RĪGAS VALSTS TEHNIKUMS

DATORIKAS NODAĻA

Izglītības programma: Programmēšana

KVALIFIKĀCIJAS DARBS

“Personīga dizaina preču pasūtīšanas un iegādes tīmekļa lietojumprogramma"

Paskaidrojošais raksts 91.lpp.

Audzēknis: Edvards Torsters-Makauskis

Prakses vadītājs: Oksana Roslova

Nodaļas vadītājs: Normunds Barbāns

Rīga 2024

ANOTĀCIJA

Šī kvalifikācijas darba mērķis ir aprakstīt un analizēt personīga dizaina preču pasūtīšanas un iegādes tīmekļa lietojumprogrammas izstrādes procesu. Tīmekļa lietojumprogramma ir domāta, lai nodrošinātu funkcionalitāti, kas atvieglo darbplūsmu personīga dizaina preču pasūtīšanā un iegādē, un to izmanto sistēmas klienti. Lietojumprogramma tiek izstrādāta, izmantojot Python un JavaScript programmēšanas valodas, kā arī HTML/CSS un MySQL valodas datu bāzes struktūras izveidei.

Kvalifikācijas darbs ietver ievadu, uzdevuma formulējumu, izvēlēto risinājumu pamatojumu, programmatūras produkta modelēšanu un dizainu, kā arī datu struktūras aprakstu, lietotāja rokasgrāmatu, secinājumus un informācijas avotus.

Ievadā tiek aprakstīta problēma un sistēmas nozīme un aktualitāte. Uzdevuma formulējums sniedz īsu aprakstu galvenajām uzdevumām un sistēmas funkcionalitātei. Prasību specifikācija ietver ievades un izvades informācijas aprakstus, kā arī prasības funkcionālajām un nefunkcionālajām prasībām. Izvēlēto risinājumu pamatojumā tiek izskaidrotas izvēlēto tehnoloģiju priekšrocības. Programmatūras modeļu un dizaina aprakstā iekļauts sistēmas struktūras modelis ar sistēmas arhitektūru, ER sistēmas modelis un funkcionalitātes sistēmas modelis ar datu plūsmas modeļiem. Datus struktūras aprakstā tiek aprakstīti visi datu bāzes tabulu lauki, to datu tipi un primārie un ārējie atslēgas.

Lietotāja rokasgrāmata ietver sistēmas prasības aparatūrai un programmatūrai, tās instalācijas un darbības uzsākšanas norādījumus, kā arī programmas aprakstu un testa piemēru. Šī dokumentācija sastāv no 91 lapaspusēm, 36 bildes ,13 tabulām un 5 pielikumiem.

ANOTATION

The aim of this qualification work is to describe and analyze the process of developing a web application for ordering and purchasing personalized design products. The web application is intended to provide functionality that facilitates the workflow of ordering and purchasing personalized design products, and it is used by the system's clients. The application is developed using Python and JavaScript programming languages, as well as HTML/CSS and MySQL languages for database structure.

The qualification work includes an introduction, task statement, justification of the chosen solution, modeling and design of the software product, description of data structure, user manual, conclusions, and information sources.

The introduction describes the problem and the significance and relevance of the system. The task statement provides a brief overview of the main tasks and functionality of the system. The requirements specification includes descriptions of input and output information, as well as functional and non-functional requirements. The justification of the chosen solution explains the advantages of the selected technologies. The software modeling and design description include a system structure model with system architecture, an ER system model, and a functional system model with data flow models. The description of data structure describes all the fields of the database tables, their data types, and primary and foreign keys.

The user manual includes system requirements for hardware and software, installation and startup instructions, as well as a program description and test example. This documentation consists of 91 pages, 36 images, 13 tables, and 5 appendices.

SATURS

IEVADS

Mūsdienu digitālajā laikmetā personalizēti dizaina izstrādājumi ir kļuvuši arvien populārāki, ļaujot indivīdiem un uzņēmumiem izveidot unikālus produktus, kas atspoguļo viņu personību un zīmolu identitāti. Šī kvalifikācijas darba mērķis ir izstrādāt tīmekļa lietojumprogrammu, kas ļautu lietotājiem ērti un intuitīvi izveidot, pasūtīt un iegādāties personalizētas dizaina preces, piemēram, apģērbu un aksesuārus.

Izstrādātā sistēma piedāvā plašu funkcionalitātes klāstu, tostarp viegli lietojamu dizaina rīku, drošu maksājumu sistēmu un iespēju pieslēgties, izmantojot sociālos tīklus. Projekta ietvaros tiks izmantotas dažādas tehnoloģijas, piemēram, Python un JavaScript programmēšanas valodas, kā arī HTML/CSS un MySQL datu bāzes struktūru veidošanai.

Darbs ietver problēmas aprakstu, sistēmas nozīmīgumu un aktualitāti, uzdevuma formulējumu, izvēlēto risinājumu pamatojumu, programmatūras modelēšanu un dizainu, datu struktūras aprakstu, lietotāja rokasgrāmatu, secinājumus un informācijas avotus. Ievadā tiek aprakstīta problēma, sistēmas nozīmīgums un aktualitāte, uzdevuma formulējums sniedz īsu aprakstu galvenajām uzdevuma sastāvdaļām un sistēmas funkcionalitātei.

1. UZDEVUMU NOSTĀDNE

Kvalifikācijas darba uzdevums ir izveidot tiešsaistes dizaina platformu. Mājaslapa būs paredzēta lietotājiem, kuri vēlas izveidot un pasūtīt personalizētas dizaina preces, piemēram, apģērbu, aksesuārus. Sistēmā būs jānodrošina iespēja izveidot un pasūtīt dizaina izstrādājumus, kā arī jābūt drošai un ērtai maksāšanas sistēmai. Turklāt, lietotājiem būs iespēja pieslēgties, izmantojot sociālos tīklus, lai padarītu lietošanu vēl ērtāku. Administrācija veiks regulāru pārbaudi, lai nodrošinātu augstu kvalitāti un klientu apmierinātību, pirms izstrādājumi tiek pievienoti mājaslapas veikalā pārdošanai.

Ir plānotas vairākas funkcijas:

• **Reģistrācija un Pieteikšanās**: Lietotājiem būs jāreģistrējas un jāizveido konts, lai piekļūtu visām platformas funkcijām un veiktu pasūtījumus. Pieteikšanās un reģistrēšanās process būs vienkāršs un ātrs, nodrošinot lietotājiem ērtu un drošu pieredzi.

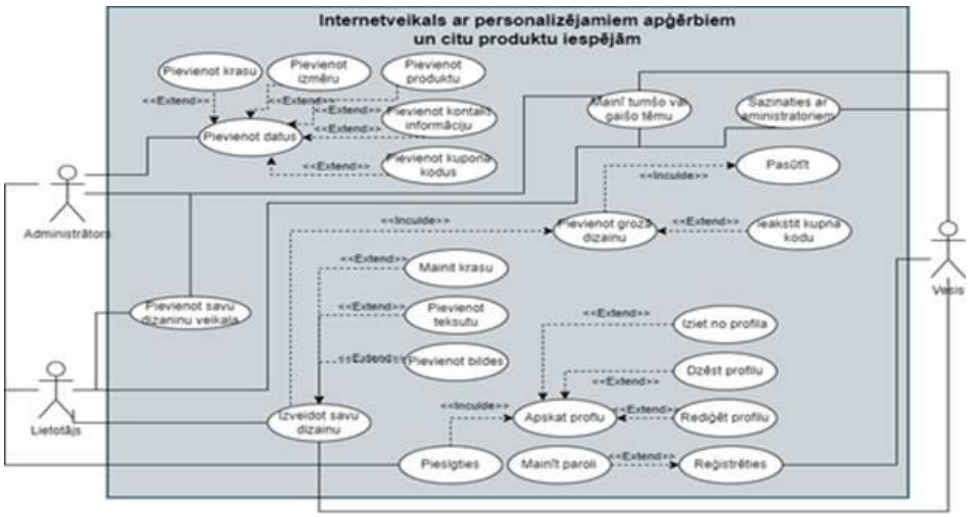
• **Dizaina Rīki:** Lietotājiem būs pieejams intuitīvs un viegli lietojams rīks, lai izveidotu un pielāgotu dizainus savām vēlmēm. Tas ietvers dažādas funkcijas, piemēram, krāsu izvēli, formas rediģēšanu un tekstapstrādi.

• **Atlaižu Sistēma:** Platforma piedāvās klientiem iespēju saņemt atlaidi par pasūtījumiem atkarībā no dažādiem kritērijiem, piemēram, pasūtījuma apjoma vai lojalitātes programmas dalībniecības.

• **Lietotāja Profila Pārvaldība:** Profila rediģēšana, paroles atiestatīšana, konta dzēšana.

• **Preču Pārvaldība**: Preču pievienošana un rediģēšana, pieejamo krāsu un izmēru pārvaldība.

• **Pasūtījumu Pārvaldība:** Preču pievienošana grozam, groza satura rediģēšana, pasūtījumu noformēšana un apstrāde.



1.att Lietojumgadījuma diagramma

2. PRASĪBU SPECIFIKĀCIJA

2.1. Ieejas un izejas informācijas apraksts

2.1.1. Ieejas informācijas apraksts

Sistēmā tiks nodrošināta šādas ieejas informācijas apstrāde.

Informācija par **Lietotājs** sastāvēs no šādiem datiem.

- E-pasts – burtu teksts ar izmēru līdz 254 rakstzīmēm, unikāls.
- Vārds – burtu teksts ar izmēru līdz 150 rakstzīmēm.
- Uzvārds – burtu teksts ar izmēru līdz 150 rakstzīmēm.
- Administrātors – loģiska vērtība, patiess vai nepatiess.
- Parole – burtu teksts ar izmēru līdz 128 rakstzīmēm.
- Lietotājvārds – burtu teksts ar izmēru līdz 150 rakstzīmēm, unikāls.
- Telefona numurs – burtu teksts ar izmēru līdz 20 rakstzīmēm, unikāls.

Informācija par lietotāju tiks izveidota kad reģistrējas.

Informācija par **Produkts** sastāvēs no šādiem datiem.

- Virsraksts – burtu teksts ar izmēru līdz 100 rakstzīmēm, unikāls.
- Bilde – .
- URL segmenta nosaukums – burtu teksts ar izmēru līdz 50 rakstzīmēm, unikāls.
- Cena – decimals skaitļu vērtība, kur aiz komata var but 2 skaitļi.
- Priekšēja bilde ar fonu –
- Priekšēja bilde bez fona –
- Aizmugureja bilde ar fonu –
- Aizmugureja bilde bez fona –

Informācija par produktu izveidosies kad administrators pievienos viņu, un tad lietoājs varēs dizainot šo produktu.

Informācija par **Krāsa** sastāvēs no šādiem datiem.

- Nosaukums – burtu teksts ar izmēru līdz 100 rakstzīmēm, unikāls.
- Krāsas kods – burtu teksts ar izmēru līdz 25 rakstzīmēm.

  Informācija par krāsu izveidosies kad administrators pievienos viņu, un administrātors pievienos produktam šo krāsu, un tad lietotājs varēs izmantot ievietoto krāsu.

Informācija par **Izmērs** sastāvēs no šādiem datiem.

- Nosaukums – burtu teksts ar izmēru līdz 100 rakstzīmēm, unikāls.
- Izmērs – burtu teksts ar izmēru līdz 10 rakstzīmēm.

  Informācija par izmēru izveidosies kad administrators pievienos viņu, un administrātors pievienos produktam šo krāsu, un tad lietotājs varēs izvēlēties ievietoto izmēru.

Informācija par **Pielāgots dizains** sastāvēs no šādiem datiem.

- Virsraksts – burtu teksts ar izmēru līdz 180 rakstzīmēm.
- Apraksts – garais buru teksts.
- Papildu piezīmes – garais buru teksts.
- Bilde –

  Informācija par Pielāgotu dizainu izveidosies kad administrators pievienos viņu, var tikai vienu reizi pievienot, un tad tikai varēs rediģēt. Lietotāji to redzēs sakumlpā.

Informācija par **Ziņojumi** sastāvēs no šādiem datiem.

- Vārds – burtu teksts ar izmēru līdz 100 rakstzīmēm.
- Uzvārds – burtu teksts ar izmēru līdz 100 rakstzīmēm.
- E-pasts – burtu teksts ar izmēru līdz 254 rakstzīmēm.
- Telefona numurs – burtu teksts ar izmēru līdz 20 rakstzīmēm.
- Vēstule – garais buru teksts.
- Atbilde – loģiska vērtība, patiess vai nepatiess, automatiski nosaka.
- Administratora virsraksts – burtu teksts ar izmēru līdz 255 rakstzīmēm.
- Administratora ziņa – garais buru teksts.

  Informācija par ziņojumu izveidosies kad lietotājs uzrakstīs kaut ko ‘sazinaties ar mums’, un tad administrators ievadot lauka ‘virsraksts, ziņa', nosūta atbildi lietotājam uz e-pastu.

Informācija par **Reitings** sastāvēs no šādiem datiem.

- Zvaigznes – skaitļu vērtība lidz 5 skaitļiem.

  Informācija par reitingu izveidosies kad lietotājs izveidos dizanu un viņa dizainu kads iedos vērtējumu.

Informācija par **Kontakti** sastāvēs no šādiem datiem.

- Adrese – burtu teksts ar izmēru līdz 255 rakstzīmēm.
- Pasta indeks – burtu teksts ar izmēru līdz 20 rakstzīmēm.
- E-pasts – burtu teksts ar izmēru līdz 254 rakstzīmēm.
- Telefona numurs – burtu teksts ar izmēru līdz 20 rakstzīmēm.
- Twiter saite – burtu teksts ar izmēru līdz 200 rakstzīmēm.
- Facebook saite – burtu teksts ar izmēru līdz 200 rakstzīmēm.
- Instagram saite – burtu teksts ar izmēru līdz 200 rakstzīmēm.

  Informācija par kontaktiem izveidosies kad administrators pievienos viņu, var tikai vienu reizi pievienot, un tad tikai varēs rediģēt. Lietotāji to redzēs 'Par mums' lapā .

Informācija par **Produkta saraksts** sastāvēs no šādiem datiem.

- Virsraksts – burtu teksts ar izmēru līdz 255 rakstzīmēm.
- Apraksts – garais buru teksts.
- Priekšēja bilde –
- Aizmugurējā bilde –

  Informācija par Produkta saraksts izveidosies kad lietotājs izveidos dizanu kadām produktam, un viņš gribēs dalīties, un ja administrators atļaus publicēt, tad visi viti cilvēki redzēs šo produktu.

Informācija par **Davanu karte** sastāvēs no šādiem datiem.

- Kods – burtu teksts ar izmēru līdz 50rakstzīmēm, unikāls.
- Aktivizēts – loģiska vērtība, patiess vai nepatiess, automatiski nosaka.
- Atlaides veids – burtu teksts ar izmēru līdz 20 rakstzīmēm.
- Atlaides summa – decimals skaitļu vērtība, kur aiz komata var but 2 skaitļi.
- Minimāla pasutijuma summa– decimals skaitļu vērtība, kur aiz komata var but 2 skaitļi.
- Sākum datums – datums.
- Daudzums – vesela skaitļa vērtība
- Neierobežots – loģiska vērtība, patiess vai nepatiess, automatiski nosaka.

  Informācija par davanu karti izveidosies kad administrators pievienos viņu, un tad lietotāji varēs izmantot šo kodu.

2.1.2. Izejas informācijas apraksts

- **E-pasta paziņojumi (verifikācija):** Lietotāji kas reģistrējās, atsutas uz lietotāja norādīto e-pasta, epasta verifikāciju, kur atsutas saite, kura ieejot lietotājs verificē savu profilu.
- **Lapas paziņojumi:** Mājaslapā tiek izmantoti lapas paziņojumi. Piemēram, ka tika nosūtita vēstule, ka kaut kas saglabājas, utt.
- **Tumšā vai gaišā fona maiņa:** Pielāgojiet veikala izskatu, izvēloties tumšo vai gaišo fonu.
- **Lokalizācija:** Varēs mainīt mājaslpas valodu, būs pieejami divas valodas (Angļu un Latviešu)

2.2. Funkcionālās prasības

1. Pieteikšanās un lietotāju pārvaldība
   1.1. Lietotājiem ir jābūt iespējai reģistrēties un izveidot kontu, norādot nepieciešamos personas datus.
   1.2. Sistēmā jābūt iespējai pieteikties ar esošo kontu, izmantojot lietotājvārdu un paroli.
   1.3. Administrātoriem jāvar pārvaldīt lietotāju kontus, ieskaitot paroles atiestatīšanu un bloķēšanu.
2. Dizaina rīka pārvaldība
   2.1. Administrātoriem jābūt iespējai pievienot jaunas preces, kas ir pieejamas dizainēšanai, un rediģēt esošo preču informāciju.
   2.2. Jābūt iespējai pārvaldīt krāsu paleti, pieejamos fontus un citus dizaina elementus.
   2.3. Jābūt iespējai pievienot vai rediģēt dizaina veidošanā izmantojamās funkcijas un rīkus.
3. Dizaina rīku funkcionalitāte
   3.1. Lietotājiem jāvar veidot un pielāgot dizainus, izmantojot pieejamos rīkus, piemēram, tekstu, attēlus, formu elementus utt.
   3.2. Jābūt iespējai saglabāt un dalīties ar izveidotajiem dizainiem citiem lietotājiem.
   3.3. Jābūt iespējai apskatīt un rediģēt saglabātos dizainus lietotāja profilā.
4. Sistēmas administrēšana
   4.1. Administrātoriem jābūt iespējai pārvaldīt visus platformas lietotājus, tostarp izdzēst kontus un rediģēt profilu informāciju.
   4.2. Jābūt iespējai pārvaldīt platformas satura un preču informāciju, tostarp pievienot jaunas preces un rediģēt esošo produktu datus.

4.3. Jābūt iespējai veikt sistēmas konfigurāciju un pielāgojumus, piemēram, sākumlapas teksta maiņu un tematu izvēli.

5. Dizains un lietotāja pieredze
   5.1. Jābūt intuitīvam un viegli izmantojamam lietotāja interfeisam, kas ļauj viegli pārvietoties pa platformu un izmantot pieejamās funkcijas.
   5.2. Dizainam jābūt pievilcīgam un estētiski patīkamam, lai veicinātu lietotāja iesaistīšanos un pozitīvo pieredzi.
   5.3. Jābūt iespējai pielāgot platformas izskatu un izvēlēties starp tumšo un gaišo tēmu, atbilstoši lietotāja preferencei.
6. Datu drošība
   6.1. Jābūt nodrošinātai lietotāju datu konfidencialitātei, tostarp personas datu, maksājumu informācijas un dizaina projektu aizsardzībai.
   6.2. Jābūt implementētai drošai autentifikācijas sistēmai, kas pasargā sistēmu no neautorizētas piekļuves.
   6.3. Jābūt regulāriem datu rezerves kopiju veidošanas procesiem, lai nodrošinātu datu atjaunojamību un nepieciešamības gadījumā to atjaunošanu.

2.3. Nefunkcionālās prasības

1. Drošības prasības:

   1.1. Datubāzes drošība: Jābūt implementētai drošai datubāzes aizsardzībai, lai novērstu neautorizētu piekļuvi un datu noplūšanu.

   1.2. Datu šifrēšana: Jānodrošina jutīgu datu šifrēšana, lai pasargātu to no neautorizētas piekļuves un izmantošanas.

   1.3. Brīdinājumu sistēma: Jābūt sistēmai, kas automātiski brīdina par iespējamām drošības pārkāpumu vai nelikumīgu darbību mēģinājumiem.

2. Ātrdarbības prasības:

   2.1. Atbildes laiks: Platformas atbildes laiks uz lietotāju pieprasījumiem jābūt mazākam par 2 sekundēm, lai nodrošinātu lietotāja pieredzes fluiditāti.

   2.2. Servera stabilitāte: Jābūt nodrošinātai sistēmas stabilitātei un augstai pieejamībai, lai izvairītos no servera pārslodzes un izkrišanas gadījumiem.

2.3. Datu pārsūtīšanas ātrums: Lietotāja izmaiņu saglabāšanas un atjaunināšanas procesiem jābūt ātriem un efektīviem, lai nodrošinātu lietotāju darbību plūsmas nevainojamu darbību.

3. Lietotāja saskarnes prasības:

3.1. Intuitīva navigācija: Lietotāja saskarnē jābūt intuitīvai un viegli saprotamai, lai lietotāji varētu ērti pārvietoties pa platformu un piekļūt funkcijām.

3.2. Paziņojumi un norādes: Lietotāja saskarnē jābūt skaidriem un saprotamiem paziņojumiem un norādēm, lai palīdzētu lietotājiem veikt nepieciešamās darbības un izvairītos no kļūdām.

4. Atbalsta un uzturēšanas prasības:

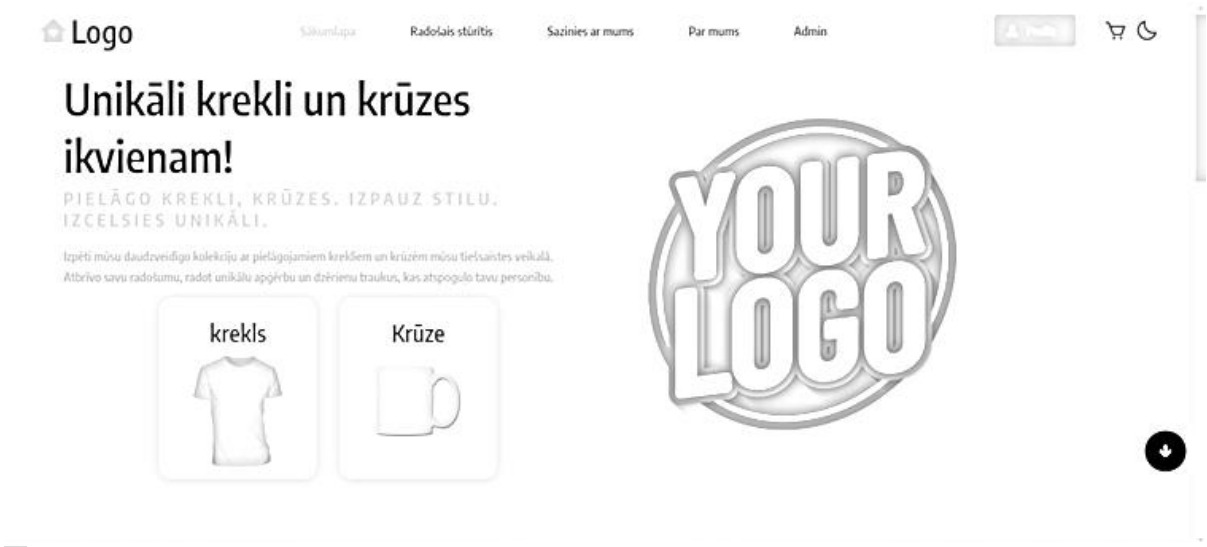
4.1. Regulāra sistēmas uzturēšana: Nepieciešama regulāra sistēmas uzturēšana un atjaunināšana, lai novērstu kļūdas un nodrošinātu platformas stabilitāti un drošību.

4.2. Atjauninājumu paziņojumi: Lietotājiem jāsaņem paziņojumi par jebkuriem sistēmas atjauninājumiem vai uzturēšanas darbiem, kas varētu ietekmēt to darbību.

5. Pieejamības prasības:

5.1. Platformas saderība: Jānodrošina platformas saderība ar dažādām ierīcēm un pārlūkiem, lai nodrošinātu pieejamību plašam lietotāju lokam.

5.2. Ātrums un efektivitāte: Platformas ielādes laikam jābūt minimālam, lai nodrošinātu ātru un efektīvu piekļuvi satura un funkcijām

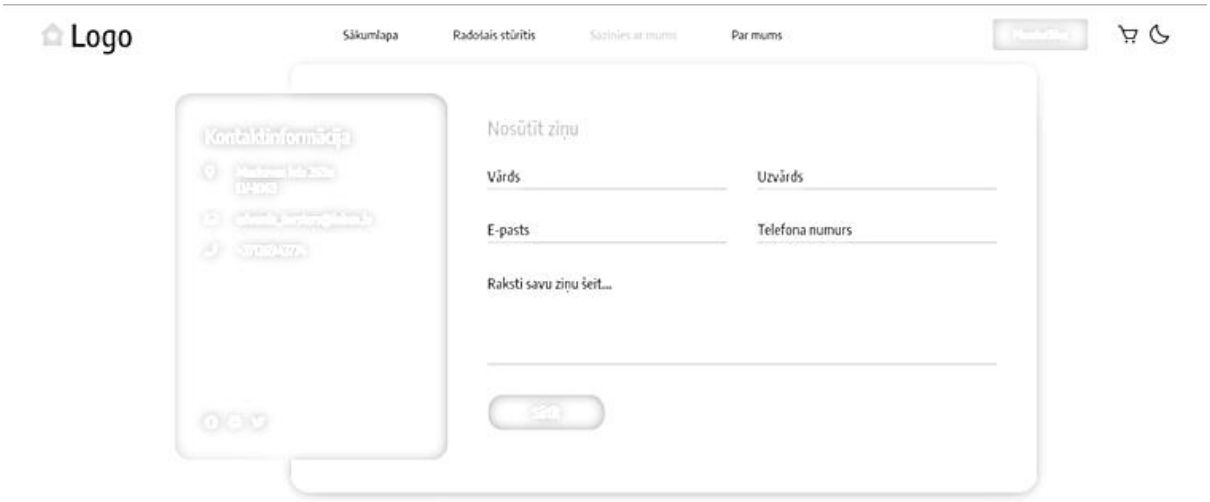


2.att. Sākum lapa

2. Attēlā redzama internetveikala sākumlapa, kur izceltas galvenās funkcijas un sadaļas. Augšpusē izvietota navigācijas josla ar svarīgākajām saitēm. Kreisajā pusē atrodas virsraksts, apraksts un papildus piezīmes, kā arī logi ar precēm, uz kurām ir iespējams veidot savus dizainus. Labajā pusē redzams uzņēmuma logo. (skat. 2. att.).

