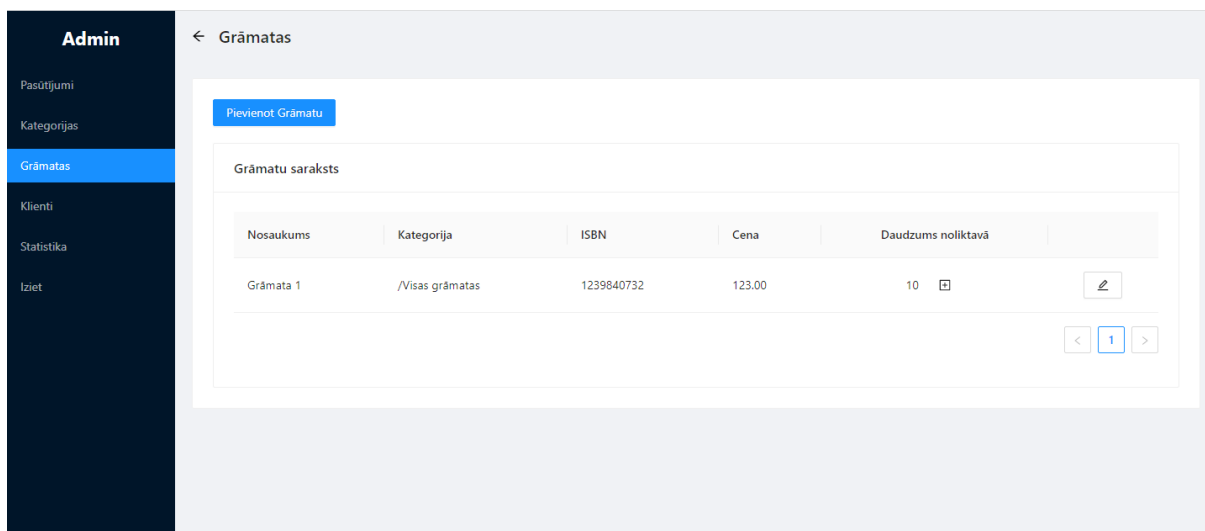
3. att. Kategorijas rediģēšanas forma (autora veidots ekrānšāviņš)

2. pielikuma turpinājums

Grāmatu (produktu) pārvaldīšanas modulis

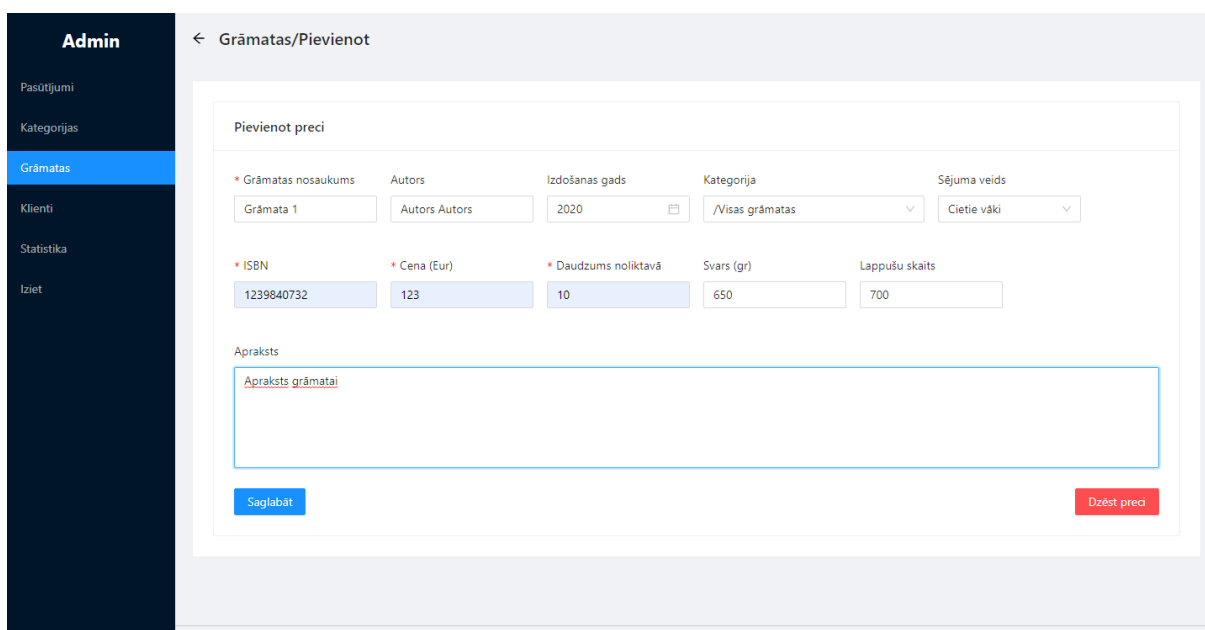
Lai apskatīt, pievienot vai rediģēt grāmatas pārejiet uz Grāmatu sadaļu, attēls 4. att, sānu navigācijas joslā nospiežot pogu “Grāmatas”.

Atvērsies grāmatu sadaļa, kurā jūs redzēsiet visu grāmatu sarakstu tabulā.