3.att. Dizain rīka lapa

3. Attēlā Dizaina rīka lapā redzami trīs galvenie lauki. Kreisajā laukā atrodas produkts, kuru dizaino, centrālajā logā ir dizaina rīks ar visām nepieciešamajām funkcijām dizaina veidošanai, bet labajā logā ir taka navbar ar funkcionalitātēm, kuras var izvēlēties un lietot, to nospiestot. Kad tiek izvēlēta kāda no funkcijām, centrālajā logā parādās atbilstošais rīks vai interfeiss šīs funkcionalitātes izmantošanai dizaina procesā. (skat. 3. att.).



4.att. Sazināties ar mums lapa

4. Attēlā saziņas lapa piedāvā divus savienotus logus: kreisajā mazajā logā ir kontaktinformācija, kurā iekļauta lokācija, e-pasta adrese un tālruņa numurs, kā arī ikonas, kas nodrošina saites uz mūsu sociālajiem tīkliem, bet galvenajā logā lietotājiem ir iespēja nosūtīt ziņu administratoram, ievadot savu vārdu, uzvārdu, e-pasta adresi, tālruņa numuru un ziņu, kuru vēlaties nosūtīt, nodrošinot ērtu un vienkāršu veidu, kā sazināties ar mums. (skat. 4. att.).

3. UZDEVUMU RISINĀŠANA LĪDZEKĻU IZVĒLES PAMATOJUMS

Sistēmas tipa skaidrojums: Internetveikals ar personalizējamiem apģērbiem un citu produktu iespējām izstrādē tiks izmantota tīmekļa lietojumprogrammas sistēma, kas darbosies pārlūkprogrammās un mobilajās ierīcēs. Šī sistēma būs veidota, izmantojot Python Django kā backend risinājumu.

Sistēmas daļas:

Sistēma sastāvēs no divām galvenajām daļām:

- Lietotāja daļā (fronend):
    - Atbildīga par spēlētāju saskarni un interaktīvo spēles pieredzi.
    - Tiks izstrādāta, izmantojot HTML, CSS un JavaScript.
- Servera Puse (Backend):
    - Nodrošinās spēles loģiku, saziņu ar datu bāzi un darbības, kas nepieciešamas spēles darbībai.
    - Tiks izstrādāta, izmantojot Python Django, kā arī tā veiks saziņu ar kamerām un mikrofoniem, nodrošinot multivides funkcionalitāti.

Pamatojums par izvēlēto tehnoloģiju

1.1 Funkcionāla apstrādes (back-end) tehnoloģija:

1.1.1 Nosaukums: Python Django

1.1.2 Versijas Numurs: 4.2.3

1.1.3 Īss Raksturojums: Python Django ir stabilas, drošas un veiksmīgas backend izstrādes platformas versija, kas piedāvā ērtu MVC struktūru.

1.2 Vizuāla un grafiskā dizaina (front-end) tehnoloģijas:

1.2.1 Nosaukums: HTML, CSS, JavaScript

1.2.3 Īss Raksturojums: Izmantos modernas tīmekļa tehnoloģijas, nodrošinot lietotājiem intuitīvu un pievilcīgu saskarni.

1.3 Datu Bāzes Izvēle

1.3.1 Nosaukums: MySql

1.3.2 Versijas Numurs: 8.0.4

1.3.3 Īss Raksturojums: MySQL ir attīstīts, ātrs un uzticams relāciju datu bāzes pārvaldības sistēma, kas ideāli piemērota spēles datu uzglabāšanai.

Katra aizgūta moduļa apraksts

1.1 Python Django

1.1.1 Kas, Ko Dara:

- Piedāvā backend struktūru un API, kas nodrošina spēles loģiku un saziņu ar datu bāzi.

1.1.2 Atsauce:

- Django dokumentācija: [https://docs.djangoproject.com/]

1.2 MySQL Datu Bāze

1.2.1 Kas, Ko Dara:

- Uzglabā spēles dati, tostarp spēlētāju informāciju, rezultātus un citus nepieciešamos datus.
- Nodrošina efektīvu datu pārvaldību un optimizētu datu atgriešanu.

1.2.2 Atsauce:

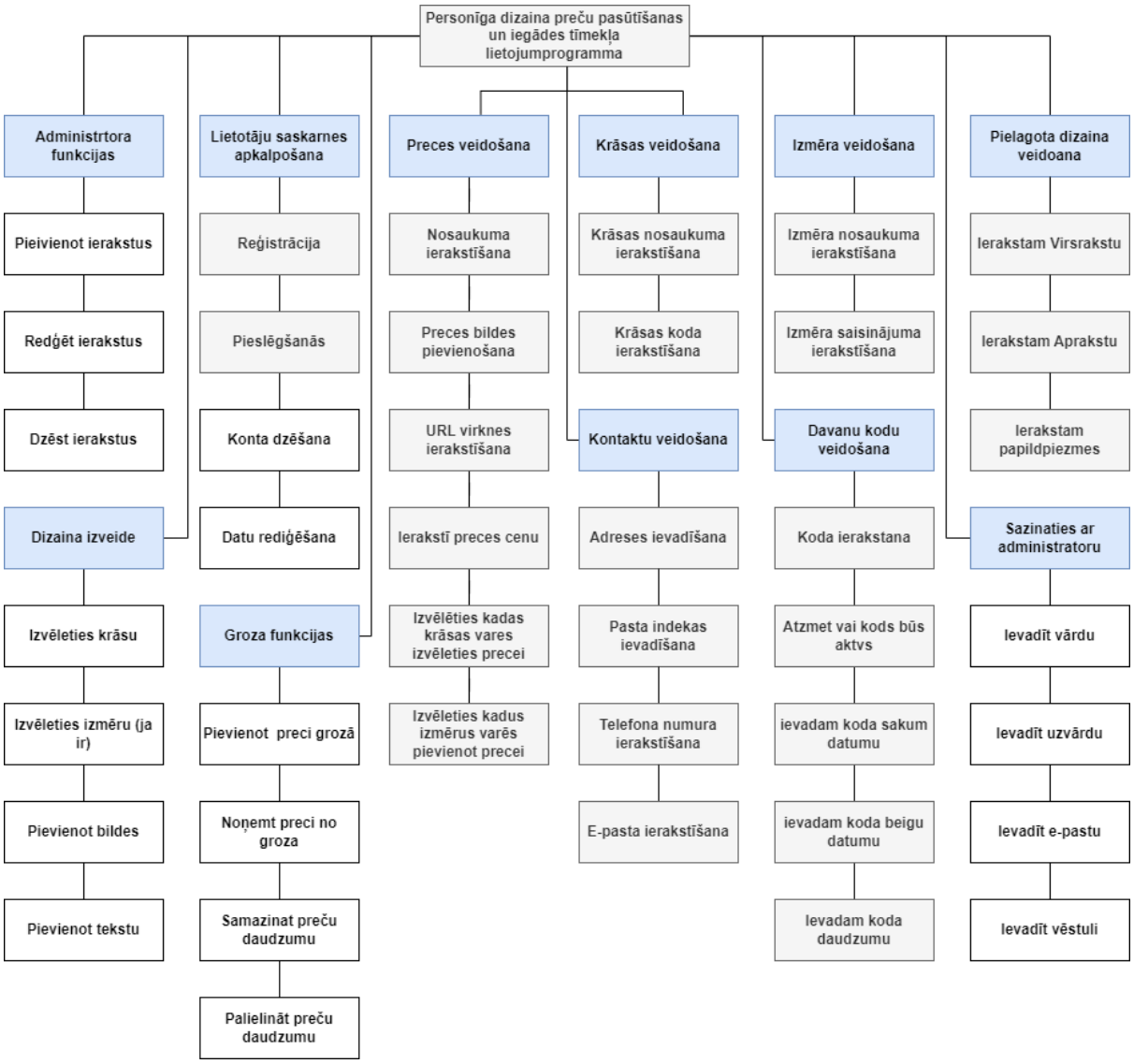
- MySQL dokumentācija: [https://dev.mysql.com/doc/]

4. PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA

4.1. Sistēmas struktūras modelis

4.1.1. Sistēmas arhitektūra

Sistēmas arhitektūras diagramma (skat. 5. att.). apraksta Personīga dizaina preču pasūtīšanas un iegādes tīmekļa lietojumprogramma galvenos komponentus, to mijiedarbību un funkcijas. Šeit ir apraksts svarīgākajiem elementiem:



5.att. Sistēmas arhitektūras diagramma

Lietotāju saskarnes apkalpošana:

- **Reģistrācija:** Sistēma piedāvā lietotājiem iespēju reģistrēties, sniedzot nepieciešamos datus un izvēloties lietotājvārdu un paroli.
- **Pieslēgšanās:** Lietotāji var pieslēgties savam kontam, ievadot lietotājvārdu un paroli.
- **Konta dzēšana**: Lietotājiem ir iespēja dzēst savu kontu, ja nepieciešams.
- **Datu reģistrēšana:** Lietotāji var pievienot vai atjaunināt savus datus, piemēram, e-pastu vai citus personiskos uzstādījumus.

Administrātora funkcijas:

- **Pievieno ierakstus:** Administrators var pievienot jaunus ierakstus par precēm, krāsām vai citiem svarīgiem elementiem.
- **Rediģēt ierakstus:** Administrators var labot esošos ierakstus, ja nepieciešams veikt izmaiņas.
- **Dzēst ierakstis**: Piedāvā iespēju administrātoram dzēst liekus vai nevajadzīgus ierakstus.

Preces veidošana:

- **Nosaukuma ierakstīšana:** Administrators ievada preci identificējošo nosaukumu, lai to viegli atpazītu un atrodot sistēmā.
- **Preces bildes pievienošana:** Administrators augšupielādē attēlu, lai vizuāli parādītu jauno preci un padarītu to pievilcīgāku pircējiem.
- **URL virknes ierakstīšana:** Sistēma automātiski vai pēc administratora norādījuma ģenerē un ieraksta URL, lai piekļūtu jaunajai precei tiešsaistē.
- **Ierakstīt preces cenu:** Administrators ievada cenu, par kuru šī prece tiks pārdota, pamatojoties uz tirgus analīzi un citiem faktoriem.
- **Izvēlēties kādas krāsas varēs izvēlēties precei:** Administrators norāda pieejamās krāsas, kuras ir pieejamas šai precei, lai pircēji varētu izvēlēties.
- **Izvēlēties kādus izmērus varēs pievienot precei:** Administrators norāda pieejamos izmērus, kuri ir pieejami šai precei, lai pircēji varētu izvēlēties atbilstošo.

Krāsas veidošana:

- **Krāsas nosaukuma ierakstīšana:** Lai krāsa būtu viegli atpazīstama un pircējiem saprotama, administrators ievada nosaukumu, kas atbilst šai krāsai.
- **Krāsas koda ierakstīšana:** Lai nodrošinātu precīzu un vienotu krāsu attēlojumu, administrators ievada krāsas kodu. Šis kods parasti tiek izmantots HTML vai CSS formātā, piemēram, hex formātā, lai precīzi noteiktu krāsas toni.

Izmēru veidošana:

- **Izmēra nosaukuma ierakstīšana:** Lai izmērs būtu viegli saprotams un atpazīstams, administrators ievada nosaukumu, kas atbilst šim izmēram.
- **Izmēra saisinājuma ierakstīšana:** Lai nodrošinātu ērtu un vienkāršu klientu izvēli, administrators ievada saīsinājumu vai kodu, kas identificē attiecīgo izmēru. Šis saīsinājums var būt burtu vai ciparu kombinācija, kas ērti identificē izmēru, piemēram, "S", "M", "L", "XL.

Pielāgota dizaina veidošana:

- **Ierakstam virsrakstu:** Lai nodrošinātu skaidru un saprotamu informāciju par mājsalpau, administrators ievada virsrakstu vai nosaukumu, kas atspoguļo dizaina galveno tematu vai mērķi.
- **Ierakstam aprakstu:** Lai sniegtu detalizētu informāciju par mājaslapu, administrators ievada aprakstu, kas apraksta tā galvenās iezīmes, funkcijas un priekšrocības. Šis apraksts palīdz klientiem labāk saprast, kāda ir mājslapa un kā tas varētu atbilst viņu vajadzībām.
- **Ierakstam papildpiezīmes:** Pēdējais solis ir papildpiezīmju ierakstīšana, kas var ietvert jebkādas citu informāciju par mājaslpau, piemēram, īpašas prasības vai norādes, tehniskās specifikācijas vai jebkādas citas noderīgas norādes, kas saistītas ar dizaina izstrādi un pielāgošanu.

Dizaina izveide:

- **Izvēleties krāsu:** Lietotāji izvēlas krāsu vai krāsu paleti, kas tiks izmantota dizainā. Šī ir svarīga izvēle, kas var ietekmēt vispārējo vizuālo izskatu un saziņas efektivitāti.
- **Izvēleties izmēru (ja ir):** Ja dizains ietver objektus vai elementus ar konkrētu izmēru, lietotāji var izvēlēties vēlamo izmēru vai proporcijas, lai panāktu vēlamo efektu.

- **Pievienot bildes:** Lietotāji var pievienot attēlus vai grafiskus elementus savam dizainam, lai padarītu to vizuāli pievilcīgāku vai informatīvāku. Tas var ietvert gan pašu veidoto attēlu augšupielādi, gan arī izvēli no iepriekš sagatavotiem attēliem.
- **Pievienot tekstu:** Beidzot, lietotāji var pievienot tekstu savam dizainam, iekļaujot virsrakstus, aprakstus, saukļus vai citu tekstu, kas papildina vizuālo saturu un sniedz vajadzīgo informāciju skatītājiem vai lietotājiem.

Kontaktu veidošana:

- **Adreses ievadīšana:** Administrators ievada vai atjauno klienta vai organizācijas adresi, norādot ielas nosaukumu, mājas numuru, dzīvokļa numuru (ja nepieciešams), pilsētu un valsti.
- **Pasta indeksa ievadīšana:** Lai nodrošinātu precīzu adresi un piegādi, administrators ievada pasta indeksu, kas palīdz identificēt konkrēto adresi vai apgabalu.
- **Telefona numura ierakstīšana:** Lai nodrošinātu saziņu ar klientu vai organizāciju, administrators ievada vai atjauno telefona numuru, kas tiek izmantots saziņai vai konsultācijām.
- **E-pasta ierakstīšana:** Administrators ievada vai atjauno e-pasta adresi, kas ir svarīga saziņas veids ar klientu vai organizāciju, kā arī var tikt izmantota informācijas un reklāmas nolūkos

Dāvanu koda veidošana:

- **Koda ierakstīšana:** Administrators ievada un izveido unikālu dāvanu kodu, kas sastāv no alfabēta burtiem, cipariem vai citiem simboliem, kas identificē konkrēto atlaidi vai piedāvājumu.
- **Atzīmē, vai kods būs aktīvs:** Administrators izvēlas, vai dāvanu kods būs aktīvs vai neaktīvs. Aktīvs kods var tikt izmantots klientiem, kamēr neaktīvs kods nav derīgs un nevar tikt izmantots.
- **Ievada koda sākuma datumu:** Lai noteiktu dāvanu koda derīguma termiņu, administrators ievada koda sākuma datumu, kad kods kļūst derīgs un var tikt izmantots.
- **Ievada koda beigu datumu:** Administrators ievada koda beigu datumu, kad kods vairs nav derīgs un nevar tikt izmantots.
- **Ievada koda daudzumu:** Administrators norāda, cik daudz reižu konkrētais dāvanu kods var tikt izmantots. Piemēram, vienu kodu var izmantot tikai vienu reizi, vai arī tas var būt pieejams noteiktam skaitam lietojumu.

Sazināties ar administrātoru:

- **Ievadīt vārdu:** Lietotāji ievada savu vārdu atbilstošā lauciņā vai formā, lai administratoram būtu informācija par sūtītāja identitāti vai vārdu.
- **Ievadīt uzvārdu:** Tāpat kā vārdu, lietotāja atbilstošā ievada savu uzvārdu lauciņā vai formā, lai administratoram būtu pilnīgāka informācija par sūtītāja identitāti.
- **Ievadīt e-pastu:** Lietotāji ievada savu e-pasta adresi atbilstošā lauciņā vai formā, lai administratoram būtu iespēja atbildēt uz saņemto vēstuli vai pieprasījumu.
- **Ievadīt vēstuli:** Lietotājs ievada vēstuli vai ziņojumu, kurā iekļauti jautājumi, komentāri, pārdomas vai citu informāciju, kas nepieciešama administratoram vai uzņēmuma pārstāvim.

Grozu funkcionalitāte:

- **Pievienot preci grozā:** Šī funkcija ļauj lietotājiem pievienot produktus savam grozam, parasti, noklikšķinot uz pogas "Pievienot grozam" vai kaut kā citā veidā norādot, ka viņi vēlas iegādāties konkrēto produktu.
- **Noņemt preci no groza:** Šī funkcija ļauj lietotājiem noņemt produktus no savas pirkumu groza, ja viņi izlemj mainīt savu pasūtījumu vai vairs negrib iegādāties konkrēto preci. To parasti var darīt, noklikšķinot uz pogas "X".
- **Samazināt preču daudzumu:** Šī funkcija ļauj lietotājiem samazināt preču daudzumu grozā, ja viņi vēlas iegādāties mazāku skaitu no konkrētā produkta. To parasti var darīt, noklikšķinot uz pogas ar simbolu "-" vai citām funkciju vadības iespējām, kas norāda, ka vēlamais produkta daudzums ir mazāks.
- **Palielināt preču daudzumu:** Šī funkcija ļauj lietotājiem palielināt preču daudzumu grozā, ja viņi vēlas iegādāties vairāk no konkrētā produkta. To parasti var darīt, noklikšķinot uz pogas ar simbolu "+" vai citām funkciju vadības iespējām, kas norāda, ka vēlamais produkta daudzums ir lielāks.

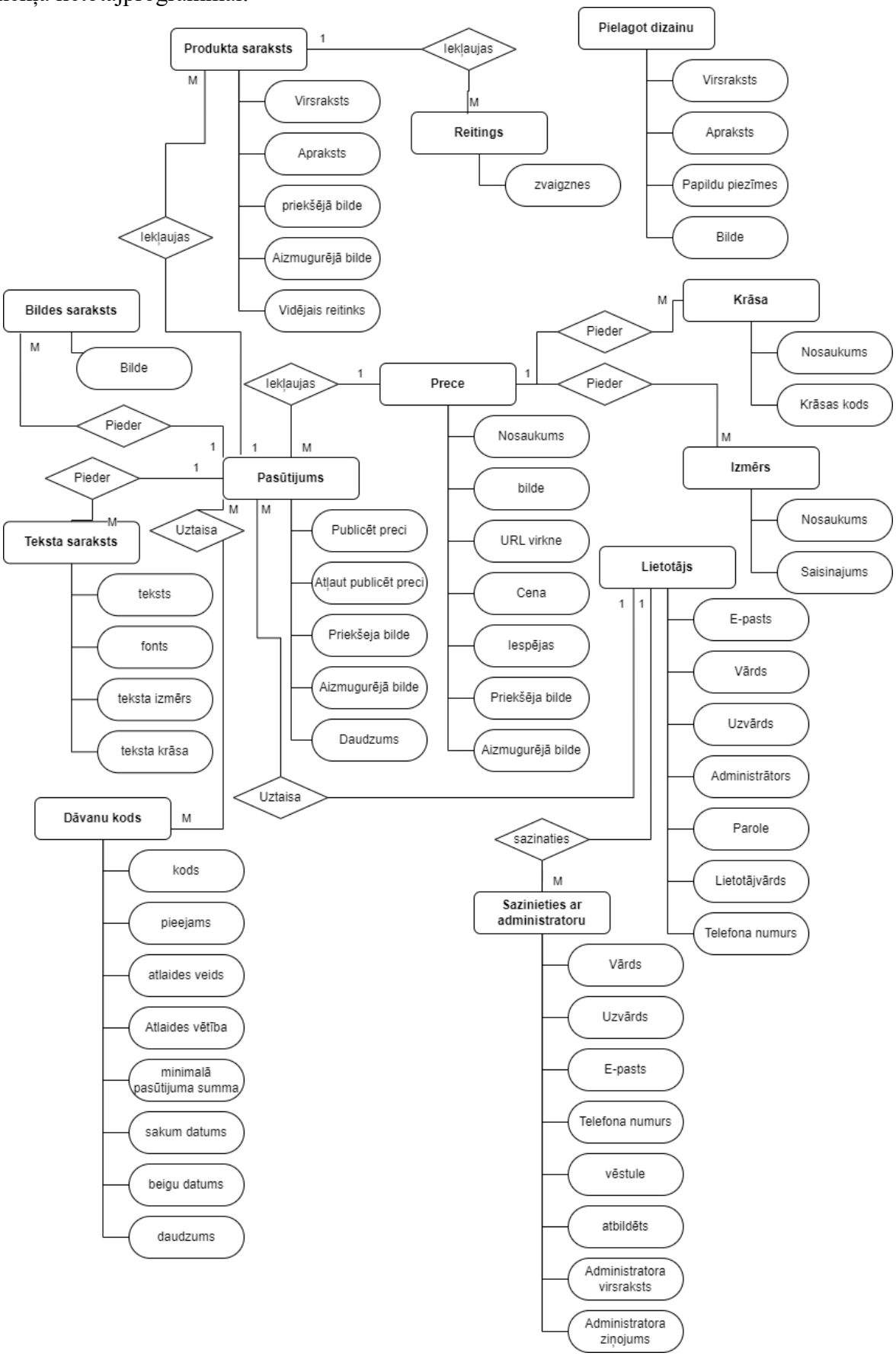
4.1.2. Sistēmas ER modelis

6. Attēlā ir attēlota sistēmas ER-modeļa shēma. ER-modelis ir datu modelēšanas valodas veids, kas izmanto entītijas, īpašības un attiecības, lai attēlotu datu struktūru.

Attēlā ir redzamas divpadsmit entītijas:

- **Produkta saraksts:** Produkta saraksts ir kopums ar dažādiem produktiem, kas pieejami mūsu tīmekļa vietnē. Katrs produkts ir aprakstīts ar nosaukumu, aprakstu, cenu un citiem atribūtiem.
- **Pielāgots dizains:** Pielāgots dizains ir sistēmas sastāvdaļa, kas ļauj administratoriem veidot un konfigurēt tīmekļa vietnes izskatu un veidlapas pēc savas izvēles.
- **Reitings:** Reitings ir novērtējums vai vērtējums, ko lietotāji var dot produktiem vai citiem elementiem, lai norādītu to kvalitāti vai atbilstību.
- **Bildes saraksts:** Bildes saraksts ir datu struktūra, kurā tiek glabātas attēlu datnes, kas saistītas ar produktiem vai citiem sistēmas elementiem.
- **Tekstu saraksts:** Teksta saraksts ir datu struktūra, kurā tiek glabāti teksta dati, piemēram, apraksti, nosaukumi vai citi teksta informācijas elementi.
- **Pasūtijums:** Pasūtījums ir datu struktūra, kas satur informāciju par lietotāja veiktajiem pasūtījumiem, to saturu un statusu.
- **Prece:** Prece ir konkrēts produkts, kas pieejams mūsu tīmekļa vietnē, un to raksturo nosaukums, apraksts, cena un citi atribūti.
- **Krāsa:** Krāsa ir datu struktūra, kas satur informāciju par pieejamajām krāsām produktiem vai citiem elementiem.
- **Izmērs:** Izmērs ir datu struktūra, kas satur informāciju par pieejamajiem izmēriem produktiem vai citiem elementiem.
- **Lietotājs:** Lietotājs ir sistēmas lietotājs, kurš ir reģistrējies un izmanto sistēmu, veicot pasūtījumus vai pārvaldot savu informāciju.
- **Sazināties ar administrātoru:** "Sazinieties ar administratoru" ir iespēja, kas ļauj lietotājiem sazināties ar sistēmas administratoru, iesniedzot ziņojumus vai jautājumus.
- **Dāvanu kods:** Dāvanu kods ir unikāls kods, ko var izmantot, lai saņemtu atlaidi vai īpašus piedāvājumus, veicot pasūtījumus mūsu tīmekļa vietnē.

Šis ER-modelis ir vienkāršs modelis, kas attēlo pamata informāciju, kas nepieciešama tīmekļa lietotājprogrammai.

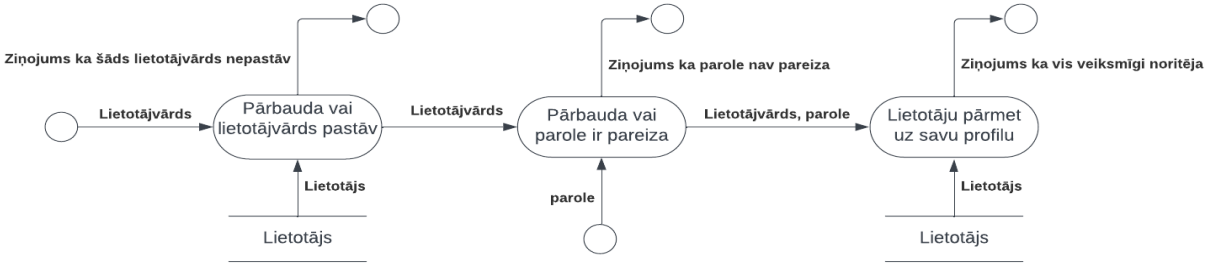


6.att. Sistēmas ER-diagramma

4.2. Funkcionālais sistēmu modelis

4.2.1. Datu plūsmas modelis

1. **Pieslēgšanās datu plūsmas modelis** (skat. 7.1. att.).



7.1.att. Datu plūsmas modelis

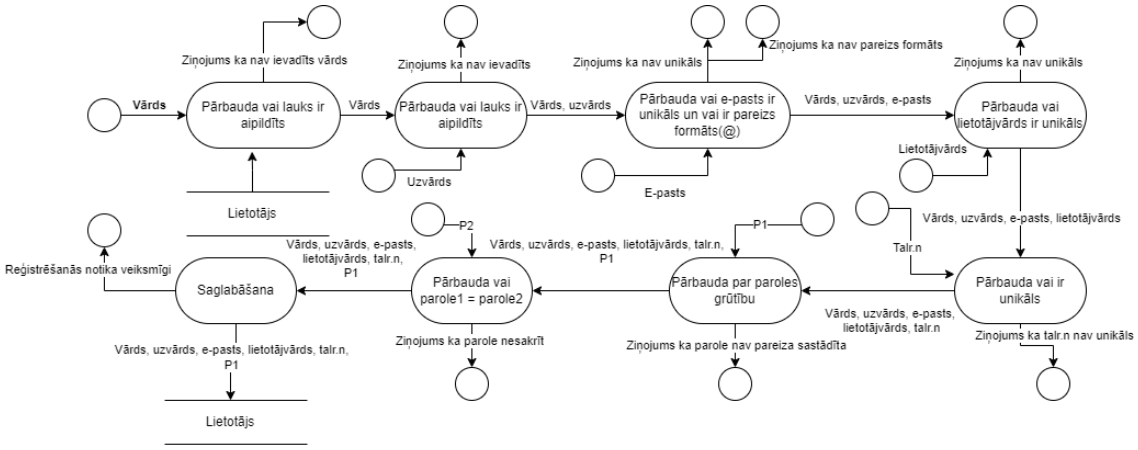
1.1. **Lietotājvārda pārbaude:** Lietotājs ievada lietotājvārdu. Datu plūsmas modelis pārbauda, vai šāds lietotājvārds eksistē sistēmā. Ja lietotājvārds ir derīgs, tiek paziņots par to.

1.2. **Paroles pārbaude:** Lietotājs ievada paroli. Datu plūsmas modelis salīdzina ievadīto paroli ar glabāto paroli attiecīgajam lietotājvārdam. Ja paroles nesakrīt, tiek paziņots par to.

1.3. **Pieslēgšanās pie lietotāja profila:** Ja lietotājvārds un parole ir derīgi un sakrīt, lietotājs tiek pārvests uz savu profilu vai darba saskarni, un tiek paziņots par veiksmīgu pieslēgšanos.

Šis datu plūsmas modelis nodrošina drošu un pārbaudītu pieslēgšanās procesu, kas ietver identifikācijas un autentifikācijas posmus, lai nodrošinātu pareizu piekļuvi lietotāja profilam vai sistēmai.

2. **Reģistrēšanās datu plūsmas modelis** (skat. 7.2. att.).



7.2.att. Datu plūsmas modelis

2.1. **Vārda pārbaude**: Lietotājs ievada savu vārdu, un sistēma pārbauda, vai lauks ir aizpildīts. Ja vārds nav ievadīts, tiek parādīts ziņojums par nepieciešamību aizpildīt šo lauku.

2.2. **Uzvārda pārbaude**: Lietotājs ievada savu uzvārdu, sistēma pārbauda, vai lauks ir aizpildīts. Ja uzvārda nav, parādīts ziņojums par nepieciešamību aizpildīt šo lauku.

2.3. **E-pasta pārbaude:** Lietotājs ievada savu e-pastu, un sistēma pārbauda, vai e-pasts ir unikāls un vai tas atbilst pareizam formātam (piemēram, satur "@"). Ja e-pasts nav unikāls vai neatbilst pareizajam formātam, tiek parādīti atbilstoši ziņojumi.

2.4. **Lietotājvārda pārbaude:** Lietotājs ievada savu lietotājvārdu, un sistēma pārbauda, vai tas ir unikāls. Ja lietotājvārds nav unikāls, tiek parādīts ziņojums par nepieciešamību izvēlēties citu lietotājvārdu.

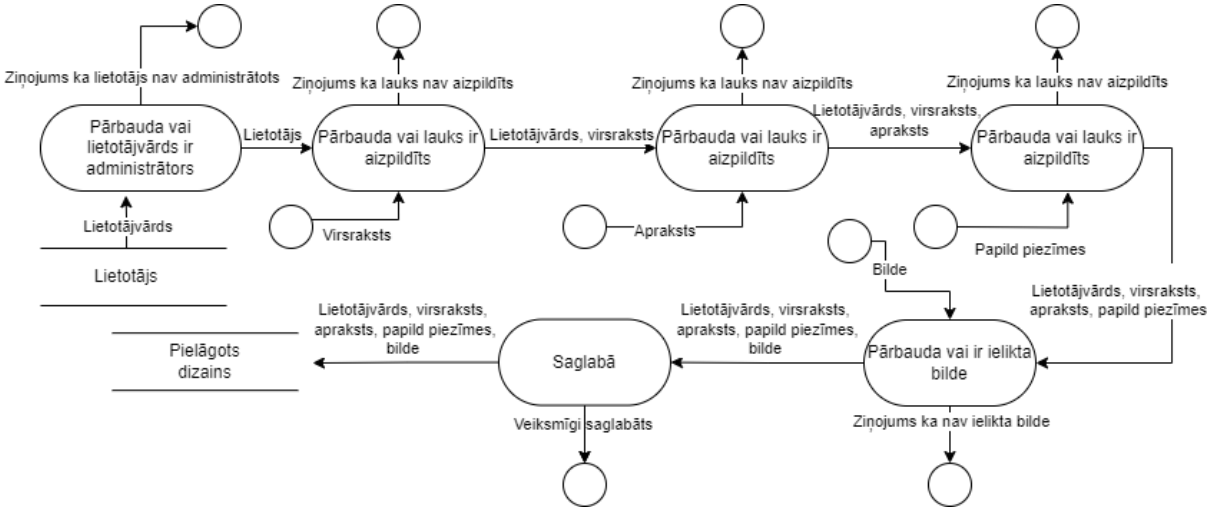
2.5. **Talruņa numura pārbaude**: Lietotājs ievada savu tālruņa numuru, un sistēma pārbauda, vai tas ir unikāls. Ja tālruņa numurs nav unikāls, tiek parādīts ziņojums par nepieciešamību izvēlēties citu tālruņa numuru.

2.6. **Paroles pārbaude**: Lietotājs ievada pirmo paroli, un sistēma pārbauda, vai parole atbilst noteikumiem par drošību (piemēram, satur lielu burtu, mazo burtu, ciparu un simbolu). Ja parole neatbilst šiem noteikumiem, tiek parādīts ziņojums par nepieciešamību izvēlēties drošāku paroli.

2.7. **Paroles salīdzināšana**: Lietotājs ievada otro paroli, un sistēma pārbauda, vai otra parole atbilst pirmajai parolei. Ja paroles nesakrīt, tiek parādīts ziņojums par nepieciešamību ievadīt pareizu otro paroli.

2.8. **Informācijas saglabāšana:** Ja visi iepriekšminētie soļi tiek veiksmīgi pabeigti, tiek parādīts ziņojums, ka reģistrēšanās notikusi veiksmīgi, un lietotāja dati tiek saglabāti.

3. **Pielāgota dizaina datu plūsmas modelis** (skat. 7.3. att.).



7.3.att. Datu plūsmas modelis

3.1. **Administratora autentifikācija:** Sistēma pārbauda, vai lietotājvārds ir "administrātors". Ja lietotājvārds nav "administrātors", tiek parādīts atbilstošs ziņojums.

3.2. **Virsraksta pārbaude**: Lietotājs ievada virsrakstu pārbauda, vai lauks ir aizpildīts. Ja virsraksts nav ievadīts, tiek ziņojums ka nepieciešamību aizpildīt šo lauku.

3.3. **Apraksta pārbaude**: Lietotājs ievada aprakstu, un sistēma pārbauda, vai lauks ir aizpildīts. Ja apraksts nav ievadīts, tiek parādīts ziņojums par nepieciešamību aizpildīt šo lauku.

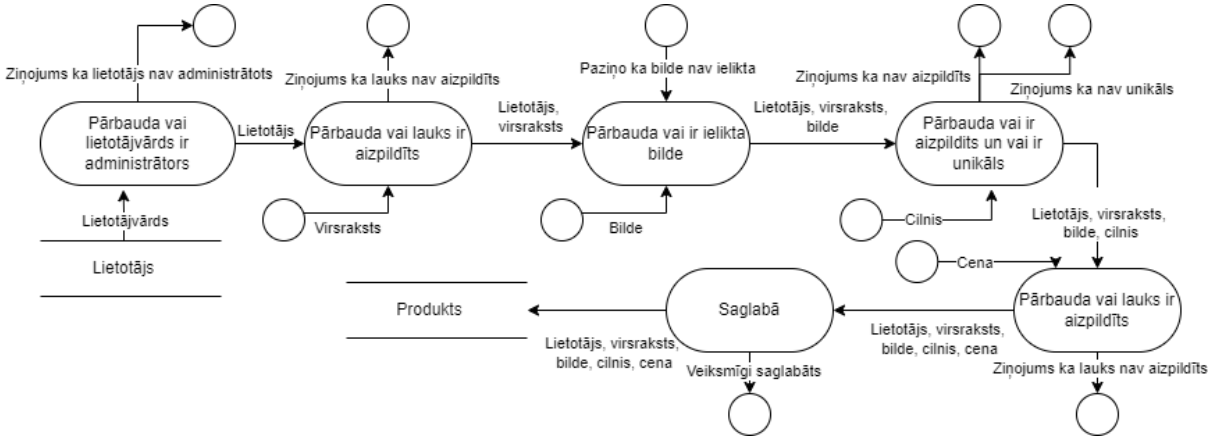
3.4. **Papild piezīmes pārbaude**: Lietotājs pievieno papildu piezīmes, un sistēma pārbauda, vai lauks ir aizpildīts. Ja piezīmes nav ievadītas, tiek parādīts ziņojums par nepieciešamību aizpildīt šo lauku.

3.5. **Attēlu pārbaude**: Lietotājs pievieno attēlu, un sistēma pārbauda, vai ir ielikta bilde. Ja bilde nav pievienota, tiek parādīts ziņojums par nepieciešamību pievienot bildi.

3.6. **Saglabāšana datu krātuvē**: Kad visi iepriekšminētie soļi ir veiksmīgi pabeigti, sistēma saglabā pielāgoto dizainu datubāzē un ziņo par veiksmīgu saglabāšanu.

Pielāgota dizaina datubāze nodrošina, ka administrātors var saglabāt un atjaunot savus izvēlētos vizuālos uzstādījumus un pielāgotus dizainus sistēmā. Tas ļauj administrātoriem saglabāt konsistenci un personalizēt sistēmas izskatu.

4. **Produkta datu plūsmas modelis** (skat. 7.4. att.).



7.4.att. Datu plūsmas modelis

4.1. **Administratora autentifikācija:**Sistēma pārbauda, vai lietotājvārds ir "administrātors". Ja lietotājvārds nav "administrātors", tiek parādīts atbilstošs ziņojums.

4.2. **Virsraksta pārbaude**: Lietotājs ievada virsrakstu, un sistēma pārbauda, vai lauks ir aizpildīts. Ja virsraksts nav ievadīts, tiek parādīts ziņojums par nepieciešamību aizpildīt šo lauku.

4.3. **Attēlu pārbaude**: Lietotājs pievieno bildi, un sistēma pārbauda, vai ir ielikta bilde. Ja bilde nav pievienota, tiek parādīts ziņojums, ka bilde nav ielikta.

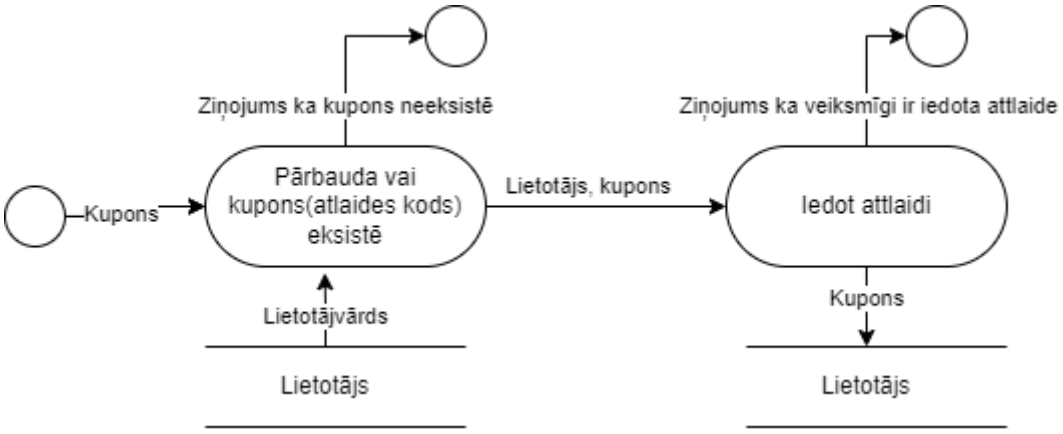
4.4. **Cilnes pārbaude:** Lietotājs ievada cilni, un sistēma pārbauda, vai cilnis ir aizpildīts un vai tas ir unikāls. Ja cilnis nav aizpildīts vai nav unikāls, tiek parādīts atbilstošs ziņojums.

4.5. **Cenu pārbaude:** Lietotājs ievada cenu, un sistēma pārbauda, vai lauks ir aizpildīts. Ja cena nav ievadīta, tiek parādīts ziņojums par nepieciešamību aizpildīt šo lauku.

4.6. **Saglabāšana datu krātuvē**: Kad visi iepriekšminētie soļi ir veiksmīgi pabeigti, sistēma saglabā produktu datubāzē un ziņo par veiksmīgu saglabāšanu.

Šī datu plūsmas diagramma nodrošina efektīvu un uzticamu produktu informācijas pārvaldību, kas ietver datu ievadīšanu, validāciju un saglabāšanu datubāzē, nodrošinot, ka visi nepieciešamie dati ir precīzi un droši saglabāti sistēmā.

5. **Kupona datu plūsmas modelis** (skat. 7.5. att.).



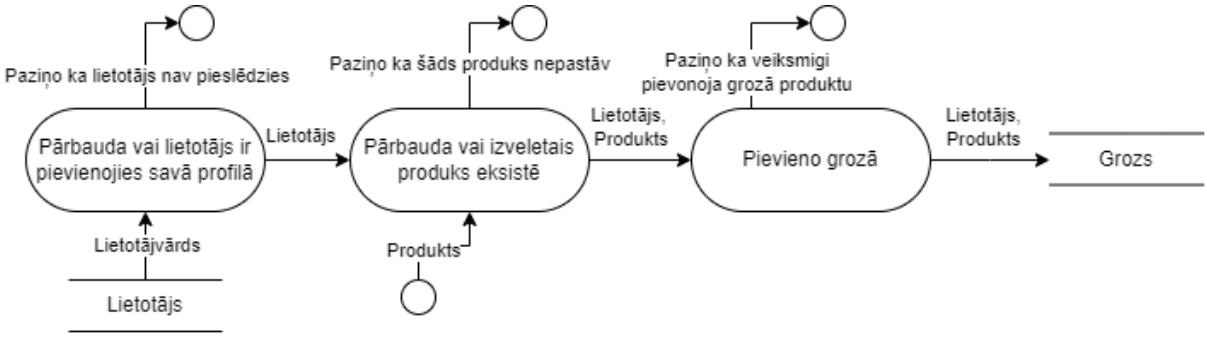
7.5.att. Datu plūsmas modelis

5.1. **Kupona pārbaude:** Lietotājs ievada kupona kodu, un sistēma pārbauda, vai šāds kupons (atlaides kods) eksistē. Ja kupons neeksistē, tiek izvadīts paziņojums, ka šāds kupons neeksistē.

5.2. **Attlaides piešķiršana:** Ja kods ir derīgs, sistēma piešķir atlaidi un ziņo par veiksmīgu atlaides piešķiršanu.

Šis proces nodrošina lietotāju ar iespēju izmantot kuponus, pārbaudot tos pret esošajiem ierakstiem sistēmā. Tas garantē, ka tikai derīgi kuponi tiek pieņemti, samazinot iespējamo kļūdu un nodrošinot labāku lietotāja pieredzi.

6. **Groza pievienošanas datu plūsmas modelis** (skat. 7.6. att.).



7.6.att. Datu plūsmas modelis

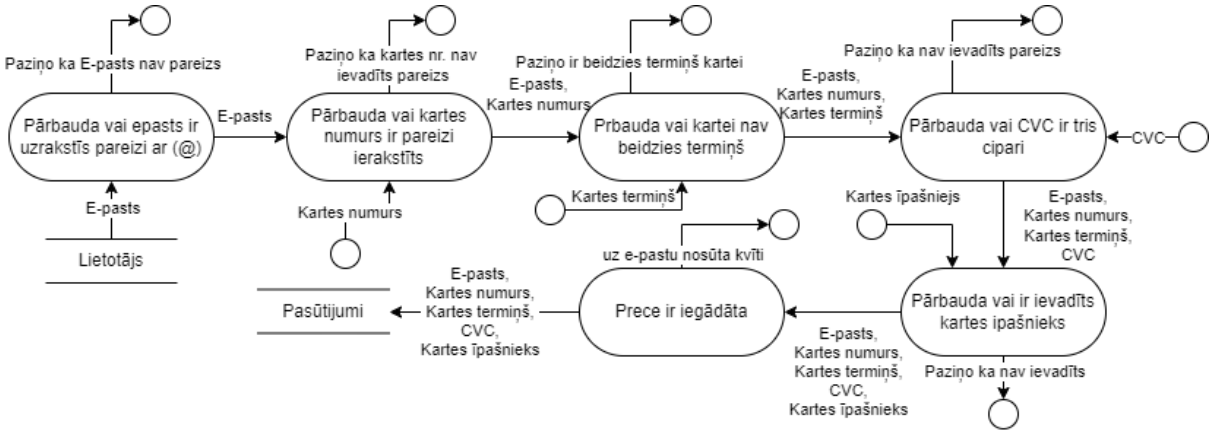
6.1. **Autorizācijas pārbaude:** Sākotnēji sistēma pārbauda, vai lietotājs ir autorizējies savā profilā. Ja lietotājs nav autorizējies vai sesija ir beigusies, tiek paziņots, ka lietotājs nav pieslēdzies.

6.2. **Produkta pievienošana grozam:** Pēc veiksmīgas autorizācijas lietotājs izvēlas produktu un mēģina to pievienot grozam. Sistēma pārbauda, vai izvēlētais produkts eksistē datu bāzē. Ja produkts ir pieejams, tas tiek veiksmīgi pievienots grozam.

6.3. **Paziņojums par veiksmīgu darbību:** Kad produkts ir veiksmīgi pievienots grozam, lietotājs saņem paziņojumu par veiksmīgu darbību. Tas informē lietotāju, ka produkts ir veiksmīgi pievienots grozam un ir gatavs turpmākai pirkšanai.

Šis process nodrošina lietotājiem vieglu un saprotamu veidu, kā pārbaudīt savu profilu un pievienot produktus grozam tieši no sava konta. Tas nodrošina arī drošību un uzticamību, pārbaudot gan lietotāja autentifikāciju, gan produkta pieejamību, pirms tas tiek pievienots grozam. Tā rezultātā lietotāji var sajusties komfortabli un paļauties uz sistēmu, kad veic pirkumu darbības.

7. **Pasūtiju noformēšana datu plūsmas modelis** (skat. 7.7. att.).



7.7.att. Datu plūsmas modelis

7.1. **E-pasta pārbaude:** Lietotājs ievada savu e-pastu, un sistēma pārbauda, vai tas satur "@" simbolu, kas norāda uz pareizu e-pasta formātu. Ja e-pasta adrese netiek pareizi ievadīta (nav "@" simbola), tiek paziņots, ka e-pasts nav pareizs.

7.2. **Kartes numura pārbaude:** Lietotājs ievada kartes numuru, un sistēma pārbauda, vai numurs ir pareizi ierakstīts. Ja ievadītais kartes numurs neatbilst pareizam formātam, tiek paziņots, ka kartes numurs nav ievadīts pareizi.

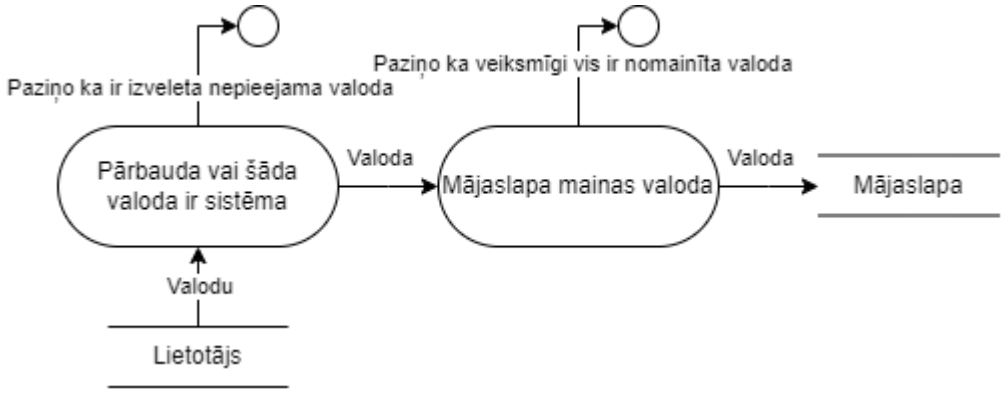
7.3. **Kartes termiņa pārbaude:** Lietotājs ievada kartes termiņu, un sistēma pārbauda, vai tas vēl nav beidzies. Ja kartes termiņš ir pagājis, tiek paziņots, ka kartei ir beidzies termiņš.

7.4. **CVC pārbaude:** Lietotājs ievada CVC (kartes verifikācijas kodu), un sistēma pārbauda, vai tas sastāv no trīs ciparіem. Ja CVC neatbilst formātam, tiek paziņots, ka tas nav ievadīts pareizi.

7.5. **Kartes īpašnieka pārbaude:** Lietotājs ievada kartes īpašnieka vārdu, uzvārdu vai uzraksta nosaukumu, un sistēma pārbauda, vai šis lauks ir ievadīts. Ja lauks ir tukšs, tiek paziņots, ka tas nav ievadīts.

7.6. **Pasūtījuma apstiprināšana un apstrāde:** Pēc veiksmīgas visu informācijas ievades un pārbaudes, sistēma apstiprina pasūtījumu un saglabā to datubāzē. Turklāt, ja prece ir veiksmīgi iegādāta, sistēma nosūta lietotājam kvīti PDF formātā uz norādīto e-pastu.

8. **Valodas maiņa (lokalizācija) datu plūsmas modelis** (skat. 7.8. att.).



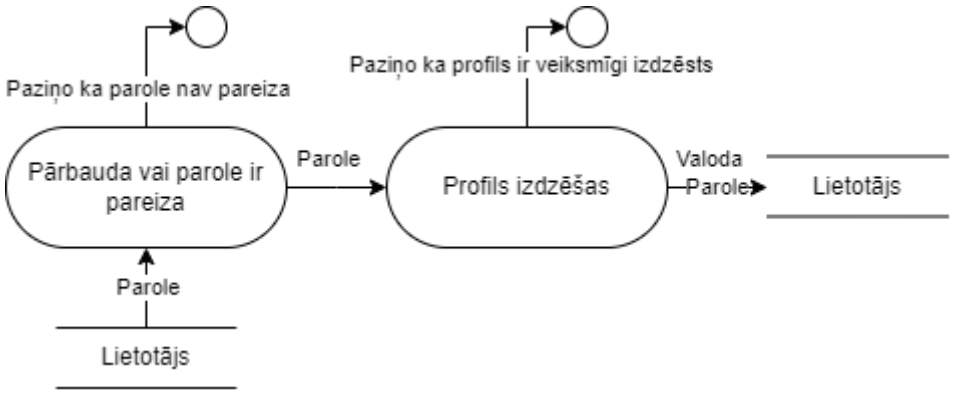
7.8.att. Datu plūsmas modelis

8.1. **Valodas izvēles pārbaude:** Lietotājs izvēlas vēlamo valodu, un sistēma pārbauda, vai šāda valoda ir pieejama sistēmā. Ja izvēlētā valoda nav pieejama, tiek paziņots, ka ir izvēlēta nepieejama valoda.