4. att. Grāmatu atvērums (autora veidots ekrānšāviņš)

Lai pievienotu grāmatu nospiediet pogu Pievienot grāmatu zilā krāsā, atvērsies grāmatas pievienošanas forma. Pēc lauku aizpildīšanas apstipriniet pievienošanu ar pogu Saglabāt. Forma apskatāma 5. att.



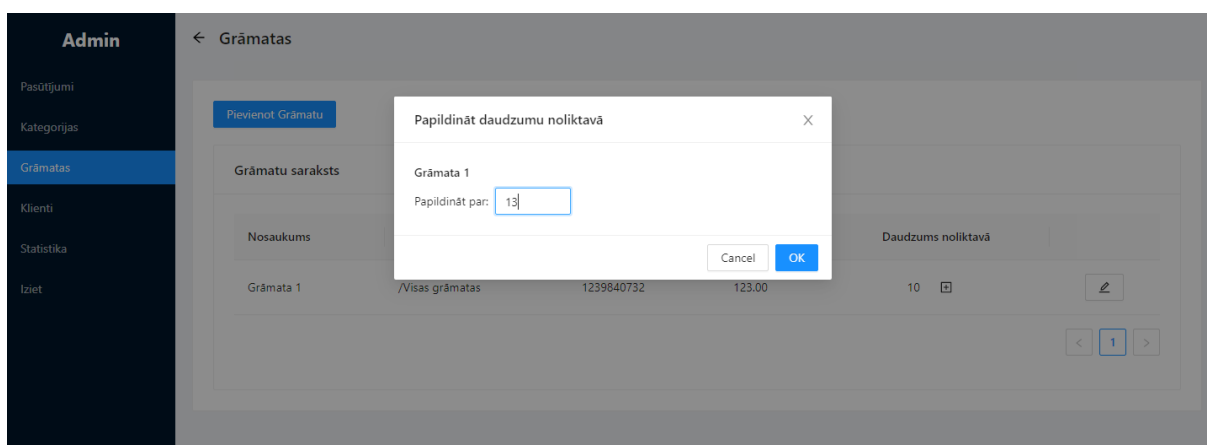
5. att. Grāmatas pievienošanas forma (autora veidots ekrānšāviņš)

2. pielikuma turpinājums

Lai rediģētu grāmatu, nospiediet grāmatu sarakstā pogu labajā kolonnā (zīmuļa ikona), atvērsies forma līdzīga jaunas Grāmatas pievienošanai, izņemot to, ka nebūs iespējams korigēt daudzumu noliktavā un būs papildus iespēja dzēst preci.

Preces daudzuma papildināšanai, kolonnā daudzums, nospiediet pogu ar “+” ikonu, atvērsies uzpeldošais logs, ar iespēju ievadīt – par cik vienībām papildināt grāmatu daudzumu, apskatāms 6. att.

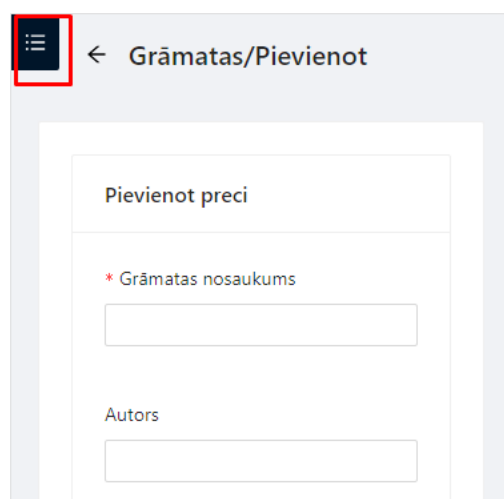
Apstipriniet ar OK.



6. att. Grāmatu daudzuma papildināšana (autora veidots ekrānšāviņš)

Mobilā versija

Lietotnes versijā uz maziem ekrāniem nav būtisku izmaiņu elementu atrašanās vietā un funkcionalitātē. Uz maziem ekrāniem tiek paslēpta navigācijas josla, kuru var parādīt vai noslēpt izmantojot pogu, 7. att.



7. att. Mobilā skata navigācijas paslēpšanas poga (autora veidots ekrānšāviņš)

Viedokļu realizācijas prioritāšu tabula

Nr.	Viedoklis	Prioritāte	Komentāri
1.	Saskarnes izstrāde	Augsta	Izstrādāt funkcionējošu lietotāja saskarni datora ekrāniem.
2.	Saskarnes pielāgošana maziem ekrāniem	Vidēja	Pielāgot saskarni maziem ekrāniem. Pirmajā ciklā tas nav tik nozīmīgi.
3.	Autorizācijas izveide	Augsta	Izveidot autorizācijas moduli, piekļuves ierobežošanai, lietotājs - Administrātors
4.	Kategoriju pievienošanas un pārvaldīšanas izvēlne	Augsta	Izveidot iespēju apskatīt, pievienot, labot kategorijas un apakškategorijas.
5.	Pievienot iespēju dzēst kategorijas	Vidēja	Pirmajā izstrādes ciklā, varētu atteikties no šīs funkcionalitātes.
6.	Preču pievienošanas un pārvaldīšanas izvēlne	Augsta	Izveidot iespēju apskatīt, pievienot un labot preces. Preces jāpiesaista kategorijai.
7.	Statistikas sadaļas izstrāde	Zema	Šī moduļa izstrāde ir iespējama pēc pasūtījumu moduļa izstrādes
8.	Pasūtījumu sadaļas izstrāde	Zema	Šī moduļa izstrāde ir iespējama pēc pirkumu moduļa izstrādes
9.	Klientu bāzes sadaļas izstrāde	Zema	Šī moduļa izstrāde ir iespējama pēc pirkumu un lietotāju moduļa izstrādes