8.2. **Mājaslapas valodas maiņa:** Pēc veiksmīgas valodas izvēles, mājaslapa automātiski maina valodu uz izvēlēto valodu. Lietotājs saņem paziņojumu, ka valoda ir veiksmīgi nomainīta uz jauno valodu.

Tādējādi sistēma nodrošina iespēju lietotājam izvēlēties vēlamo valodu un veikt mājaslapas valodas maiņu, vienlaikus informējot par izvēles veiksmīgu vai neveiksmīgu rezultātu.

9. **Profila dzēšana datu plūsmas modelis** (skat. 7.9. att.).



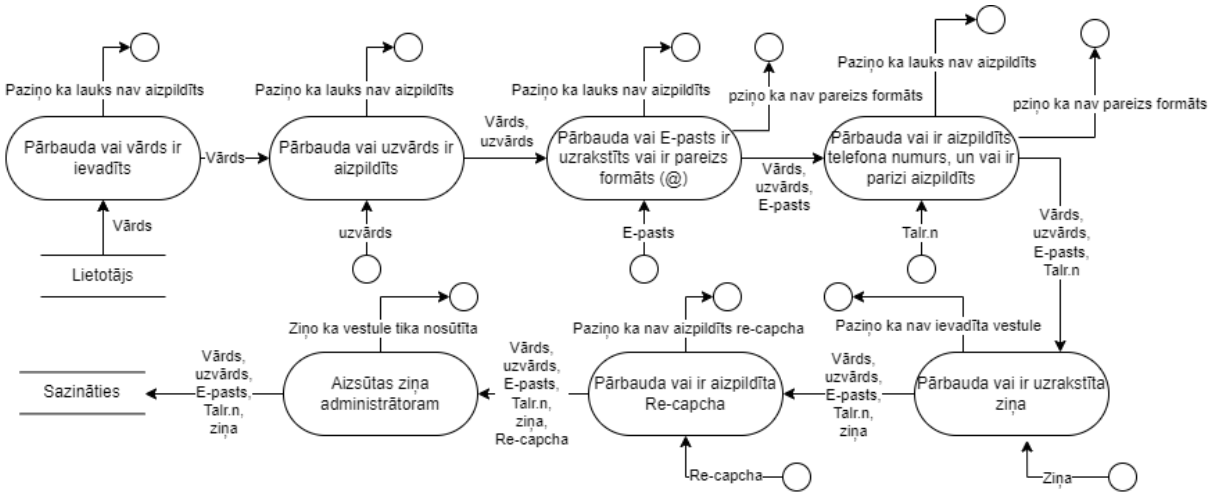
7.9.att. Datu plūsmas modelis

9.1. **Paroles pārbaude:** Lietotājs ievada paroli, un sistēma pārbauda, vai ievadītā parole ir pareiza. Ja ievadītā parole neatbilst saglabātajai parolei, tiek paziņots, ka parole nav pareiza.

9.2. **Profila dzēšana:** Pēc veiksmīgas paroles pārbaudes lietotājs pieprasa savu profilu dzēst. Sistēma izdzēš lietotāja profilu no datubāzes un visiem saistītajiem datiem. Lietotājs saņem paziņojumu, ka profils ir veiksmīgi izdzēsts.

Šī datu plūsmas diagramma tādējādi palīdz uzlabot sistēmas drošību, nodrošinot, ka tikai autorizēti lietotāji var piekļūt sistēmai un veikt darbības, kā arī piedāvājot lietotājiem kontroli pār saviem datiem un kontiem

10. **Sazināties ar administratoriem datu plūsmas modelis** (skat. 7.10. att.).



7.10.att. Datu plūsmas modelis

10.1. **Vārda pārbaude:** Pirms turpināt, sistēma pārbauda, vai lietotājs ir ievadījis vārdu. Ja lauks ir tukšs, tiek paziņots, ka vārds nav aizpildīts.

10.2. **Uzvārda pārbaude:** Pēc vārda ievades sistēma pārbauda, vai lietotājs ir ievadījis uzvārdu. Ja uzvārda lauks ir tukšs, tiek paziņots, ka uzvārds nav aizpildīts.

10.3. **E-pasta pārbaude:** Kad vārds un uzvārds ir ievadīts, sistēma pārbauda, vai e-pasta adrese ir pareizi ievadīta un satur "@" simbolu. Ja e-pasta adrese nav pareizi ievadīta, paziņo ka lauks nav aizpildīts vai ka tā formāts nav pareizs.

10.4. **Telefona numura pārbaude:** Pēc e-pasta ievades sistēma pārbauda, vai lietotājs ir ievadījis telefona numuru un vai tas ir pareizi formāts. Ja telefona numurs nav ievadīts vai tā formāts nav pareizs, tiek paziņots, ka lauks nav aizpildīts vai ka tā formāts nav pareizs.

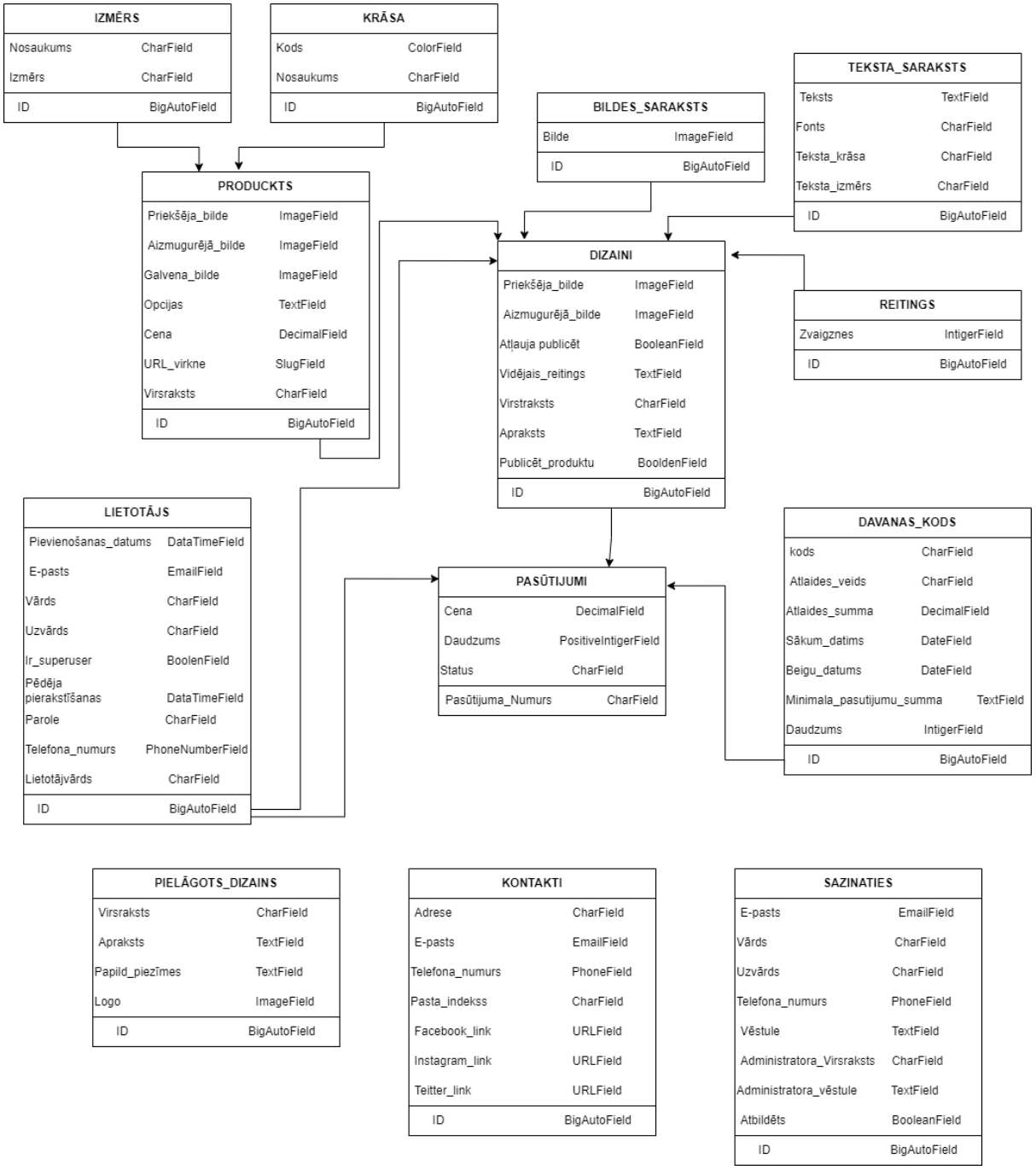
10.5. **Ziņas pārbaude:** Kad ir ievadīti personas dati, sistēma pārbauda, vai lietotājs ir ievadījis ziņu. Ja ziņa nav ievadīta, tiek paziņots, ka vestule nav ievadīta.

10.6. **Re-Captcha pārbaude:** Lietotājam tiek prasīts aizpildīt Re-Captcha lauku, lai pārliecinātos, ka sistēmā nav roboti. Ja Re-Captcha lauks nav aizpildīts, tiek paziņots, ka tas nav aizpildīts.

10.7. sistēma nosūta ziņu administrātoram un saglabā to datubāzē, lai administrācija varētu sazināties ar lietotāju.

5. DATU STRUKTŪRAS APRAKSTS

Datubāze tiks izveidota ar 13 tabulām, kurās tiks glabāta informācija par lietotājiem, produktiem, pasutijumiem, dizainiem , reitingu, davanas kodiem, utt.. Zemāk redzama datubāzes tabulu relāciju shēma (skat. 8. att.).



8.att. Datubāzes tabulu relāciju shēma

Tabula “Lietotāji” glabā datus par lietotājiem, kuri ir autentificējušies sistēmā.

1.Tabula

Tabulas “LIETOTAJI” struktūra

Nr.	Lauka saturs	Lauka nosaukums	Tips, garums	Piezīmes
1.	Lietotāja E-pasts	E-pasts	VarChar(250)	Obligāts, unikāls
2.	Lietotāja vārds	Vārds	VarChar(150)	Obligāts
3.	Lietotāja uzvārds	Uzvārds	VarChar(150)	Obligāts
4.	Vai lietotājs ir administrātors	Administrātors	Boolen	
5.	Lietotāja profila parole	Parole	VarChar(150)	Obligāts
6.	Lietotāja lietotājvārds	Lietotājvārds	VarChar(128)	Obligāts, unikāls
7.	Telefona numurs	Talr_n	VarChar(128)	Obligāts, unikāls

Tabula “Dizaini” glabā datus par dizainiem, kuri ir izveidojuši lietotāji.

2.Tabula

Tabulas “Dizaini” struktūra

Nr.	Lauka saturs	Lauka nosaukums	Tips, garums	Piezīmes
1.	Produkta pirkšēja bilde ar lietotāja dizainu	Priekšēja_bilde	Image	
2.	Produkta aizmugurējā bilde ar lietotāja dizainu	Aizmugurējā_bilde	Image	
3.	Administrators atļauj publicēt darbu	Atļauja publicēt	Boolen	
4.	Dizaina vidējais reitings	Vidējais_reitings	INT	
5.	Dizaina virsraksts	Virstraksts	VarChar(255)	
6.	Dizaina Apraksts	Apraksts	Text	
7.	Vai vēlas publicēt darbu	Publicēt_produktu	Boolen	

Tabula “Produkts” glabā datus par esošajiem produktiem, kuru izveido administrators.

3.Tabula

Tabulas “Produkts” struktūra

Nr.	Lauka saturs	Lauka nosaukums	Tips, garums	Piezīmes
1.	Priekšēka bilde lai varetu dizainot	Priekšēja_bilde	Image	Obligāti
2.	Aizmugurējā bilde lai varetu dizainot	Aizmugurējā_bilde	Image	Obligāti
3.	Galvenā bilde kur rādisies	Galvena_bilde	Image	Obligāti
4.	Priekšrocības	Opcijas	Text	
5.	Produkta cena	Cena	Decimal	Obligāti
6.	Dizaina lapas url	URL_virkne	VarChar(50)	Unikāls, Obligāti
7.	Preces virsraksts	Virsraksts	VarChar(100)	Unikāls, Obligāti

Tabula “Davanas kods” glabā datus par esošajiem davana kodu, ar kuriem lietotāji varēs dabūt atlaides visādas.

4.Tabula

Tabulas “Dāvanas kods” struktūra

Nr.	Lauka saturs	Lauka nosaukums	Tips, garums	Piezīmes
1.	Dāvanas kods	kods	VarChar(50)	Unikāls, Obligāti
2.	Kads bus atlaides veids	Atlaides_veids	VarChar(20)	
3.	Summa vai %	Atlaides_summa	VarChar(20)	
4.	Kad sakas kupons	Sākum_datums	Date	
5.	Kad beidzas kupons	Beigu_datums	Date	
6.	Pasutijumu minimāla vērtība	Minimala_pasutijumu_summa	Decimal	
7.	Kuponu daudzums	Daudzums	INT	

Tabula “Sazināties” glabā datus par uzrakstītiām vēstulem no klientu puses.

5.Tabula

Tabulas “Sazināties” struktūra

Nr.	Lauka saturs	Lauka nosaukums	Tips, garums	Piezīmes
1.	Lietotāja E-pasts	E-pasts	VarChar(250)	Obligāts
2.	Lietotāja vārds	Vārds	VarChar(150)	Obligāts
3.	Lietotāja uzvārds	Uzvārds	VarChar(150)	Obligāts
4.	Telefona numurs	Talr_n	VarChar(128)	Obligāts
5.	Vēstule ko klients sūtija	Vēstule	Text	Obligāts
6.	Administratora virsraksts vestulei	Administrātora_virsraksts	VarChar(255)	
7.	Administratora vestule ko sutis atpakaļ klientam	Administrātora_vēstule	Text	
8.	Atzimē ka ir atbildēts	Atbildēts	Boolen	

Tabula “Kontakti” glabā datus par uzņēmuma informāciju.

6.Tabula

Tabulas “Kontakti” struktūra

Nr.	Lauka saturs	Lauka nosaukums	Tips, garums	Piezīmes
1.	Firmas adrese	Adrese	VarChar(255)	Obligāts
2.	Firmas E-pasts	E-pasts	VarChar(250)	Obligāts
3.	Telefona numurs	Talr_n	VarChar(128)	Obligāts
4.	Firmas pasta indeks	Pasta_indekss	VarChar(20)	
5.	Firmas facebook saite	Facebook_link	VarChar(200)	
6.	Firmas Instagram saite	Instagram_link	VarChar(200)	
7.	Firmas twiter saite	Twiter_link	VarChar(200)	

Tabula “Pielāgots dizains” glabā sākumlapas izskatu.

7.Tabula

Tabulas "Pielāgots_dizains" struktūra

Nr.	Lauka saturs	Lauka nosaukums	Tips, garums	Piezīmes
1.	Sakumlapas virsraksts	Virsraksts	VarChar(100)	
2.	Aprkasts	Apraksts	Text	
3.	Papild piezīmes	Papild_piezīmes	Text	
4.	Sakumlapas Logo	Logo	Image	

Tabula "Teksta saraksts" glabā lietotāju teksta informāciju ko ir ielikuši sava dizaina.

8.Tabula

Tabulas "Teksta saraksts" struktūra

Nr.	Lauka saturs	Lauka nosaukums	Tips, garums	Piezīmes
1.	Ievietotais teksts	Teksts	Text	
2.	Izvēletais fonts	Fonts	VarChar(50)	
3.	Izvēlētā krāsa	Teksta krāsa	VarChar(20)	
4.	Izvēletais izmērs	Teksta_izmērs	VarChar(25)	

Tabula "Pasūtijumi" glabā lietotāju teksta informāciju ko ir ielikuši sava dizaina.

9.Tabula

Tabulas "Pasūtijumi" struktūra

Nr.	Lauka saturs	Lauka nosaukums	Tips, garums	Piezīmes
1.	Pasūtijuma numurs	Pasūtījumu_numurs	VarChar(100)	Unikāls, Obligāts
2.	Kopēja cenas summa	Cena	Decimal	
3.	Daudzums	Daudzums	INT	
4.	Pasūtijumu status	Status	VarChar(20)	

Tabula "Krāsa" glabā Produkta pieejamās krāsas.

10.Tabula

Tabulas "Krāsa" struktūra

Nr.	Lauka saturs	Lauka nosaukums	Tips, garums	Piezīmes
1.	Krāsas kods	Kods	VarChar(20)	Obligāts
2.	Krāsas nosaukums	Nosaukums	VarChar(100)	Obligāts

Tabula “Izmērs” glabā Produkta pieejamos izmērus.

11.Tabula

Tabulas “Izmērs” struktūra

Nr.	Lauka saturs	Lauka nosaukums	Tips, garums	Piezīmes
1.	Izmēra saisinājums	izmērs	VarChar(10)	Obligāts
2.	izmēra nosaukums	Nosaukums	VarChar(100)	Obligāts

Tabula “bildes saraksts” glabā lietotāju bildes informāciju ko ir ielikuši sava dizaina.

12.Tabula

Tabulas “Bildes saraksts” struktūra

Nr.	Lauka saturs	Lauka nosaukums	Tips, garums	Piezīmes
1.	Ievietotās bildes	Bilde	Image	

Tabula “Reitings” glabā lietotāju dizaina reitingu.

13.Tabula

Tabulas “Reitings” struktūra

Nr.	Lauka saturs	Lauka nosaukums	Tips, garums	Piezīmes
1.	Zivaigznes daudzums	Zvaigznes	INT	

6. LIETOTĀJA CEĻVEDIS

6.1.Sistēmas prasības aparatūrai un programmatūrai

- **Interneta savienojums:** Lai piekļūtu mājaslapai, lietotājiem ir jābūt pieejamam interneta savienojumam. Vislabākais savienojuma veids ir Wi-Fi vai mobilais datu tīkls.
- **Pārlūkprogramma:** Lietotājiem ir nepieciešama pārlūkprogramma, lai piekļūtu mājaslapai. Rekomendējamās pārlūkprogrammas ir Google Chrome, Mozilla Firefox, Safari un Microsoft Edge.
- **Interneta pārlūkprogrammas atbilstība:** Mājaslapa ir optimizēta darbam vispopulārākajās pārlūkprogrammās un to jaunākajās versijās. Lai nodrošinātu labu lietotāja pieredzi, ieteicams izmantot vienu no minētajām pārlūkprogrammām.
- **JavaScript atbalsts:** Mājaslapa izmanto dažādas interaktīvas funkcijas, kas balstītas uz JavaScript. Lietotājiem ir jābūt iespējai izmantot JavaScript savā pārlūkprogrammā, lai pilnībā izmantotu mājaslapas funkcionalitāti.
- **Autorizācija un autentifikācija:** Mājaslapa prasa lietotāju autorizāciju un autentifikāciju, lietotājiem ir jābūt gataviem pieņemt sīkdatnes un atbalstīt šīs funkcijas pārlūkprogrammā.

6.2.Sistēmas instalācijas un palaišana

Šī dokumentācija sniedz pamata norādes Django programmas instalēšanai un palaišanai lokālajā izstrādes vidē, izmantojot jau esošu projektu.

1. **Lejupielādējiet Python:**

Lejupielādējiet un instalējiet atbilstošo Python versiju no Python oficiālās lejupielādes lapas. (https://www.python.org/downloads/)

2. **Klonējiet Projektu:**

Atveriet termināli vai komandrindu un izpildiet šo komandu, lai noklonētu projektu:

‘‘Git clone https://github.com/BasistaPlay/Printing’’

3. **Instalējiet Nepieciešamās Bibliotēkas**

Pirms palaistat projektu, pārliecinieties, ka instalējat visas nepieciešamās bibliotēkas, kas norādītas requirements.txt failā.

Navigējiet uz projektu lokāciju un izpildiet šo komandu terminālī, lai instalētu nepieciešamās bibliotēkas:

“pip install -r requirements.txt”

4. **Konfigurējiet Datu bāzi:**

Rediģējiet settings.py failu savā Django projektā, lai norādītu datu bāzes konfigurāciju, piemēram, datu bāzes tipu, nosaukumu, lietotājvārdu un paroli.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'urban_prod',
        'USER': 'u_urban',
        'PASSWORD': '123',
        'HOST': 'localhost',
        'PORT': '',
    }
}
```

5. Izveidojiet Migrācijas:

Izveidojiet datu bāzes migrācijas, izmantojot šo komandu terminālī:

"python manage.py makemigrations"

6. Veiciet Migrācijas:

Veiciet faktisko datu bāzes migrēšanu ar šo komandu:

"python manage.py migrate"

7. Palaidiet Serveri:

Palaidiet izstrādes serveri, izpildot šo komandu:

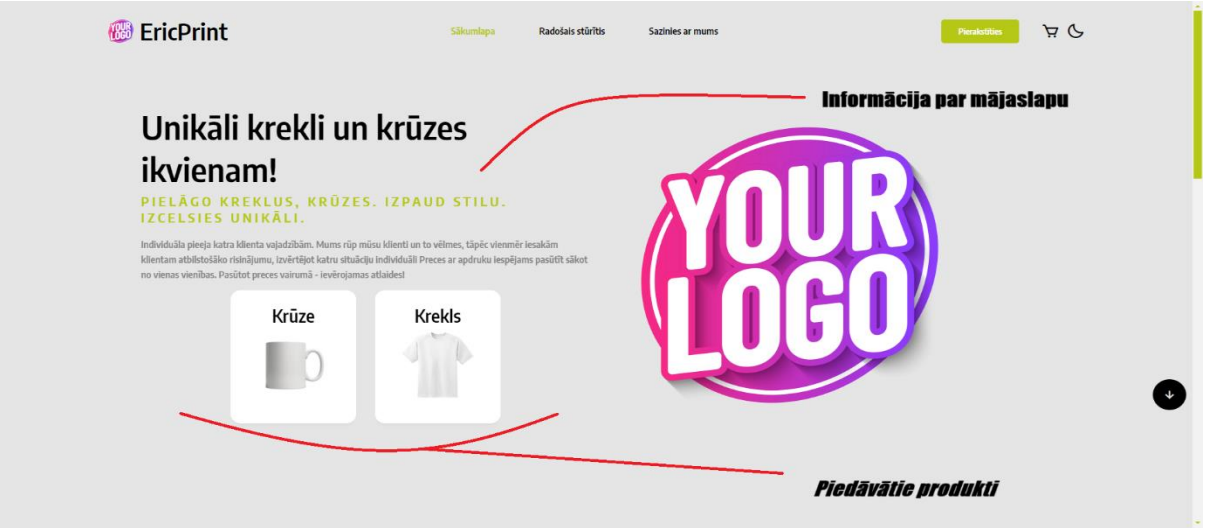
"python manage.py runserver"

Pēc šīm darbībām jūsu Django programma ir gatava lietošanai. Apmeklējiet http://localhost:8000/ savā pārlūkprogrammā, lai pārliecinātos, ka viss darbojas pareizi.

6.3.Programmas apraksts

1. Sākumplapa:

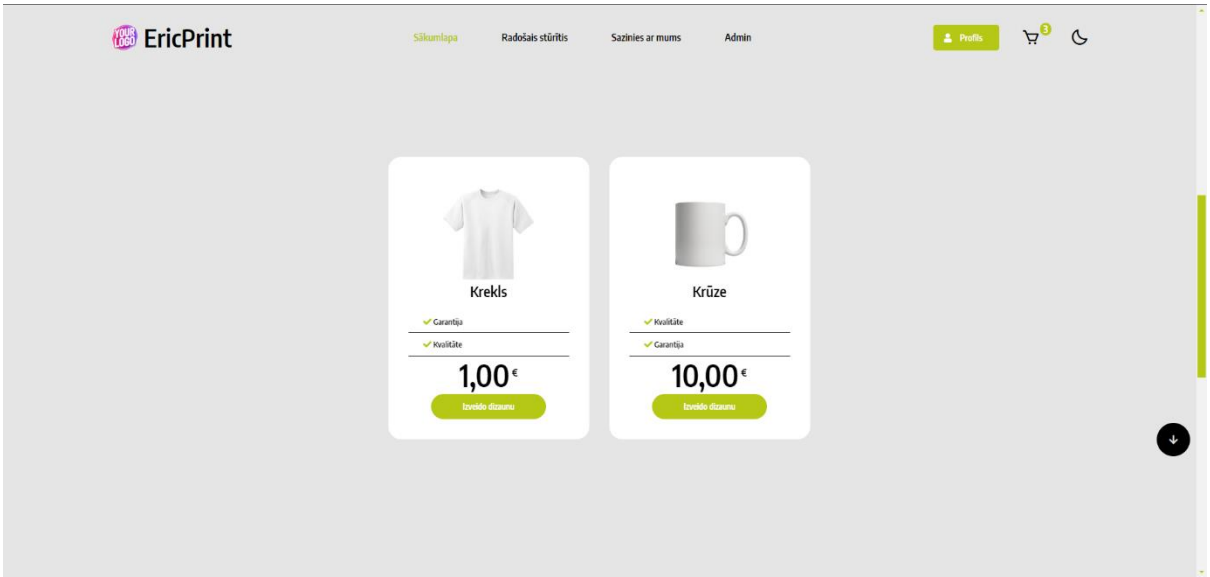
8. Attēlā var redzēt sākumlapu ar informāciju par majaslapu, cenrādi un 3 populārākos dizainus, lai tikutu sākumlapā jāieiet pārlūk lapā www.ericprint.com.



8.att. Sākumlapa

8. Attēlā redzama tīmekļa lapas sākumlapa ar nosaukumu "EricPrint", kas specializējas unikālu T-kreklu un krūzīšu dizainu izveidē. Tīmekļa lapa ir iekārtota ar šādām sadaļām un elementiem:

1. Logo: Lapas kreisajā augšējā stūrī atrodas uzņēmuma, kas simbolizē, ka šeit vēl jāpievieno individuāls uzņēmuma logo.
2. Navigācijas izvēlne: Augšējā daļā ir navigācijas josla ar sadaļām "Sākumlapa", "Radošais stūrītis", "Sazināties ar mums", un "Admin", kā arī "Pieslēgties" poga labajā augšējā stūrī.
3. Galvenais virsraksts: Lapas centrā ir liels, ievērību piesaistošs virsraksts "Unikāli krekli un krūzes ikvienam!", kas pauž, ka šeit var iegādāties unikālu apģērbu un trauku dizainus.
4. Apakšvirsraksts un apraksts: Zem virsraksta ir apakšvirsraksts "Pielāgo kreklus, krūzes. Izbaud stilu. Izcelies unikāli.", kas aicina pārvērst idejas dizainos jūsu apģērbu kolekcijai. Zem tā atrodas teksts, kas aicina radīt lieliskus dizainus, izmantojot bezmaksas rīkus, un izvēlēties produktus no piedāvātajiem aksesuāriem.
5. Produktu izvēlne: Zem apraksta ir divas ikonas - viena T-krekla, otra krūzes attēlojumam. Šie attēli kalpo kā saites, kas ļauj lietotājiem viegli izvēlēties, ko viņi vēlas personalizēt.

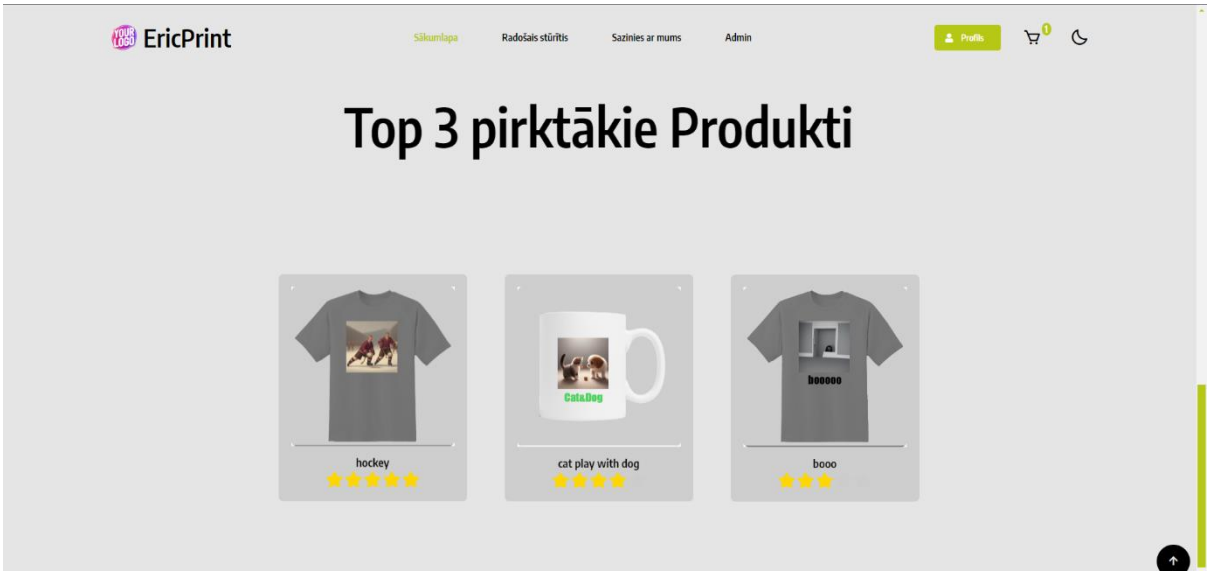


9.att. Sākumlapa cenrādis

9.Attēls redzams divas produktu kartes no tīmekļa lapas, kurās piedāvāti personalizēti produkti: T-krekls un krūze.

Produktu karte redzams:

Produkta bilde, produkts nosaukums, produuktu apraksts, produktu cena, poga: "Izveido dizainu" – aicina lietotāju personalizēt produktu.

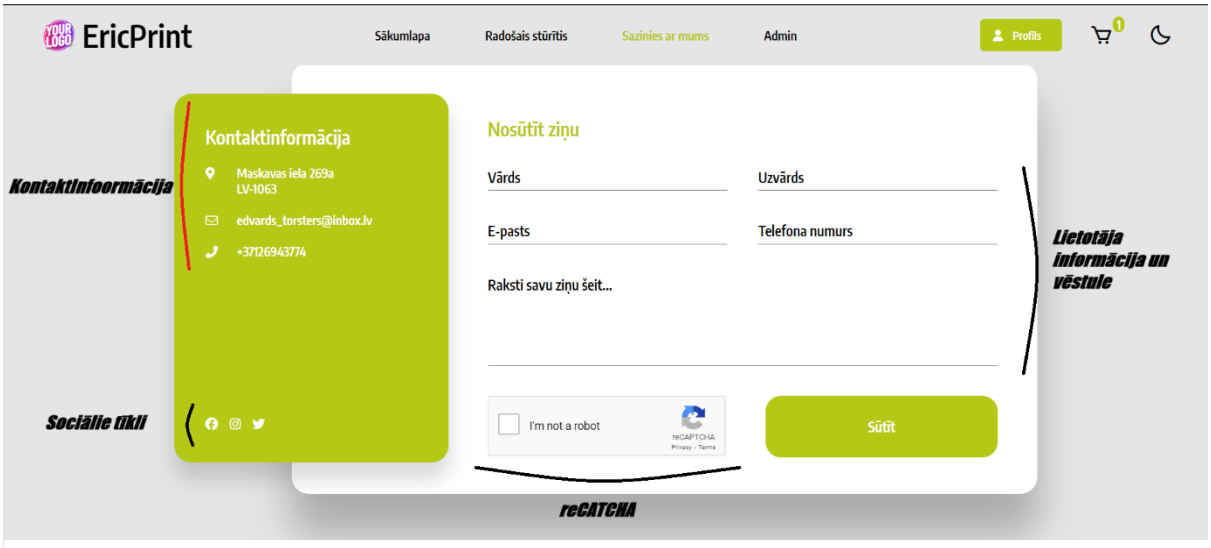


10.att. Sākumlapa

10.Attēlā redzama tīmekļa lapas sadaļa ar nosaukumu "Top 3 pirktākie Produkti", kurā parādīti trīs visvairāk pārdotie un novērtētākie produkti:

Katra produkta apakšā ir zvaigznes, kas norāda uz klientu vērtējumu, un nosaukums, kas raksturo produktu vai tā dizainu. Tas palīdz pircējiem ātri iegūt informāciju par populārākajiem produktiem un to popularitāti.

6.3.1. Sazināties ar mums lapa:



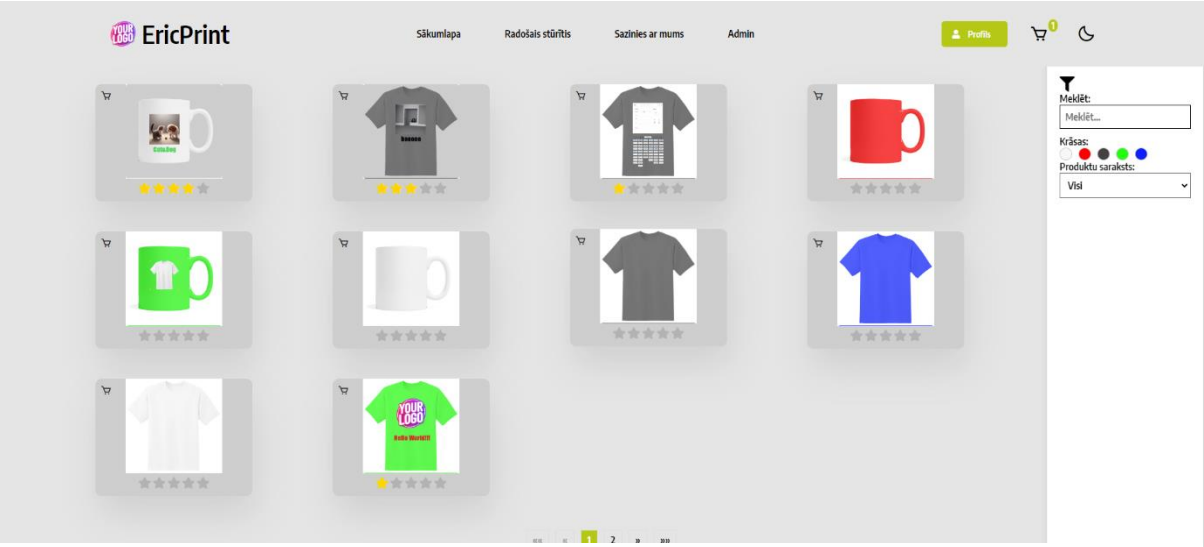
11.att. Sazināties ar mums lapa

11. Attēlā ir redzama EricPrint tīmekļa lapas sadaļa, kas paredzēta saziņai ar uzņēmumu lai tiktu šaja lapa jāieiet pārlūk lapā https://ericprint.com/lv/contact-us/. Lapā ir divas galvenās daļas:

1. Kontaktinformācijas panelis:
   - Šajā zaļajā panelī ir iekļauta fiziskā adrese ("Maskavas iela 269a, LV-1063"), e-pasta adrese ("edvards_torsters@inbox.lv") un tālruņa numurs ("+37126943774").
   - Zem kontaktinformācijas ir ikonas sociālajiem medijiem (Facebook, Instagram, Twitter), kas ļauj ātri piekļūt uzņēmuma sociālo mediju lapām.
2. Lietotāja ziņas sūtīšanas forma:
   - Šajā baltajā formā lietotāji var ievadīt savu vārdu, uzvārdu, e-pasta adresi un tālruņa numuru, kā arī rakstīt ziņu uzņēmumam.
   - Apakšā ir pogas "Sūtīt" un reCAPTCHA aizsardzība pret automatizētiem ziņojumiem, nodrošinot, ka ziņas sūta īsti cilvēki.

Šī sadaļa ir nozīmīga klientu atbalsta un komunikācijas veicināšanai, ļaujot apmeklētājiem ērti sazināties ar uzņēmumu.

6.3.2. Radošais stūītis lapa



12.att. Radošais stūrītis

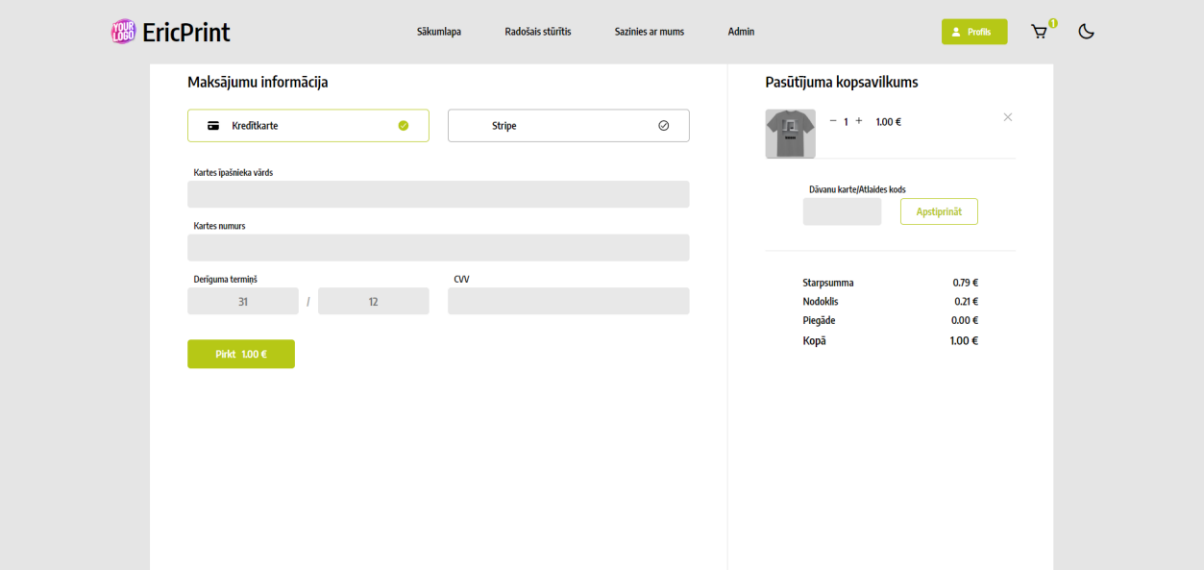
12. Attēlā redzama tīmekļa lapas produkta galerija, kurā tiek izvietoti dažādi personalizējami produkti, piemēram, T-krekli un krūzes, ar dažādiem dizainiem un klientu novērtējumiem.

Galvenie elementi:

1. Produktu Kartes:
   - Katra karte attēlo produktu, piemēram, T-kreklu vai krūzi, ar unikālu dizainu.
   - Produktiem ir pievienotas zvaigznītes, kas norāda uz klientu novērtējumiem, no piecām iespējamām zvaigznēm.
   - Produktu dizaini ir dažādi: no attēliem ar tekstiem līdz grafiskiem zīmējumiem un krāsainiem izdrukām.
2. Navigācijas Iespējas:
   - Lapas augšējā labajā stūrī ir meklēšanas josla, kur var ievadīt meklēšanas atslēgvārdus, lai atrastu konkrētus produktus pēc nosaukuma vai pec autora.
   - Zem meklēšanas joslas ir krāsu izvēles opcijas, kas ļauj filtrēt produktus pēc krāsas.
   - Ir arī izvēlne "Produktu saraksts", kur var izvēlēties produktu kategoriju, kas jāparāda.
3. Lapas Navigācija:
   - Produkta kartes ir izvietotas rindās, un lapas apakšā ir lapu navigācijas numuri, kas ļauj pārvietoties starp vairākām galerijas lapām.

Šī produkta galerija ir noderīgs rīks lietotājiem, lai ātri atrastu un personalizētu produktus pēc savām vēlmēm, izmantojot pieejamās filtrēšanas un meklēšanas funkcijas.

6.3.3. Groza lapa



13.att. Grozs

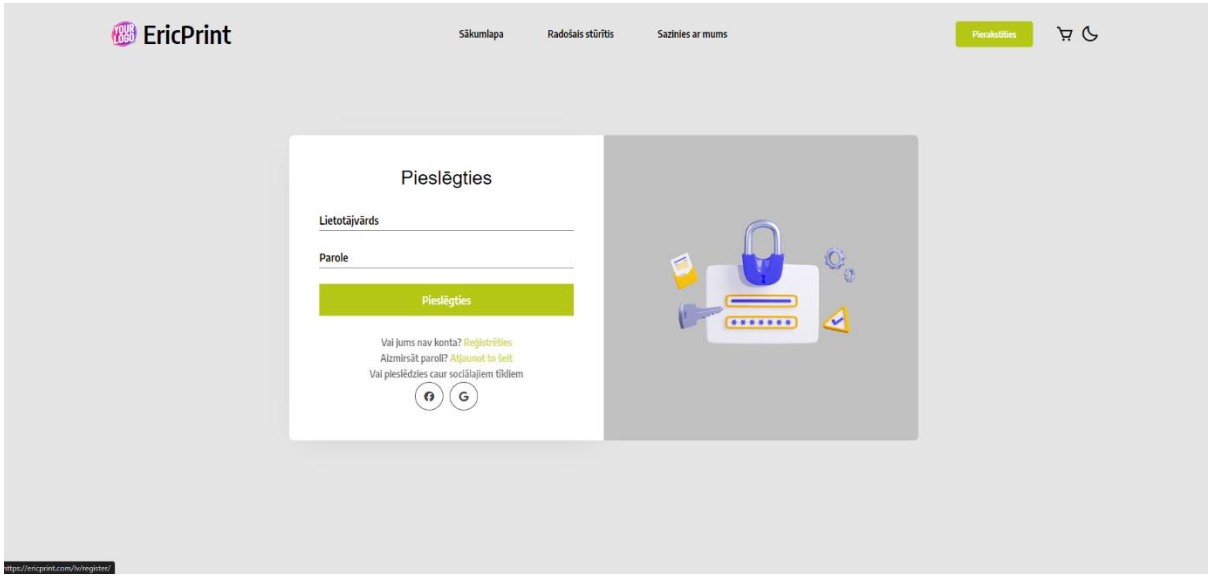
13. Attēlā redzama tīmekļa lapas maksājumu un pasūtījuma apstrādes sadaļa:

1. Maksājumu informācija:
    - Iespēja veikt maksājumu ar kredītkarti.
    - Lietotājam jāievada kredītkartes informācija: kārtes īpašnieka vārds, kartes numurs, derīguma termiņš un CV kods.
    - Maksājuma pogas dizains ir zaļš, norādot iespēju veikt maksājumu.
2. Pasūtījuma kopsavilkums:

- Attēlots produkti ar cenām, kuras ir sadalītas starpsummā, nodoklī un piegādes izmaksās.
- Galīgā summa, kas jāapmaksā.
- Ir lauks dāvanu kartes vai atlaižu koda ievadīšanai, ar pogu "Apstiprināt".

Šī sadaļa sniedz galīgo soli pirms pasūtījuma apstiprināšanas, ļaujot klientam pārbaudīt savu pasūtījumu un veikt maksājumu.

6.3.4. Pieslēgšanās un Reģistrācijas lapa



14.att. Pieslēgšanās lapa

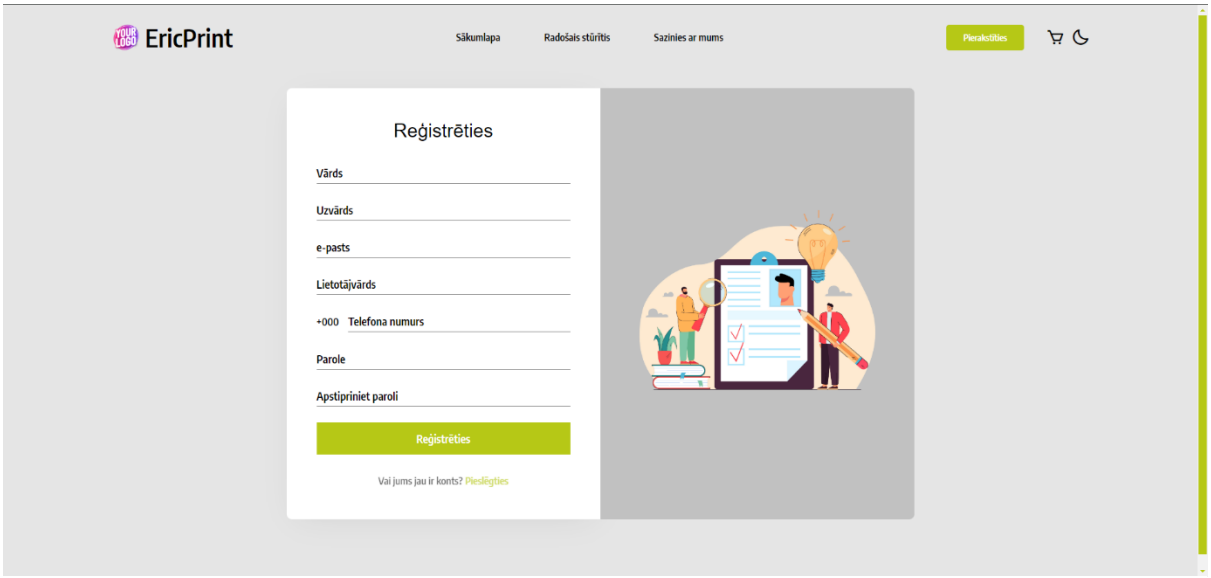
14. Attēlā ir redzama tīmekļa vietnes pieteikšanās lapa, kas ir sadalīta divās daļās: kreisajā un labajā pusē.

Kreisajā pusē:

- Virsraksts: Lapas augšpusē melniem burtiem rakstīts "Pieslēgties".
- Ievades lauki: Zem virsraksta ir divi ievades lauki, kur lietotājam jāievada "Lietotājvārds" un "Parole". Ievades lauki ir norobežoti ar melnām līnijām.
- Poga: Zem ievades laukiem ir liela, zaļa poga ar tekstu "Pieslēgties". Šī poga ir paredzēta, lai lietotājs varētu iesniegt savus pieteikšanās datus.
- **Papildu saites:**
    - "Vai jums nav konta? Reģistrēties" – aicinājums izveidot jaunu kontu, ja lietotājam tāda vēl nav.
    - "Aizmirsāt paroli? Atjaunot to šeit" – saite, kas palīdz lietotājam atjaunot aizmirstu paroli.
    - "Vai pieslēdzies caur sociālajiem tīkliem" – aicinājums pieslēgties, izmantojot sociālo tīklu kontus.
    - Sociālo tīklu ikonas: Apakšā ir divas apaļas ikonas, viena ar Facebook logotipu un otra ar Google logotipu, lai lietotāji varētu pieteikties, izmantojot savus sociālo tīklu kontus.

Labajā pusē:

- Bilde: Šajā sadaļā redzams ilustratīvs attēls.



15.att. Reģistrēšanās lapa

15. Atēllā Fons abām sadaļām ir pelēks, bet pati pieteikšanās forma ir izvietota baltā taisnstūra rāmī, kas piešķir tai tīru un mūsdienīgu izskatu.

Šajā attēlā ir redzama tīmekļa vietnes reģistrācijas lapa, kas ir sadalīta divās daļās: kreisajā un labajā pusē.

Kreisajā pusē:

- Virsraksts: "Reģistrēties" melnā krāsā.
- Ievades lauki: Seši ievades lauki, kur lietotājam jāievada:

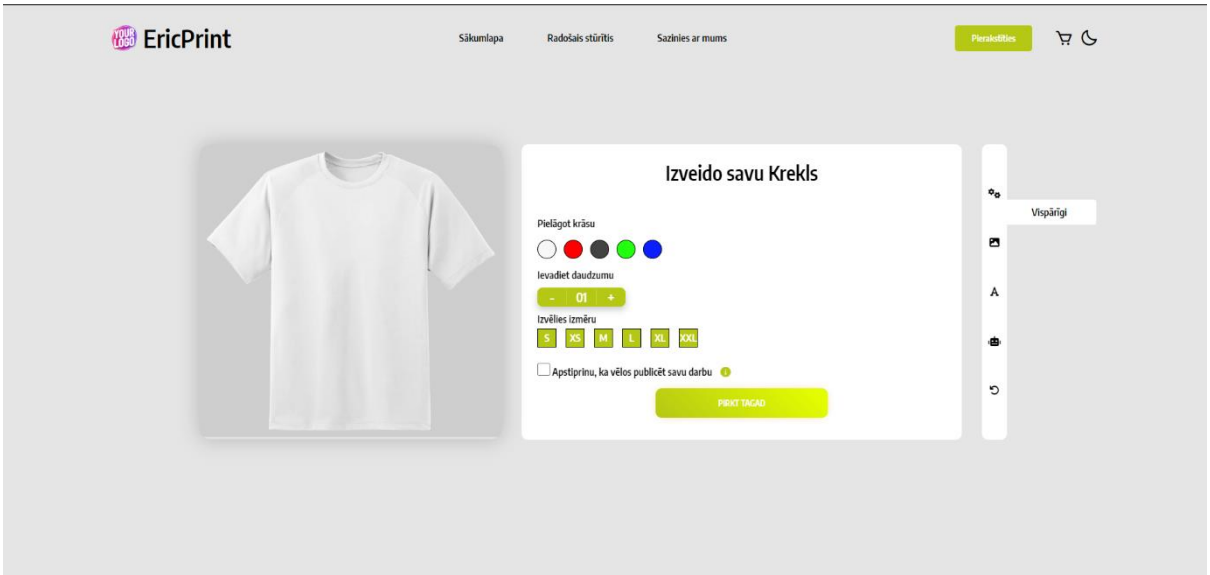
  (Vārds, Uzvārds, e-pasts, Lietotājvārds, Telefona numurs, Parole, Apstiprināt paroli.)
- Poga: Zaļā poga ar tekstu "Reģistrēties".
- Papildu saite: "Vai jums jau ir konts? Pieslēgties" – aicinājums pieslēgties, ja lietotājam jau ir konts.

Labajā pusē:

- Bilde: Šajā sadaļā redzams ilustratīvs attēls.

Fons abām sadaļām ir pelēks, bet pati reģistrācijas forma ir izvietota baltā taisnstūra rāmī, kas piešķir tai tīru un mūsdienīgu izskatu.

6.3.5. Dizaina rīka lapa



16.att. Dizaina rīka pilnā lapa

16. Attēlā ir redzama tīmekļa vietnes lapa, kas paredzēta personalizētu kreklu izveidei un pasūtīšanai. Lapa ir sadalīta divās galvenajās daļās: kreisajā pusē ir kreklu priekšskatījums, bet labajā pusē ir izvēlnes un iespējas personalizēšanai un pasūtīšanai.

Kreisā puse:

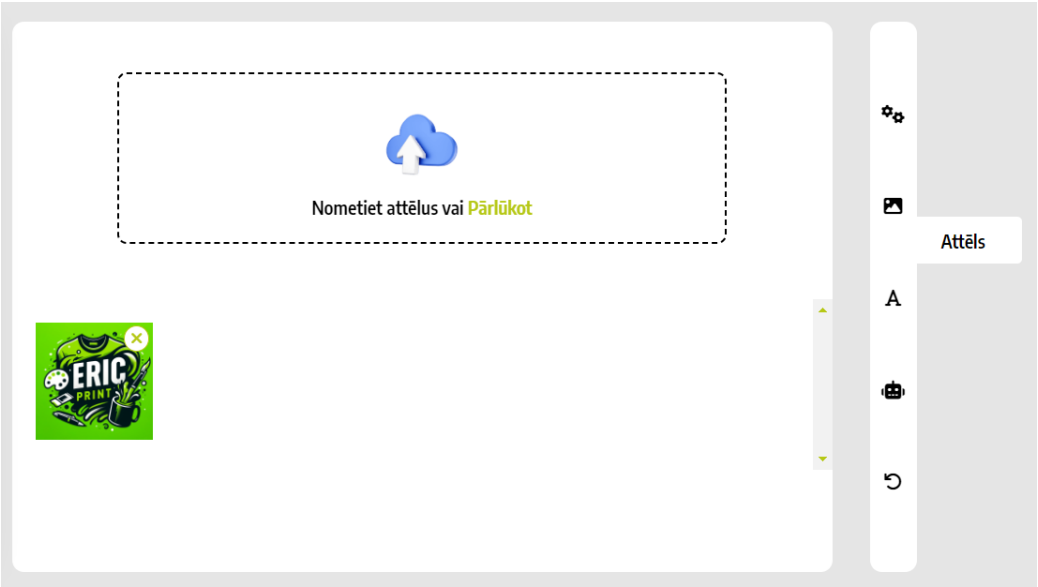
- Produktu priekšskatījums: Lapas kreisajā pusē ir redzams balts krekls, kurš kalpo kā personalizējamā krekla paraugs.

Labā puse:

- Virsraksts: "Izveido savu Kreklu" melnā krāsā.
- Krāsas izvēle: Piecas krāsu pogas, kuras var izvēlēties krekla krāsu – balta, sarkana, melna, zaļa un zila.
- Daudzuma ievade: Skaitītājs daudzuma izvēlei, kas sākas no 1. Lietotājs var palielināt vai samazināt daudzumu ar plusa un mīnusa pogām.
- Izmēra izvēle: Izmēru pogas ar izvēlēm S, XS, M, L, XL un XXL.
- Papildu opcijas:
    - Apstiprinājuma izvēles rūtiņa ar tekstu "Apstiprinu, ka vēlos publicēt savu darbu".
- Poga: Zaļa poga ar tekstu "PIRKT TAGAD", kas paredzēta pasūtījuma veikšanai.

Papildu izvēlnes:

- Labās malas vertikālajā joslā ir vairākas ikonas, kas piedāvā papildu opcijas, piemēram, vispārīgos iestatījumus, attēlu pievienošanu, teksta pievienošanu un citas opcijas.



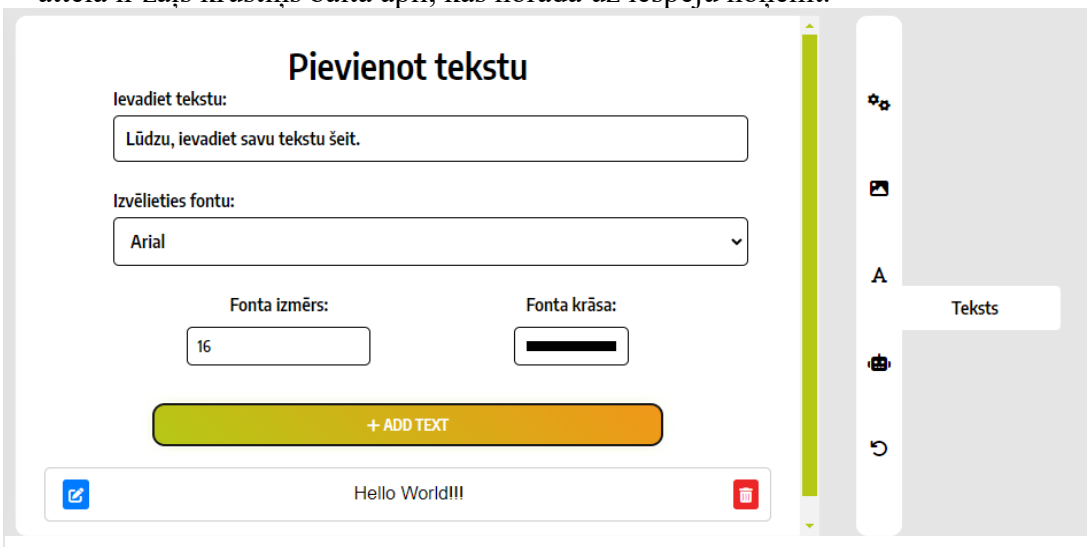
17.att. Dizaina rīka bildes pievienošanas logs

Augšējā daļa:

- Augšupielādes laukums: Centrā ir liels taisnstūrveida laukums ar pārtrauktu līniju robežu. Šajā laukā ir mākoņa ikona ar augšupvērstu bultiņu.
- Teksts laukuma centrā: "Nometiet attēlus vai Pārlūkot". Teksts norāda, ka lietotājs var vilkt un nomest attēlus šajā laukā vai izvēlēties tos, izmantojot pārlūkprogrammas funkciju.

Apakšējā daļa:

- Augšupielādēts attēls: Kreisajā apakšējā stūrī ir redzams augšupielādētie attēli. Uz attēla ir zaļš krustiņš baltā aplī, kas norāda uz iespēju noņemt.



18.att. Dizaina rīka teksta pievienošanas logs

18. Attēlā ir redzama tīmekļa vietnes sadaļa, kas paredzēta teksta pievienošanai un rediģēšanai.

Augšējā daļa:

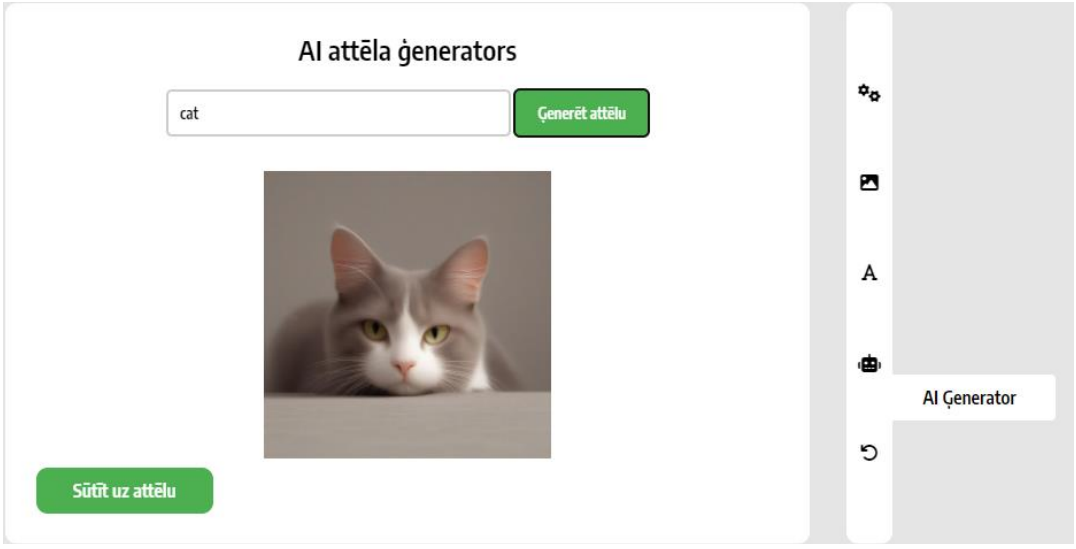
- Ievades lauks: Apzīmēts ar "Ievadiet tekstu:", un zemāk ir ievades lauks ar tekstu "Lūdzu, ievadiet savu tekstu šeit."
- Fonta izvēle: Apzīmēts ar "Izvēlieties fontu:", un zemāk ir nolaižamā izvēlne, kurā ir izvēlēts fonts "Arial".
- Fonta izmērs: Apzīmēts ar "Fonta izmērs:", un zemāk ir ievades lauks ar vērtību "16".
- Fonta krāsa: Apzīmēts ar "Fonta krāsa:", un zemāk ir taisnstūris ar melnu krāsu, kas norāda izvēlēto fonta krāsu.

Poga:

- Poga: Zaļa poga ar tekstu "+ ADD TEXT", kas paredzēta, lai pievienotu tekstu.

Pievienotais teksts:

- Apakšā ir teksts pelēkā rāmī ar rediģēšanas ikonu kreisajā pusē un dzēšanas ikonu labajā pusē.



19.att. Dizaina rīka AI generators

19. Attēlā ir redzama tīmekļa vietnes sadaļa, kas paredzēta AI attēlu ģenerēšanai.

Augšējā daļa:

- Ievades lauks: Balts ievades lauks, kur lietotājs var ievadīt vajadzīgo attēla tēmu vai atslēgvārdus.

- Poga: Zaļa poga ar tekstu "Ģenerēt attēlu", kas paredzēta attēla ģenerēšanas procesa sākšanai.

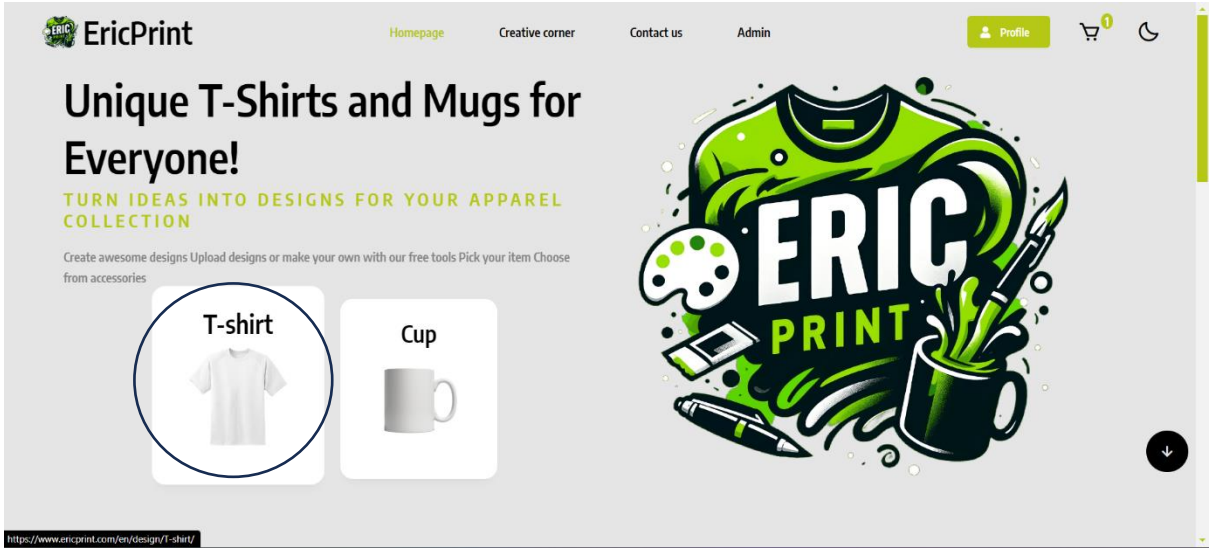
Poga:

- Apakšā kreisajā pusē ir zaļa poga ar tekstu "Sūtīt uz attēlu", kas, iespējams, ir paredzēta, lai nosūtītu ģenerēto attēlu tālākai izmantošanai vai saglabāšanai.

6.4.Testa Piemērs

Šis testa piemērs apraksta darbības, kas jāveic, lai izveidotu personalizētu T-kreklu, izmantojot EricPrint platformu.

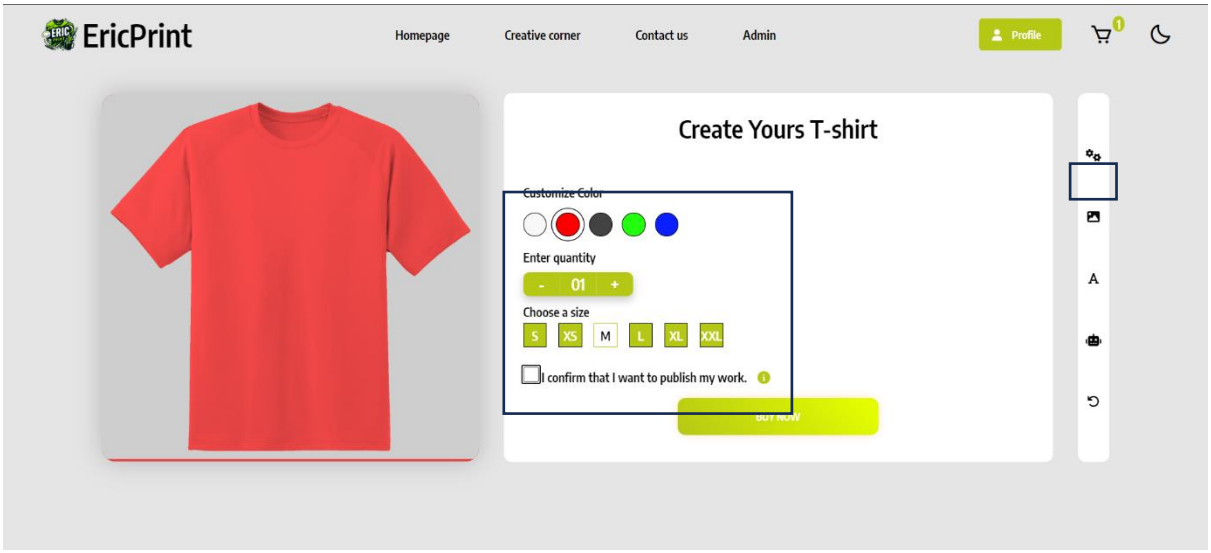
1. Mēs izvēlamies kreklu (20.att.)
    - Atvērsim "Create Your T-shirt" sadaļu, kur ir redzams dažādu kreklu piedāvājums.
    - Piemēram, mēs izvēlēsimies sarkanu kreklu (redzams otrajā attēlā).



20.att. Sākumlapa

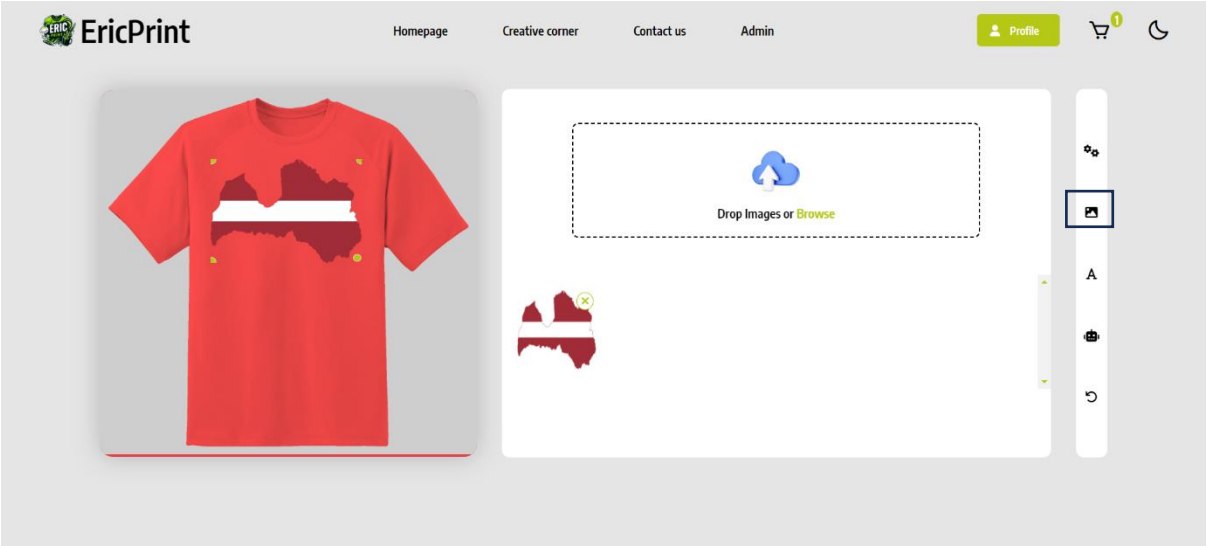
2. Mēs izvēlamies krāsu un izmēru (21.att.)
    - Pieejamās krāsas: balta, sarkana, zaļa, zila.
    - Izvēlēsimies sarkanu krāsu.

- Izvēlēsimies izmēru, piemēram, "M" (Medium).



21.att. Dizaina rīka vispārīgie iestatijumi

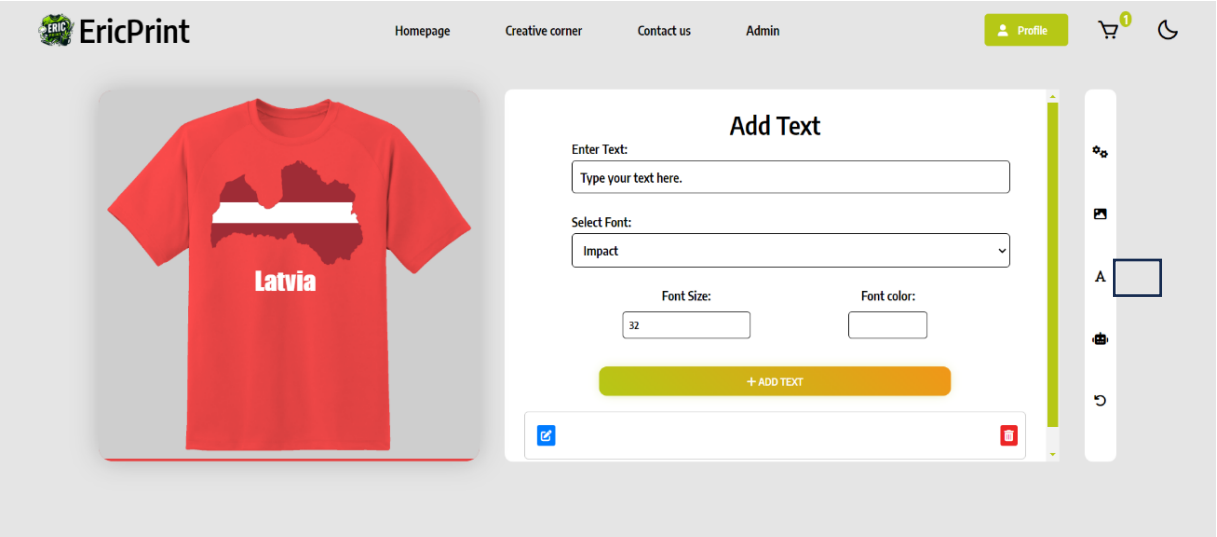
3. Mēs pievienojam bildi (22.att.)
    - Izvēlamies bildes augšuplādi.
    - Augšupielādējam vēlamo bildi.
    - Piemēram, Latvijas kontūru ar karogu, kā redzams trešajā attēlā.



22.att. Dizaina rīka bildes piwvienošana

4. Mēs pievienojam tekstu (23.att.)
    - Izvēlamies teksta pievienošanu
    - Ierakstām tekstu, piemēram, "Latvia".
    - Izvēlamies fontu, piemēram, Impact, un fonta izmēru (32).
    - Teksta krāsu varam atstāt balto.

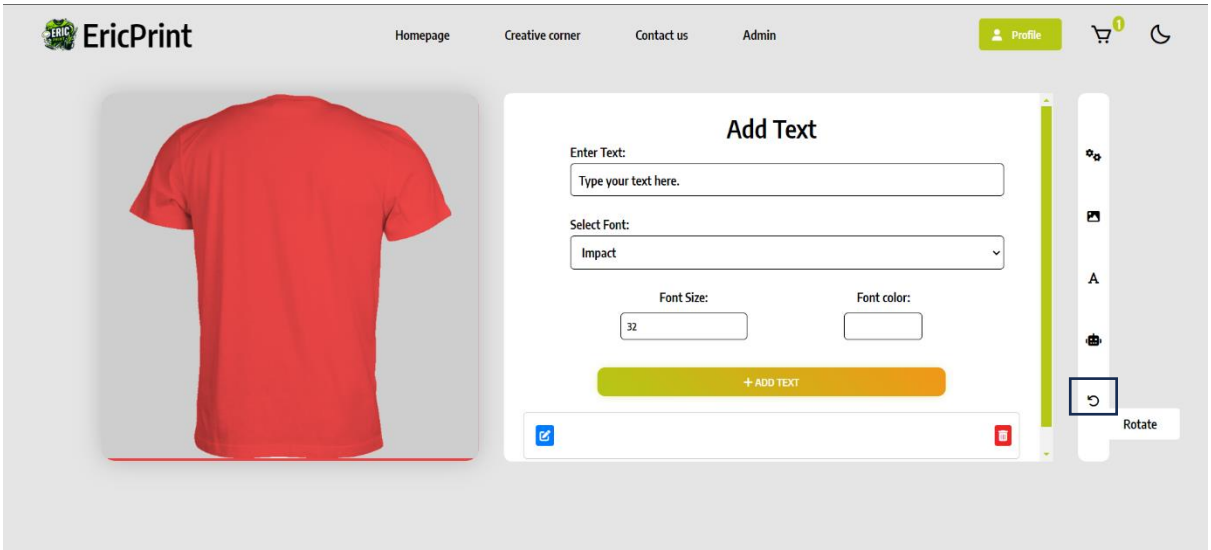
- Teksta pievienošana kā redzams ceturtajā attēlā.



23.att. Dizaina rīka teksta pievienošanas logs

5. Mēs apgriežam kreklu uz otru pusi (24.att.)

    - Apgriežam kreklu, lai varam redzēt mugurpusi (piektajā attēlā).



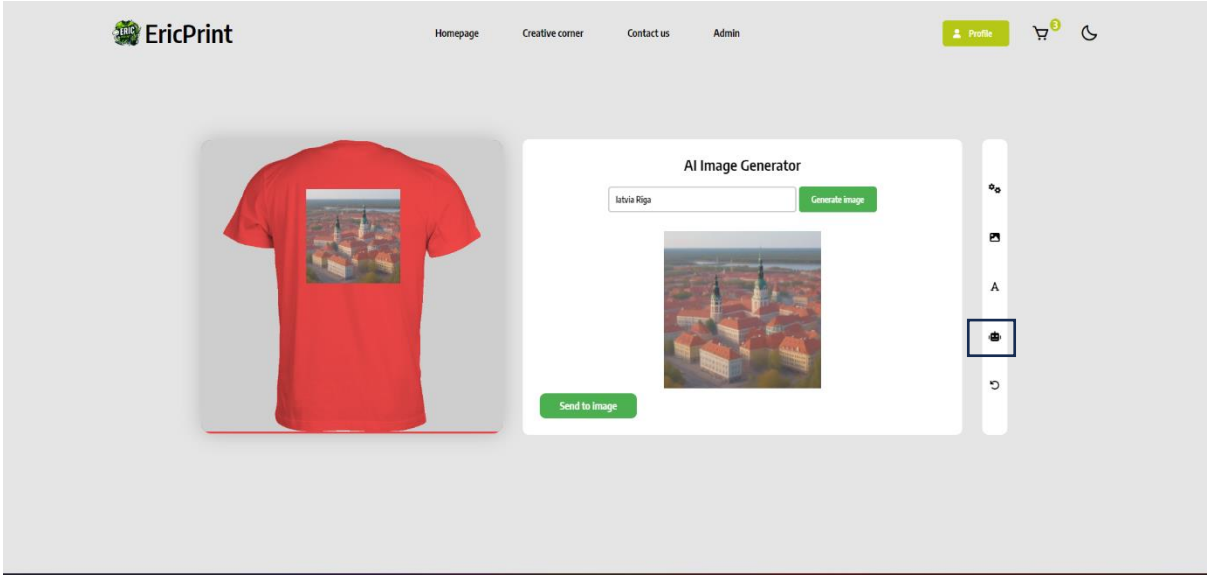
24.att. Dizaina rīka produkta apgriešana

6. Mēs izveidojam ar AI bildi "Latvia Riga" (25.att.)

    - Izvēlamies Ai generātoru

    - Izmantojam AI Image Generator, ierakstām "Latvia Riga" un ģenerējam bildi.

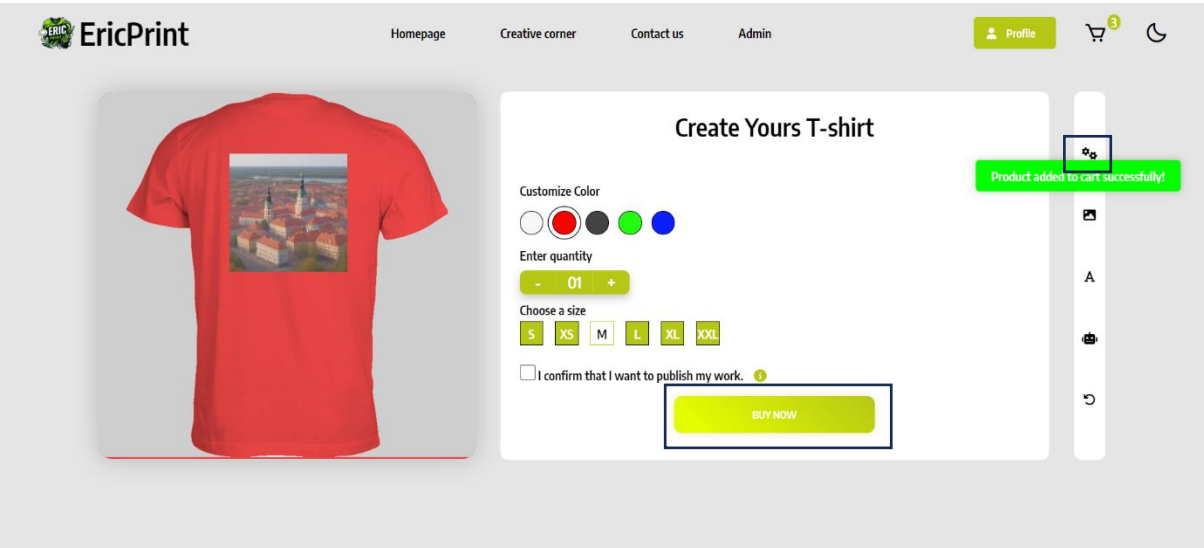
- Piemērs redzams sestajā attēlā, kur ģenerēta Rīgas panorāma.



25.att. Dizaina rīka AI generatora bildes pievienošana

7. Mēs ieliekam grozā kreklu (26.att.)

   - Ejam atpakal uz visparīgs
   - Pārliecināmies, ka viss ir kārtībā un pievienojam kreklu grozā.
   - Galarezultāts ir redzams septītajā attēlā, kur parādās apstiprinājums "Product added to cart successfully!".



26.att. Dizaina rīka vispārīgie iestatijumi

NOBEIGUMS

Šī kvalifikācijas darba ietvaros tika izstrādāta tīmekļa lietojumprogramma, kas ļauj lietotājiem viegli un intuitīvi izveidot, pasūtīt un iegādāties personalizētas dizaina preces, piemēram, apģērbu un aksesuārus. Projekta mērķis bija izveidot funkcionālu un lietotājam draudzīgu platformu, kas atbilstu mūsdienu digitālās pasaules prasībām un lietotāju vajadzībām.

Projekta izstrādes gaitā tika izmantotas dažādas tehnoloģijas, lai nodrošinātu lietojumprogrammas funkcionalitāti un veiktspēju. Projekts sastāv no šādām galvenajām sastāvdaļām:

- JavaScript (72.5%): Galvenā programmēšanas valoda, kas nodrošina interaktivitāti un funkcionalitāti lietotāja saskarnē.
- CSS (15.5%): Stilu lapas, kas nodrošina lietotāja saskarnes vizuālo izskatu un dizainu.
- HTML (11.5%): Struktūras veidošanai un satura organizēšanai tīmekļa lapās.
- Python (0.5%): Back-end apstrādes un servera puses loģikas nodrošināšanai.

Izstrādātā lietojumprogramma piedāvā plašu funkcionalitāti, tostarp iespēju pieslēgties, izmantojot sociālos tīklus, viegli lietojamu dizaina rīku, drošu maksājumu sistēmu un intuitīvu pasūtījumu veikšanas procesu. Lietotāji var pielāgot produktus pēc saviem ieskatiem, izvēloties krāsas, izmērus, pievienojot attēlus un tekstu, kā arī pielāgojot teksta fontus, izmērus un krāsas.

Testa piemērs demonstrēja sistēmas darbību, parādot, kā lietotājs var izveidot personalizētu krekla dizainu un veikt pirkumu. Šis piemērs apstiprināja sistēmas lietošanas ērtumu un funkcionalitāti.

Šī projekta izstrāde ilga četrus mēnešus, kuru laikā tika veikta rūpīga plānošana, programmēšana, testēšana un pilnveidošana, lai nodrošinātu augstas kvalitātes un lietotājam draudzīgu rezultātu. Kopumā projekts sasniedza izvirzītos mērķus, izstrādājot modernu un lietotājam draudzīgu tīmekļa lietojumprogrammu personalizētu dizaina preču izveidei un pasūtīšanai. Šī lietojumprogramma var kalpot kā pamats turpmākai attīstībai un paplašināšanai, piedāvājot vēl vairāk iespēju un funkcionalitātes lietotājiem.

INFORMĀCIJAS AVOTI

1. Django administratora paneļa bibleoteka (15.02.2024) - https://django-jazzmin.readthedocs.io/
2. Python valodas bibleoteka django instrukcija (22.02.2024)- https://django-jazzmin.readthedocs.io/
3. Auterizākcijas sistēme django (09.4.2024)- https://docs.djangoproject.com/en/4.2/topics/auth/default/
4. Par django datubāzēm (28.03.2024)- https://docs.djangoproject.com/en/4.2/ref/databases/
5. MDN Web Docs (12.05.2024)– https://developer.mozilla.org/
6. WebRTC dokumentācija (05.05.2024)- https://webrtc.org/
7. MySQL dokumentācija (19.04.2024)- https://dev.mysql.com/doc/

Pielikumi

Pielikums 1

Datubāze:

```
class user(AbstractUser):
    phone_number = PhoneNumberField(blank=True, null=True)

    def __str__(self):
        return self.username

    class Meta:
        verbose_name = _("Lietotājs")
        verbose_name_plural = _("Lietotāji")


class Product(models.Model):
    title = models.CharField(_('Virsraksts'), max_length=100, unique=True,
blank=False, help_text=_("Ievadiet produkta nosaukumu."))
    image = models.ImageField(_('Bilde'), upload_to='products/', blank=False)
    slug = models.SlugField(_('Slug'), unique=True, help_text=_("Ievadiet URL
draudzīgu nosaukumu."), blank=True)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    options = models.TextField(blank=True, help_text="Ievadiet opcijas kā
sarakstu ar komatiem")

    front_image_with_background = models.ImageField(_('Priekšējā bilde ar
fonu'), upload_to='products/', blank=True)
    front_image_not_background = models.ImageField(_('Priekšējā bilde bez
fona'), upload_to='products/', blank=True)
    back_image_with_background = models.ImageField(_('Aizmugurējā bilde ar
fonu'), upload_to='products/', blank=True)
    back_image_not_background = models.ImageField(_('Aizmugurējā bilde bez
fona'), upload_to='products/', blank=True)

    available_colors = models.ManyToManyField('Color',
related_name='products', blank=True)
    available_sizes = models.ManyToManyField('Size', related_name='products',
blank=True)

    def __str__(self):
        return self.title

    def get_options_list(self):
        return [option.strip() for option in self.options.split(',')]

    class Meta:
        verbose_name = _("Produkts")
```

```
        verbose_name_plural = _("Produkti")


class Color(models.Model):
    name = models.CharField(max_length=100)
    code = ColorField()

    def __str__(self):
        return self.name

    class Meta:
        verbose_name = _("Krāsa")
        verbose_name_plural = _("Krāsas")


class Size(models.Model):
    name = models.CharField(max_length=100)
    size = models.CharField(max_length=10)

    def __str__(self):
        return self.name

    class Meta:
        verbose_name = _("Izmērs")
        verbose_name_plural = _("Izmēri")


class CustomDesign(models.Model):
    title = models.CharField(_('Virsraksts'), max_length=100)
    description = models.TextField(_('Apraksts'),)
    additional_notes = models.TextField(_('Papildu piezīmes'))
    image = models.ImageField(_('Bilde'), upload_to='page/', blank=False)

    def __str__(self):
        return self.title

    class Meta:
        verbose_name = _("Pielāgots dizains")
        verbose_name_plural = _("Pielāgoti dizaini")


class ContactMessage(models.Model):
    first_name = models.CharField(_('Vārds'), max_length=100)
    last_name = models.CharField(_('Uzvārds'), max_length=100)
    email = models.EmailField(_('E-pasts'))
    phone_number = models.CharField(_('Telefona numurs'), max_length=15)
    message = models.TextField()
    replied = models.BooleanField(_('Atbildēts'), default=False)
```

```
    admin_subject = models.CharField(_('Administrātora virsraksts'),
max_length=255, blank=True, null=True)
    admin_message = models.TextField(_('Administrātora vēstule'), blank=True,
null=True)

    def __str__(self):
        return f"{self.first_name} {self.last_name}"

    class Meta:
        verbose_name = _("Kontaktziņojums")
        verbose_name_plural = _("Kontaktziņojumi")


class Contact(models.Model):
    address = models.CharField(_('Adrese'),blank=True, null=True,
max_length=255)
    postal_code = models.CharField(_('Pasta indekss'),blank=True, null=True,
max_length=20)
    phone_number = models.CharField(_('Telefona numurs'),blank=True,
null=True, max_length=15)
    email = models.EmailField(_('E-pasts'), blank=True, null=True,)
    twitter_link = models.URLField(blank=True, null=True,
verbose_name=_('Twitter saite'))
    facebook_link = models.URLField(blank=True, null=True,
verbose_name=_('Facebook saite'))
    instagram_link = models.URLField(blank=True, null=True,
verbose_name=_('Instagram saite'))

    def __str__(self):
        return f'Kontaktinformācija: {self.address}, {self.postal_code},
{self.phone_number}, {self.email}'

    class Meta:
        verbose_name = _("Kontakts")
        verbose_name_plural = _("Kontakti")

class GiftCode(models.Model):
    code = models.CharField(max_length=50, unique=True)
    is_valid = models.BooleanField(default=True)
    discount_type = models.CharField(max_length=20, choices=[('percentage',
'Percentage'), ('fixed', 'Fixed')])
    discount_value = models.DecimalField(max_digits=10, decimal_places=2)
    min_order_amount = models.DecimalField(max_digits=10, decimal_places=2,
default=0.00)
    start_date = models.DateField(null=True, blank=True)
    end_date = models.DateField(null=True, blank=True)
    quantity = models.IntegerField(null=True, blank=True)
    unlimited_usage = models.BooleanField(default=False)
```

```
    def __str__(self):
        return self.code

    def is_active(self):
        today = timezone.now().date()
        return (
            self.is_valid and
            (not self.start_date or today >= self.start_date) and
            (not self.end_date or today <= self.end_date) and
            (self.unlimited_usage or self.quantity is None or self.quantity >
0)
        )

    class Meta:
        verbose_name = _("Dāvanu kods")
        verbose_name_plural = _("Dāvanu kodi")

class Order(models.Model):
    author = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE)
    publish_product = models.BooleanField(default=False)
    allow_publish = models.BooleanField(default=False)
    front_image = models.TextField(blank=True)
    back_image = models.TextField(blank=True)
    product_color = models.ForeignKey(Color, on_delete=models.CASCADE,
null=True)
    product_amount = models.IntegerField(default=0)
    product_size = models.ForeignKey(Size, on_delete=models.CASCADE,
null=True)
    product = models.ForeignKey(Product, on_delete=models.CASCADE, null=False)
    title = models.CharField(max_length=255, blank=True)
    description = models.TextField(blank=True)
    average_rating = models.FloatField(default=0)

    def update_average_rating(self):
        ratings = Rating.objects.filter(order=self)
        if ratings.exists():
            self.average_rating =
ratings.aggregate(models.Avg('stars'))['stars__avg']
        else:
            self.average_rating = 0
        self.save()


class TextList(models.Model):
    order_text = models.ForeignKey(Order, on_delete=models.CASCADE)
    text = models.TextField(blank=True)
    font = models.CharField(max_length=50, blank=True)
    text_size = models.CharField(blank=True, max_length=25)
```

```
    text_color = models.CharField(max_length=20, blank=True)

class ImageList(models.Model):
    order_images = models.ForeignKey(Order, on_delete=models.CASCADE)
    image = models.TextField(blank=True)

    def __str__(self):
        return f'Image for Order {self.order_images.id}'


class Rating(models.Model):
    user = models.ForeignKey(user, on_delete=models.CASCADE)
    order = models.ForeignKey(Order, on_delete=models.CASCADE)
    stars = models.IntegerField(default=0, validators=[MinValueValidator(0),
MaxValueValidator(5)])

    class Meta:
        verbose_name = _("Vērtējums")
        verbose_name_plural = _("Vērtējumi")

    @staticmethod
    def get_top_rated_popular_products():
        top_rated_popular_products = Order.objects.annotate(
            num_ratings=Count('rating'),
            avg_rating=Avg('rating__stars')
        ).order_by('-num_ratings', '-avg_rating')

        top_three_products = list(top_rated_popular_products[:3])
        product_cycle = cycle(top_three_products)

        return [next(product_cycle) for _ in range(3)]


@receiver(post_save, sender=Rating)
def update_product_average_rating(sender, instance, **kwargs):
    product = instance.order
    product.update_average_rating()

class Purchase(models.Model):
    ORDER_STATUS_CHOICES = [
        ('PENDING', _('Gaida apstiprinājumu')),
        ('PROCESSING', _('Apstrāde')),
        ('SHIPPED', _('Nosūtīts')),
        ('DELIVERED', _('Piegādāts')),
        ('CANCELLED', _('Atcelts')),
    ]
    order_number = models.CharField(max_length=100)
    amount = models.DecimalField(max_digits=10, decimal_places=2)
```

```
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE)
    status = models.CharField(max_length=20, choices=ORDER_STATUS_CHOICES,
default='PENDING')
    created_at = models.DateTimeField(default=timezone.now)

    def __str__(self):
        return f"Order {self.order_number} by {self.user.username}"


class PurchaseProduct(models.Model):
    purchase = models.ForeignKey(Purchase, on_delete=models.CASCADE,
related_name='purchase_products')
    product = models.ForeignKey(Order, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField()

    def __str__(self):
        return f"{self.product.title} - {self.quantity}"
```

Pirkšanas BackEnd:

```
stripe_keys = StripeKeys.objects.first()
if stripe_keys:
    stripe_public_key = stripe_keys.public_key
    stripe_secret_key = stripe_keys.secret_key
    stripe_endpoint_secret = stripe_keys.endpoint_secret
else:
    pass

@csrf_exempt
@login_required(login_url='/account/login/')
def stripe_config(request):
    if request.method == 'GET':
        stripe_config = {'publicKey': stripe_public_key}
        return JsonResponse(stripe_config, safe=False)

def create_checkout_session(request):
    user = request.user
    stripe.api_key = stripe_secret_key
    products_in_cart = request.session.get('cart', {})

    if not products_in_cart:
        return JsonResponse({'error': 'The cart is empty'})

    product_id_list = []
    product_id_list = ",".join(str(product_info['product_id']) for
product_info in products_in_cart.values())


    line_items = []

    for product_id, product_info in products_in_cart.items():
        product = Order.objects.get(id=product_id)

        line_item = {
            'price_data': {
                'currency': 'eur',
                'unit_amount': int(product.product.price * 100),
                'product_data': {
                    'name': product.title,
                    'description': product.description,
                },
            },
```

```
            'quantity': product_info['quantity'],
        }
        line_items.append(line_item)

    try:
        checkout_session = stripe.checkout.Session.create(
            success_url=request.build_absolute_uri(reverse('success')),
            cancel_url=request.build_absolute_uri(reverse('cancel')),
            payment_method_types=['card'],
            line_items=line_items,
            phone_number_collection={"enabled": True},
            # billing_address_collection={'required': True},
            # billing_address_collection={"enabled": True},
            mode='payment',
            metadata={
                'user_id': user.id,
                'Product_id': product_id_list
            }
        )

        return JsonResponse({'sessionId': checkout_session.id})
    except Exception as e:
        return JsonResponse({'error': str(e)})

@csrf_exempt
def stripe_webhook(request):
    if request.method == 'POST':
        endpoint_secret = stripe_endpoint_secret
        payload = request.body
        sig_header = request.META.get('HTTP_STRIPE_SIGNATURE')

        try:
            event = stripe.Webhook.construct_event(
                payload, sig_header, endpoint_secret
            )
        except ValueError as e:
            return HttpResponse(status=400)
        except stripe.error.SignatureVerificationError as e:
            return HttpResponse(status=400)

        if event['type'] == 'checkout.session.completed':
            session = event['data']['object']
            line_items = stripe.checkout.Session.list_line_items(session.id)

            customer_email = session['customer_details']['email']
            product_id = session['metadata']['Product_id']
            user_id = session['metadata']['user_id']
            order_number = session['created']
```

```
            amount = session['amount_total']
            quantity_order = line_items.data[0]['quantity']

            buffer = BytesIO()
            p = canvas.Canvas(buffer, pagesize=letter)
            pdfmetrics.registerFont(TTFont('ArialUnicode', 'arial.ttf'))

            company_name = "Erika druka"
            logo_path = CustomDesign.objects.first().image.path
            logo = ImageReader(logo_path)
            p.drawImage(logo, 50, 670, width=100, height=100)

            company_code = "Firmas kods: XXXXXXXX"
            address = "Ielas nosaukums, Pilsēta, Valsts, Pasta indekss"
            phone = "Telefona numurs: +1234567890"
            email = "E-pasta adrese: info@example.com"

            p.setFillColorRGB(0, 0, 0)
            p.setFont("ArialUnicode", 12)
            p.drawString(200, 750, company_name)

            p.setFont("ArialUnicode", 10)
            p.drawString(200, 730, company_code)
            p.drawString(200, 710, address)
            p.drawString(200, 690, phone)
            p.drawString(200, 670, email)

            order_number_text = f"Pasūtījuma numurs: {order_number}"
            product_id_text = f"Produkta ID: {product_id}"
            user_id_text = f"Lietotāja ID: {user_id}"
            amount_text = f"Pasūtījuma cena: {amount / 100} EUR"
            quantity_order_text = f"Pasūtījuma daudzums: {quantity_order}"
            thank_you_text = "Paldies, ka iepērkaties pie mums!"

            y_coordinate = 600
            for text in [order_number_text, product_id_text, user_id_text,
amount_text, quantity_order_text, thank_you_text]:
                p.drawString(50, y_coordinate, text)
                y_coordinate -= 20

            p.save()
            buffer.seek(0)

            subject = 'Paldies ka iegadajaties produktu no mūsu veikala'
            from_email = 'balticctech@gmail.com'
            to_email = [customer_email]

            text_content = 'Paldies, par pirkumu.'
```

```
            html_content = render_to_string('e-mail/thank_you_email.html',
{'customer_email': customer_email})
            email = EmailMultiAlternatives(subject, text_content, from_email,
to_email)
            email.attach_alternative(html_content, "text/html")

            email.attach('purchase_receipt.pdf', buffer.getvalue(),
'application/pdf')
            email.send()

            metadata = session['metadata']
            product_ids = metadata['Product_id'].split(',')

            user_id = metadata['user_id']

            order_number = session['created']
            amount = session['amount_total']
            quantity_order = line_items.data[0]['quantity']

            purchase = Purchase.objects.create(
                order_number=order_number,
                amount=amount / 100,
                user_id=user_id
            )

            for product_id, item in zip(product_ids, line_items.data):
                PurchaseProduct.objects.create(
                    purchase=purchase,
                    product_id=product_id,
                    quantity=item['quantity']
                )

        return HttpResponse(status=200)
    else:
        return HttpResponse(status=400)

class SuccessView(TemplateView):
    template_name = 'cart.html'


class CancelledView(TemplateView):
    template_name = 'cart.html'
```

Dizaina veidošanas js:

```
// -----------Side-----------
document.addEventListener('DOMContentLoaded', function() {
    let currentSide = localStorage.getItem('currentSide');

    if (!currentSide || (currentSide !== 'front' && currentSide !== 'back')) {
        currentSide = 'front';
        localStorage.setItem('currentSide', currentSide);
    } else if (currentSide === 'back') {
        currentSide = 'front';
        localStorage.setItem('currentSide', currentSide);
    }

// -----------Color-----------
    const colorElements = document.querySelectorAll('.color-select');
    const colorContainer = document.querySelector('.color');

    colorElements.forEach(function(colorElement, index) {
        colorElement.addEventListener('click', function() {
            const selectedColor = colorElement.style.backgroundColor;

            colorContainer.style.backgroundColor = selectedColor;

            colorElements.forEach(function(element) {
                element.classList.remove('active-color');
            });
            colorElement.classList.add('active-color');
        });
    });

// -----------quantity-content-----------

    const plus = document.querySelector(".plus"),
        minus = document.querySelector(".minus"),
        num = document.querySelector(".num");

    let a = 1;

    plus.addEventListener('click', ()=>{
        a++;
        a = (a < 10) ? '0' + a : a
        num.innerText = a;
    })

    minus.addEventListener('click', ()=>{
```

```
        if(a > 1){
            a--;
            a = (a < 10) ? '0' + a : a;
            num.innerText = a;
        }
    })

    // -----------Size-----------
    var sizeOptions = document.querySelectorAll('.size-option');

    sizeOptions.forEach(function(option) {
        option.addEventListener('click', function() {

            sizeOptions.forEach(function(opt) {
                opt.classList.remove('active');
            });

            this.classList.add('active');
        });
    });

// -----------Info-----------

    var infoIcon = document.querySelector('.info-icon');
    var modal = document.getElementById('info-modal');
    var closeBtn = document.querySelector('.close');

    infoIcon.addEventListener('click', function() {
        modal.style.display = 'block';
    });

    closeBtn.addEventListener('click', function() {
        modal.style.display = 'none';
    });

    window.addEventListener('click', function(event) {
        if (event.target == modal) {
            modal.style.display = 'none';
        }
    });

// -----------Image-----------
    $('.upload-area').click(function() {
        $('#upload-input').trigger('click');
    });

    $('#upload-input').change(event => {
        if (event.target.files) {
```

```
            handleFiles(event.target.files);
        }
    });

    $('.upload-area').on('drop', function(event) {
        event.preventDefault();
        let files = event.originalEvent.dataTransfer.files;
        handleFiles(files);
    });

    $('.upload-area').on('dragover', function(event) {
        event.preventDefault();
    });



    let resizableElement = null;
    let previousImageWidth = 0;

$(document).on('mousedown', '.editable-image', function(event) {
    if (resizableElement && !$(this).is(resizableElement)) {
        resizableElement.resizable("destroy");
    }

    $(this).resizable({
        handles: 'ne, se, sw, nw, middle-n, middle-w, middle-s, middle-e',
        ghost: false,
        border: false
    });

    resizableElement = $(this);

    previousImageWidth = resizableElement.width();
    $('.ui-wrapper').draggable();
});


$(document).on('click', '.remove-btn', function() {
    let imageIdToRemove = $(this).parent().data('image-id');

    $('[data-image-id="' + imageIdToRemove + '"]').remove();

    let listItemToRemove = $('.text-list-item').eq(imageIdToRemove);
    listItemToRemove.remove();
});

$(document).on('mousedown', function(event) {
    if (!$(event.target).is('.editable-image')) {
        let newImageWidth = resizableElement.width();
```

```
        if (Math.abs(newImageWidth - previousImageWidth) > 10) {
            resizableElement.resizable("destroy");
            resizableElement = null;
        }

        previousImageWidth = 0;
    }
});

    function handleFiles(files) {
        if (files) {
            let filesAmount = files.length;

            for (let i = 0; i < filesAmount; i++) {
                let reader = new FileReader();
                reader.onload = function(event) {
                    let imageId = Date.now();

                    let htmlList = `
                    <div class='uploaded-img ${currentSide}' data-image-
id='${imageId}' id='save-img'>
                        <img src='${event.target.result}' draggable='true'>
                        <button type='button' class='remove-btn'>
                            <i class='fas fa-times'></i>
                        </button>
                    </div>
                    `;
                    $('.upload-img').append(htmlList);

                    let htmlKrekls = `
                    <div class='uploaded-img element-image ${currentSide}'
data-image-id='${imageId}'>
                        <img src='${event.target.result}' class='editable-
image resizable-image' draggable='true'>
                    </div>
                `;
                    let selectedContainer;
                    if (currentSide === 'front') {
                        selectedContainer = $('#front');
                    } else {
                        selectedContainer = $('#back');
                    }
                    let parentContainer = $('<div class="parent-container"
style="position: relative; width: 100%; height: 100%;"></div>');

                    selectedContainer.prepend(htmlKrekls);
```

```
                    $('.remove-btn').click(function() {
                        let imageIdToRemove = $(this).parent().data('image-
id');
                        $('[data-image-id="' + imageIdToRemove +
'"]').remove();
                    });

                    $(`.uploaded-img[data-image-id='${imageId}'] .editable-
image`).resizable({
                        handles: 'ne, se, sw, nw, middle-n, middle-w, middle-
s, middle-e',
                        ghost: false,
                        border: false
                    });