Funkcionālās un nefunkcionālās prasības

Funkcionālās prasības		
	Prasība	Funkcijas
1.	Parādīt kategoriju sarakstu	<ul style="list-style-type: none"> • Atlasīt kategorijas • Saņemt datus no servera • Ievietot datus tabulā
2.	Parādīt preču sarakstu	<ul style="list-style-type: none"> • Atlasīt preces • Saņemt datus no servera • Ievietot datus tabulā
3.	Pievienot jaunu preci	<ul style="list-style-type: none"> • Validēt formas laukus • Nosūtīt datus uz serveri • Saglabāt datus
4.	Pievienot jaunu kategoriju	<ul style="list-style-type: none"> • Validēt formas laukus • Nosūtīt datus uz serveri • Saglabāt datus
5.	Labot preci	<ul style="list-style-type: none"> • Izvēlēties preci tabulā • Atvērt formu un aizpildīt formas laukus • Validēt jaunus datus • Nosūtīt uz serveri • Saglabāt datus
6.	Labot kategoriju	<ul style="list-style-type: none"> • Izvēlēties kategoriju tabulā • Atvērt formu un aizpildīt formas laukus • Validēt jaunus datus • Nosūtīt uz serveri • Saglabāt datus
7.	Pievienot precei kategoriju	<ul style="list-style-type: none"> • Saņemt no servera kategoriju sarakstu • Preces pievienošanas vai labošanas formā, parādīt sarakstu • Ļaut izvēlēties no saraksta vienu vai vairākas kategorijas • Saglabāt preci

4. pielikuma turpinājums
tabulas turpinājums

8.	Aizsargāt piekļuvi ar paroli un lietotāja piekļuves vārdu	<ul style="list-style-type: none"> • “Šajā posmā lietotājs būs mākslīgi pievienots DB” • Validēt formas ievadītos datus • Nosūtīt uz datubāzi • Saņemt atbildi
9.	Izrakstīties no sistēmas	<ul style="list-style-type: none"> • Nosūtīt uz serveri • Izrakstīt lietotāju no datubāzes
10.	Uzglabāt informāciju par kategorijām	<ul style="list-style-type: none"> • Dati glabājās datubāzē
11.	Uzglabāt preču bāzi	<ul style="list-style-type: none"> • Dati glabājās datubāzē
Nefunkcionālās prasības		
	Prasība	Validācija
1.	Ātrdarbība	<ul style="list-style-type: none"> • Pieprasīt ierobežotu datu daudzumu no servera • Dalīt lappusēs
2.	Intuitīva lietojamība	<ul style="list-style-type: none"> • Minimāls soļu daudzums darbības veikšanai • Pēc iespējas mazāk dalīt formas cilnēs. • Pievienot informatīvus aprakstus katram laukam.
3.	Pieejamība uz dažāda izmēra ekrāniem	<ul style="list-style-type: none"> • Mazākais iespējamais ekrāna platums 320px, lielākais 1980px
4.	Aizsardzība piekļuvei	<ul style="list-style-type: none"> • Ierakstīšanās forma
5.	Paplašināmība	

Galvojums

Ar šo es Dmitrijs Jaunslavietis galvoju, ka studiju / kvalifikācijas darbs

(vārds uzvārds)

(vajadzīgo pasvītrot)

“TĪMEKĻA VIETNES IZSTRĀDE GRĀMATU IZDEVĒJIEM”

(darba nosaukums)

ir izstrādāts patstāvīgi, tajā nav pieļauts citu personu intelektuālā īpašuma tiesību pārkāpums vai plaģiāts – citas personas radošās darbības rezultātu tālākā paušana savā vārdā. No citiem avotiem ņemtajiem darbiem, definējumiem un citātiem darbā ir uzrādītas atsauces. Izmantoti citu autoru pētījumu rezultāti un datu avoti ir norādīti atsaucēs. Darbs nekad nav publicēts un pirmo reizi tiek iesniegts aizstāvēšanai

Studiju darbu aizstāvēšanas komisijā / Valsts noslēguma pārbaudījuma komisijā.

(vajadzīgo pasvītrot)

Apliecinu, ka Alberta koledžas elektroniskajā sistēmā augšupielādētā darba teksts ir identisks papīra formātā iesniegtā darba tekstam.

_____ / _____ Dmitrijs Jaunslavietis _____ /

(studējošā paraksts)

(vārds, uzvārds)

2022. gada _____