                    $('.ui-wrapper').draggable();


                };
                reader.readAsDataURL(files[i]);
            }

            $('.upload-img').css('padding', '20px');
        }
    }

    $('.remove-btn').click(function() {
        let imageIdToRemove = $(this).parent().data('image-id');
        $('[data-image-id="' + imageIdToRemove + '"]').remove();
    });

    const generalContent = document.getElementById('general');
    const uploadContent = document.getElementById('upload');
    const AiContent = document.getElementById('Ai-generator');
    const Text = document.getElementById('Text');
    const Back = document.getElementById('back');
    const Front = document.getElementById('front');

    const generalIcon = document.getElementById('generalIcon');
    const imageIcon = document.getElementById('imageIcon');
    const textIcon = document.getElementById('textIcon');
    const ai = document.getElementById('Ai-generatoricon');
    const rotate = document.getElementById('roatateicon');

    generalIcon.addEventListener('click', function() {
        showContent(generalContent);
        hideContent(uploadContent);
        hideContent(AiContent);
        hideContent(Text);
```

```
});

textIcon.addEventListener('click', function() {
    showContent(Text);
    hideContent(uploadContent);
    hideContent(AiContent);
    hideContent(generalContent);
});

imageIcon.addEventListener('click', function() {
    showContent(uploadContent);
    hideContent(generalContent);
    hideContent(AiContent);
    hideContent(Text);
});

ai.addEventListener('click', function() {
    showContent(AiContent);
    hideContent(generalContent);
    hideContent(uploadContent);
    hideContent(Text);
});

rotate.addEventListener('click', function() {
    if (currentSide === 'front') {
        showContent(Back);
        hideContent(Front);
        currentSide = 'back';
    } else {
        showContent(Front);
        hideContent(Back);
        currentSide = 'front';
    }

    localStorage.setItem('currentSide', currentSide);
});

function hideContent(element) {
    element.style.display = 'none';
}

function showContent(element) {
    element.style.display = 'block';
}

let editingIndex = -1;
let isDragging = false;
let draggedElement = null;
```

```
    // -----------Text-----------


    document.getElementById('addTextButton').addEventListener('click',
addText);

    function addText() {
        const textInput = document.getElementById('text-input');
        const textList = document.getElementById('text-list');
        const fontSize = document.getElementById('font-size').value + 'px';
        const editButton = document.getElementById('addTextButton');

        editButton.innerHTML = '<i class="fas fa-plus"></i> Add text';

        const text = textInput.value.trim();

        if (text !== '') {
            const currentSide = getCurrentSide();
            const textContainer = document.getElementById('text-container');
            const textElement = document.createElement('div');
            textElement.innerHTML = `<span class="editable-text" style="font-
size: ${fontSize}; color: ${document.getElementById('font-color').value};
font-family: ${document.getElementById('font-select').value}; z-index:
6;">${text}</span>`;
            const listItem = document.createElement('li');
            listItem.className = 'text-list-item';
            textContainer.appendChild(textElement);
            listItem.innerHTML = `
                <span style="font-size: ${fontSize}; color:
${document.getElementById('font-color').value}; font-family:
${document.getElementById('font-select').value};">${text}</span>
                <button class="edit-button" onclick="editTextInList(this)"><i
class="fas fa-edit"></i></button>
                <button class="delete-button" onclick="deleteText(this)"><i
class="fas fa-trash-alt"></i></button>
            `;

            textList.appendChild(listItem);

            textElement.setAttribute('draggable', 'true');
            textElement.setAttribute('class', 'text-a');

            $('.editable-text').draggable();

            if (currentSide === 'front') {
                $('#front #text-container').append(textElement);
            } else {
```

```
                $('#back #text-container').append(textElement);
            }

            textInput.value = '';
        } else {
            alert('Please enter text before adding to the list.');
        }
    }

    window.editTextInList = function(button) {
        const listItem = button.parentNode;
        const textSpan = listItem.querySelector('span');
        const textInput = document.getElementById('text-input');
        const textContainer = document.getElementById('text-container');
        const editButton = document.getElementById('addTextButton');

        if (editButton) {
            editButton.innerHTML = '<i class="fas fa-edit"></i> Edit text';
        } else {
            console.error("Element with ID 'addTextButton' not found!");
        }

        const previousTextContent = textSpan.textContent;
        const previousFontFamily = textSpan.style.fontFamily;
        const previousFontSize = textSpan.style.fontSize;
        const previousFontColor = textSpan.style.color;

        textSpan.textContent = textInput.value;
        textSpan.style.fontFamily = document.getElementById('font-
select').value;
        textSpan.style.fontSize = document.getElementById('font-size').value +
'px';
        textSpan.style.color = document.getElementById('font-color').value;

        editingIndex =
Array.from(listItem.parentNode.children).indexOf(listItem);

        const editedElement =
textContainer.children[editingIndex].querySelector('span');
        editedElement.textContent = textInput.value;
        editedElement.style.fontFamily = textSpan.style.fontFamily;
        editedElement.style.fontSize = textSpan.style.fontSize;
        editedElement.style.color = textSpan.style.color;

        textInput.value = previousTextContent;
        document.getElementById('font-select').value = previousFontFamily;
        document.getElementById('font-size').value =
parseInt(previousFontSize);
```

```
        document.getElementById('font-color').value = previousFontColor;
        hideEmptyTextItems();
    }

    function hideEmptyTextItems() {
        const textList = document.getElementById('text-list');
        const listItems = textList.querySelectorAll('.text-list-item');

        listItems.forEach(item => {
            const textSpan = item.querySelector('span');
            if (textSpan.textContent.trim() === '') {
                item.style.display = 'none';
            } else {
                item.style.display = '';
            }
        });
    }

    window.deleteText = function(button) {
        const listItem = button.parentNode;
        const textList = document.getElementById('text-list');
        const textContainer = document.getElementById('text-container');

        listItem.remove();
        textContainer.innerHTML = "";

        for (let i = 0; i < textList.children.length; i++) {
            const listItem = textList.children[i];
            const textSpan = listItem.querySelector('span');

            const textElement = document.createElement('div');
            textElement.innerHTML = `<span class="editable-text" style="font-
size: ${textSpan.style.fontSize}; color: ${textSpan.style.color}; font-family:
${textSpan.style.fontFamily};">${textSpan.textContent}</span>`;
            textContainer.appendChild(textElement);
        }

        editingIndex = -1;
    }

    function getCurrentSide() {
        return currentSide;
    }

});

// --------Parādas virsraksta un apraksta inputi--------
```

```
var publishCheckbox = document.getElementById('publish-checkbox');
var additionalInfo = document.getElementById('additional-info');

publishCheckbox.addEventListener('change', function() {

    if (this.checked) {
        additionalInfo.style.display = 'block';
    } else {
        additionalInfo.style.display = 'none';
    }
});

// --------saglaba datubaze--------
function saveImage(side, callback) {
    var productDiv = document.querySelector('.product');
    var parentWidth = productDiv.offsetWidth;
    var parentHeight = productDiv.offsetHeight;

    productDiv.style.width = parentWidth + 'px';
    productDiv.style.height = parentHeight + 'px';

    var canvas = document.createElement('canvas');
    var context = canvas.getContext('2d');

    canvas.width = parentWidth;
    canvas.height = parentHeight;

    html2canvas(productDiv).then(function (renderedCanvas) {
        context.drawImage(renderedCanvas, 0, 0, parentWidth, parentHeight);

        var base64URL = canvas.toDataURL('image/png');
        callback(side, base64URL);
    });
}

$('#buy-button').click(function() {
    var publishCheckbox = $('#publish-checkbox').is(":checked");
    var numValue = $('.num').text();
    var activeSize = $('.size-option.active').attr('data-value');
    var activeColor = $('.color-select.active-color').attr('data-color-name');
    var productSlug = $('#product-slug').val();
    var title = $('#title-input').val();
    var description = $('#description-input').val();

    var errorHtml = '';

    if (!activeColor) {
        errorHtml += '<p>' + 'Please select a color' + '</p>';
```

```
}

if (!activeSize && $('.size-option').length > 0) {
    errorHtml += '<p>' + 'Please select a size' + '</p>';
}

if (publishCheckbox) {
    if (!title.trim()) {
        errorHtml += '<p>' +  'Title is required' + '</p>';
    }

    if (!description.trim()) {
        errorHtml += '<p>' + 'Description is required' + '</p>';
    }
}
if (errorHtml) {
    $('#error-messages').html(errorHtml).addClass('show');
    setTimeout(function() {
        $('#error-messages').removeClass('show');
    }, 10000);
    return;
}

var texts = [];
$('#Text #text-list .text-list-item').each(function() {
    var text = $(this).find('span').text().trim();
    if (text !== '') {
        var fontSize = $(this).find('span').css('font-size');
        var fontColor = $(this).find('span').css('color');
        var fontFamily = $(this).find('span').css('font-family');

        texts.push({
            'text': text,
            'font_size': fontSize,
            'text_color': fontColor,
            'font_family': fontFamily
        });
    }
});

var images = [];
$('#save-img img').each(function() {
    var imageData = $(this).attr('src');
    images.push(imageData);
});

var formData = new FormData();
formData.append('publish_product', publishCheckbox);
```

```
    formData.append('num_value', numValue);
    formData.append('product_color', activeColor);
    formData.append('product_size', activeSize);
    formData.append('product_slug', productSlug);
    formData.append('product_title', title);
    formData.append('product_description', description);
    formData.append('images', JSON.stringify(images));
    formData.append('texts', JSON.stringify(texts));

    $('#front').css('display', 'block');
    $('#back').css('display', 'none');

    saveImage('front', function(side, base64URLFront) {
        formData.append(side + '_image', base64URLFront);
        $('#front').css('display', 'none');
        $('#back').css('display', 'block');
        saveImage('back', function(side, base64URLBack) {
            formData.append(side + '_image', base64URLBack);

            var csrfToken = $("input[name='csrfmiddlewaretoken']").val();
            formData.append('csrfmiddlewaretoken', csrfToken);

            $.ajax({
                type: 'POST',
                url: '/save_order/',
                data: formData,
                contentType: false,
                processData: false,
                success: function(response) {
                    var orderId = response.order_id;
                    AddToCart(orderId)
                    displaySuccessMessage('Your order has been successfully
saved!');
                },
                error: function(xhr, status, error) {
                    console.error(error);
                }
            });
        });
    });
});

function AddToCart(order_id) {
    var formData = new FormData();
    formData.append('product_id', order_id);

    $.ajax({
        type: 'POST',
```

```
        url: '/cart/add/' + order_id + '/',
        data: formData,
        contentType: false,
        processData: false,
        beforeSend: function(xhr, settings) {
            xhr.setRequestHeader("X-CSRFToken", getCookie('csrftoken'));
        },
        success: function(response) {
            console.log(response);
            displaySuccessMessage('Product added to cart successfully!');
            $(document).ready(function() {
                var cartCountElement = $('#cart-count');
                if (cartCountElement.length === 0) {
                    var newCartCountElement = $('<span id="cart-
count"></span>');
                    newCartCountElement.text(response.cart_count);
                    $('#cart').append(newCartCountElement);
                } else {
                    cartCountElement.text(response.cart_count);
                }
            });
        },
        error: function(xhr, status, error) {
            console.error(error);
        }
    });
}

function getCookie(name) {
    var cookieValue = null;
    if (document.cookie && document.cookie !== '') {
        var cookies = document.cookie.split(';');
        for (var i = 0; i < cookies.length; i++) {
            var cookie = cookies[i].trim();
            if (cookie.substring(0, name.length + 1) === (name + '=')) {
                cookieValue = decodeURIComponent(cookie.substring(name.length
+ 1));
                break;
            }
        }
    }
    return cookieValue;
}

function displaySuccessMessage(message) {
    $('#success-message').text(message);
    $('#success-message').fadeIn();
```

```
    setTimeout(function() {
        $('#success-message').fadeOut();
    }, 5000);
}

// --------Ai-generator--------
const token = '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';
const InputTxt = document.getElementById('textInput-Ai');
const image = document.getElementById('image');
const button = document.getElementById('generateButton');
const loader = document.getElementById('loader');

async function query() {
    loader.style.display = 'block';

    const response = await fetch(
        "https://api-inference.huggingface.co/models/ZB-Tech/Text-to-Image",
        {
            headers: { Authorization: `Bearer ${token}` },
            method: "POST",
            body: JSON.stringify({'inputs': InputTxt.value}),
        }
    );
    const result = await response.blob();

    loader.style.display = 'none';

    image.onload = function() {
        document.getElementById('sendToImageBtn').style.display = 'flex';
    };

    image.src = URL.createObjectURL(result);
}

button.addEventListener('click', async function(){
    document.getElementById('sendToImageBtn').style.display = 'none';
    query();
});

document.getElementById('sendToImageBtn').addEventListener('click', function()
{
    const imageContainer = document.getElementById('image');
    const uploadImgContainer = document.querySelector('.upload-img');
    const imageId = Date.now();

    let selectedContainer;
    const frontContainer = document.getElementById('front');
    const backContainer = document.getElementById('back');
```

```
    if (frontContainer.style.display === 'block') {
        selectedContainer = frontContainer;
    } else {
        selectedContainer = backContainer;
    }

    if (selectedContainer) {
        const canvas = document.createElement('canvas');
        const ctx = canvas.getContext('2d');
        canvas.width = imageContainer.width;
        canvas.height = imageContainer.height;
        ctx.drawImage(imageContainer, 0, 0, canvas.width, canvas.height);
        const base64data = canvas.toDataURL();

        const sideId = selectedContainer.id;

        const newImg = document.createElement('img');
        newImg.className = 'uploaded-img element-image resizable-image';
        newImg.src = base64data;


        const htmlImage = `
            <div class='uploaded-img element-image ${sideId}' data-image-
id='${imageId}'>
                <img src='${base64data}' class='editable-image resizable-
image' draggable='true'>
            </div>
        `;

        selectedContainer.insertAdjacentHTML('afterbegin', htmlImage);
    } else {
        const errorMessage = document.getElementById('error-messages');
        errorMessage.innerText = 'Nevarēja noteikt aktīvo pusi. Lūdzu,
izvēlieties aktīvo pusi, kurai pievienot attēlu.';
        errorMessage.style.display = 'block';
    }

    const canvas2 = document.createElement('canvas');
    const ctx2 = canvas2.getContext('2d');
    canvas2.width = imageContainer.width;
    canvas2.height = imageContainer.height;
    ctx2.drawImage(imageContainer, 0, 0, canvas2.width, canvas2.height);
    const base64data2 = canvas2.toDataURL();

    const newImg2 = document.createElement('img');
    newImg2.className = 'uploaded-img front';
    newImg2.src = base64data2;
```

```
    const newDiv2 = document.createElement('div');
    newDiv2.className = 'uploaded-image-container'; // pievienots, lai
atvieglotu identifikāciju
    newDiv2.id = 'save-img';
    newDiv2.setAttribute('data-image-id', imageId)

    const removeBtn2 = document.createElement('button');
    removeBtn2.type = 'button';
    removeBtn2.className = 'remove-btn';
    removeBtn2.innerHTML = '<i class="fas fa-times" aria-hidden="true"></i>';
    removeBtn2.addEventListener('click', function() {
        let imageIdToRemove = $(this).parent().data('image-id');
        console.log(imageIdToRemove)

        $('[data-image-id="' + imageIdToRemove + '"]').remove();

        let listItemToRemove = $('.text-list-item').eq(imageIdToRemove);
        listItemToRemove.remove();
    });
    newDiv2.appendChild(newImg2);
    newDiv2.appendChild(removeBtn2);

    uploadImgContainer.appendChild(newDiv2);
});
```

Urls.py:

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('i18n/', include('django.conf.urls.i18n')),
    path('', page.homepage, name='homepage'),
    path('login/', page.login_view, name='login'),
    path('register/', page.register, name='register'),
    path('logout/', page.logout_view, name='logout'),
    path('reset_password/', auth_views.PasswordResetView.as_view(),
name='password_reset'),
    path('reset_password_sent/', auth_views.PasswordResetDoneView.as_view(),
name='password_reset_done'),
    path('reset/<uidb64>/<token>/',
auth_views.PasswordResetConfirmView.as_view(), name='password_reset_confirm'),
    path('reset_password_complete/',
auth_views.PasswordResetCompleteView.as_view(),
name='password_reset_complete'),
    path('contact-us/', page.contact_us, name='contact_us'),
    path('creative-corner/', page.creativecorner, name='creativecorner'),
    path('creative-corner/<str:user>/<str:product_title>/<int:order_id>/',
page.detail, name='detail'),
    path('save-rating/', page.save_rating, name='save_rating'),
    path('cart/', page.cart, name='cart'),
    path('cart/add/<int:id>/', page.cart_add, name='cart_add'),
    path('cart/item_clear/<int:id>/', page.item_clear, name='item_clear'),
    path('cart/item_increment/<int:id>/',
         page.item_increment, name='item_increment'),
    path('cart/item_decrement/<int:id>/',
         page.item_decrement, name='item_decrement'),
    path('cart/cart_clear/', page.cart_clear, name='cart_clear'),
    path('cart/cart-detail/',page.cart_detail,name='cart_detail'),
    path('check_discount_code/',page.check_discount_code,
name='check_discount_code'),
    path('account/',page.account, name='account'),
    path('save_user_data/', page.save_user_data, name='save_user_data'),
    path('account/change_password/', page.change_password,
name='change_password'),
    path('account/delete_account/', page.delete_profile,
name='delete_profile'),
    path('design/<slug:slug>/', page.design, name='design_detail'),
    path('accounts/', include('allauth.urls')),
    path('accounts/', include('allauth.socialaccount.urls')),
    path('save_order/', page.save_order, name='save_order'),
```

```
    path('<str:slug_url>/create-checkout-session/',
         stripe.create_checkout_session),
    path('config/', stripe.stripe_config),
    path('webhook', stripe.stripe_webhook),
    path('create-checkout-session/', stripe.create_checkout_session),
    path('success/', stripe.SuccessView.as_view(), name='success'),
    path('cancelled/', stripe.CancelledView.as_view(), name='cancel'),


]

urlpatterns += i18n_patterns(
    path('', page.homepage, name='homepage'),
    path('admin/', admin.site.urls),
    path('login/', page.login_view, name='login'),
    path('register/', page.register, name='register'),
    path('reset_password/', auth_views.PasswordResetView.as_view(),
name='password_reset'),
    path('reset_password_sent/', auth_views.PasswordResetDoneView.as_view(),
name='password_reset_done'),
    path('reset/<uidb64>/<token>/',
auth_views.PasswordResetConfirmView.as_view(), name='password_reset_confirm'),
    path('reset_password_complete/',
auth_views.PasswordResetCompleteView.as_view(),
name='password_reset_complete'),
    path('contact-us/', page.contact_us, name='contact_us'),
    path('cart/', page.cart, name='cart'),
    path('creative-corner/<str:user>/<int:product_list_id>/', page.detail,
name='detail'),
    path('account/',page.account, name='account'),
    path('save_user_data/', page.save_user_data, name='save_user_data'),
    path('account/change_password/', page.change_password,
name='change_password'),
    path('design/<slug:slug>/', page.design, name='design_detail'),
    path('creative-corner/', page.creativecorner, name='creativecorner'),
    path('creative-corner/<str:user>/<str:product_title>/<int:order_id>/',
page.detail, name='detail'),
)

handler404 = page.handler404
handler500 = page.handler500


if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

Projektaa iestatījumi:

```
"""
Django settings for ecomerce project.

Generated by 'django-admin startproject' using Django 4.1.2.

For more information on this file, see
https://docs.djangoproject.com/en/4.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.1/ref/settings/
"""

from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-v1fiaup#2ni$7o)+u(uzkc9n$((#nrayt9__52wtm#651%-
tpj'
RECAPTCHA_PROXY = {'http': 'http://127.0.0.1:8000', 'https':
'https://127.0.0.1:8000'}
RECAPTCHA_REQUIRED_SCORE = 0.85

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = False

ALLOWED_HOSTS = ['www.ericprint.com', '104.248.195.146', 'ericprint.com',
'https://ericprint.com']

CSRF_COOKIE_SECURE = False
CSRF_TRUSTED_ORIGINS = ['https://ericprint.com', 'https://www.ericprint.com']

# Application definition

INSTALLED_APPS = [
    'modeltranslation',
    'django_recaptcha',
    'jazzmin',
```

```
    'cart',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'colorfield',
    'django.contrib.staticfiles',
    'ckeditor',
    "phonenumber_field",
    'ckeditor_uploader',
    'home.apps.HomeConfig',
    'stripe_integration',
    'chartjs',
    'django_extensions',
    'allauth',
    'allauth.account',
    'allauth.socialaccount',
    'allauth.socialaccount.providers.google',
    'allauth.socialaccount.providers.facebook',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.locale.LocaleMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'allauth.account.middleware.AccountMiddleware',
]

ROOT_URLCONF = 'ecomerce.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'cart.context_processor.cart_total_amount',
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
```

```
            ],
        },
    },
]

WSGI_APPLICATION = 'ecomerce.wsgi.application'


DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'urban_prod',
        'USER': 'u_urban',
        'PASSWORD': '123',
        'HOST': 'localhost',
        'PORT': '',
    }
}

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

abspath = lambda *p: os.path.abspath(os.path.join(*p))

PROJECT_ROOT = abspath(os.path.dirname(__file__))

MEDIA_ROOT = abspath(PROJECT_ROOT, 'media')
MEDIA_URL = '/media/'

AUTH_USER_MODEL = 'home.user'

# Internationalization
# https://docs.djangoproject.com/en/4.1/topics/i18n/
```

```
LANGUAGE_CODE = 'lv-lv'

TIME_ZONE = 'Etc/GMT+2'

USE_I18N = True

USE_TZ = True

gettext = lambda s: s
LANGUAGES = (
    ('lv', gettext('Latvian')),
    ('en', gettext('English')),
)

MODELTRANSLATION_DEFAULT_LANGUAGE = 'lv'
MODELTRANSLATION_LANGUAGES = ('lv', 'en')

LOCALE_PATHS = [
    os.path.join(BASE_DIR, 'locale'),
]

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.1/howto/static-files/

STATIC_URL = '/static/'

STATIC_ROOT = '/home/urban/Printing/static/'

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'


JAZZMIN_SETTINGS = {
    "language_chooser": True
}

EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_HOST_USER = 'mafiagameeee@gmail.com'
EMAIL_HOST_PASSWORD = 'hjmyxkzttzahgfib'
EMAIL_USE_TLS = True
DEFAULT_FROM_EMAIL = 'mafiagameeee@gmail.com'

try:
    from ecomerce.local_settings import *
except ImportError:
    pass
```

```
CKEDITOR_UPLOAD_PATH = "uploads/"

CKEDITOR_CONFIGS = {
    'default': {
        'height': 300,
        'width': 600,
    'toolbar': 'Custom',
        'toolbar_Custom': [
            ['Link', 'Unlink', 'Anchor'],
            ['Image', 'Flash', 'Table', 'HorizontalRule'],
            ['TextColor', 'BGColor'],
            ['Smiley', 'SpecialChar'], ['Source'],
            ['JustifyLeft','JustifyCenter','JustifyRight','JustifyBlock'],
            ['NumberedList','BulletedList'],
            ['Indent','Outdent'],
            ['Maximize'],
            ['Styles', 'Format', 'Font', 'FontSize'],
        ]
    },
}

JAZZMIN_SETTINGS = {
    "site_title": "Design",
    "site_header": "Design",
    "welcome_sign": "Welcome",
    "show_sidebar": True,
    "language_chooser": True,
    "language": "lv",
    "navigation_expanded": True,
    "hide_apps": ["'sites'"],
    "show_ui_builder": False,
    "related_modal_active": True,
    "topmenu_links": [],
    "usermenu_links": [],
    "theme": "home",
    "icons": {
        "app.Model": "fas fa-icon",
        "auth.User": "fas fa-user",
        "home.Product": "fas fa-shopping-bag",
        "home.CustomDesign": "fas fa-cogs",
        "home.ContactMessage": "fas fa-envelope",
        "home.Contact": "fas fa-address-book",
        "home.user": "fas fa-user",
        "home.Price": "fas fa-dollar-sign",
        "home.Product_list": "fas fa-list",
        "home.GiftCode": "fas fa-gift",
        "home.Color": "fas fa-paint-brush",
        "home.Size": "fas fa-ruler",
```

```
        "home.Order": "fas fa-shopping-cart",
        "home.Purchase": "fas fa-receipt",
        "stripe_integration.StripeKeys": "fas fa-key",
        "django_recaptcha.RecaptchaKeys": "fas fa-key"
    }
}

CART_SESSION_ID = 'cart'

SOCIALACCOUNT_LOGIN_ON_GET = True

AUTHENTICATION_BACKENDS = [
    'django.contrib.auth.backends.ModelBackend',
    'allauth.account.auth_backends.AuthenticationBackend',
]

LOGIN_REDIRECT_URL = '/'

SOCIALACCOUNT_AUTO_SIGNUP = True
SOCIALACCOUNT_EMAIL_REQUIRED = True
SOCIALACCOUNT_QUERY_EMAIL = True
```