JELGAVAS TEHNIKUMS

RALFS CVIKLINSKIS

Mājas lapas “NoRokas” prototipa izstrāde

kvalifikācijas darbs ieguvei kvalifikācijā - programmēšanas tehniķis

Darba izpildītājs: R.C. 05.06.2026 410. gr. Izgl. R. Cviklinskis

paraksts, datums

Jelgava 2026

ANOTĀCIJA

Ralfs C. *WEB* lapas izstrāde mantu pārdošanai: kvalifikācijas darbs. Jelgava: JT, 2026. 85 lpp., 39 att., 10 tab., 33 pielikumi.

Šis darbs risina problēmu, ka pērkot mantas un produktus no citiem cilvēkiem, nevar zināt to kvalitāti, piemēram, malku, dārzeņus u.c., tāpēc šajā saitē to problēmu atrisināt varēs ar iespēju komentēt zem produktiem un lasīt citu komentārus.

Mērķis ir izveidot *WEB* lapu kur var ielogoties, veidot rakstus, tos rediģēt, dzēst, redzēt citu lietotāju rakstus un zem tiem komentēt.

Darbs tiek izstrādāts individuāli.

ANNOTATION

Ralfs C. WEB page development for selling goods: qualification work. Jelgava: JT, 2026. 85 p., 39 pic., 10 tbl., 33 att.

This work solves the problem that, when buying goods and products from other people, you cannot know their quality, for example, firewood, vegetables, etc., so this site will solve this problem with the ability to comment under products and read other people's comments.

The goal is to create a WEB page where you can log in, create posts, edit, delete them, see other users posts and comment under them.

The work is developed individually.

SATURS

ATTĒLU SARAKSTS

TABULU SARAKSTS

IEVADS

Darba mērķis:

Izstrādāt prototipa WEB lapu, kas nodrošina platformu lietotājiem pašražotu preču, pakalpojumu, mantu publicēšanai un pārdošanai.

Darba uzdevumi:

1. Izveidot iespēju, kur lietotājs var pieteikties, reģistrēties un mainīt profila iestatījumus.
2. Izveidot iespēju lietotājam veidot rakstus, kas saturēs informāciju par to, ko lietotājs vēlas pārdot.
3. Izveidot iespēju lietotājam rediģēt un dzēst savus rakstus.
4. Izveidot mājas lapu, kur var redzēt visus rakstus kas ir veidoti.
5. Izveidot filtrēšanas iespēju.
6. Izveidot iespēju lietotājam komentēt visus rakstus.
7. Izveidot iespēju redzēt zem raksta komentārus.
8. Testēt programmu tā prototipa izstrādes laikā.

SAĪSINĀJUMI UN AKRONĪMI, TO SKAIDROJUMI

PHP	atvērtā pirmkoda skriptu valoda, kura sākotnēji bija paredzēta servera puses lietojumos dinamiska tīmekļa lapu ģenerēšanai[6].
JS	programmēšanas valoda un tīmekļa galvenā tehnoloģija tīmekļa lapas darbības nodrošināšanai klienta pusē[8].
HTML	standarta iezīmēšanas valoda dokumentiem, kas paredzēti rādīšanai tīmekļa pārlūkprogrammā. Tas nosaka tīmekļa satura saturu un struktūru[7].
CSS	stila lapas valoda, ko izmanto, lai norādītu dokumenta prezentāciju un stilu, kas rakstīts iezīmēšanas valodā, piemēram, HTML vai XML[9].
DOM	Dokumentu objektu modelis (*DOM*) savieno tīmekļa lapas ar skriptiem vai programmēšanas valodām, attēlojot dokumenta struktūru, piemēram, HTML, kas attēlo tīmekļa lapu, atmiņā.[2]
API	Lietojumprogrammu saskare (*API*) ir savienojums starp datoriem vai datorprogrammām. Tā ir programmatūras saskares veids, kas piedāvā pakalpojumu citām programmatūras daļām.[14]
PFP	Profila bilde saites lietotajam.

DOKUMENTĀCIJĀ IZMANTOTIE TERMINI UN TO SKAIDROJUMI

Post	raksts, attēls vai cits satura vienums, kas publicēts tiešsaistē, parasti emuārā vai sociālajos saziņas līdzekļos[10].
Tags	*tags* ir atslēgvārds vai termins, kas piešķirts kādai informācijai (piemēram, interneta grāmatzīmei, multividei, datu bāzes ierakstam vai datora failam)[11].

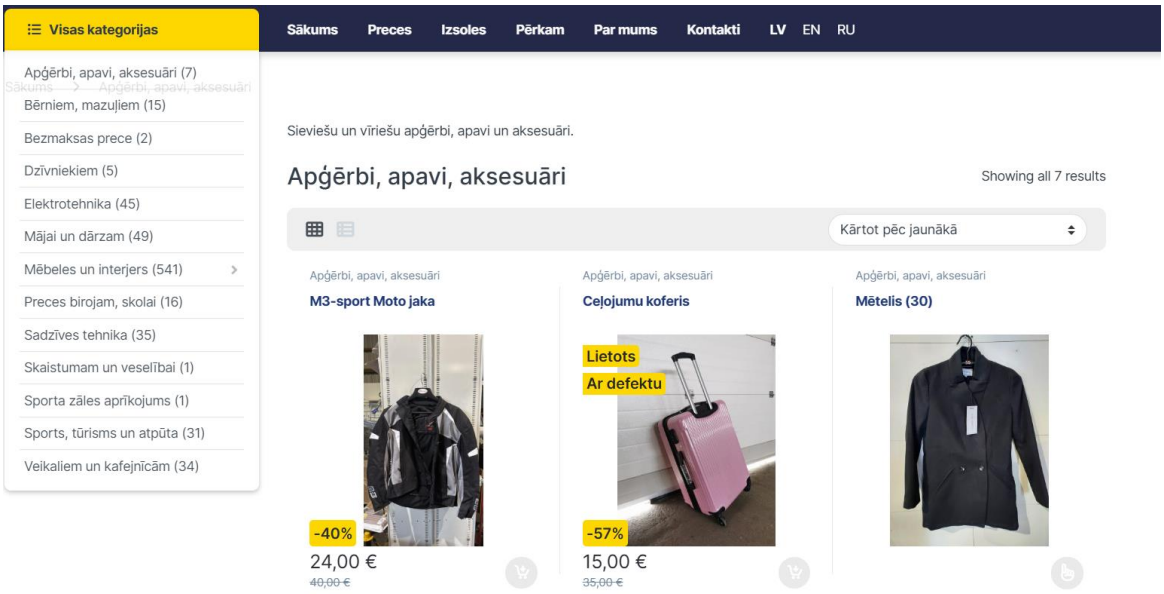
1. UZDEVUMA NOSTĀDNE

Manā projektā būs iespējams izveidot rakstus kur var ielikt bildes, nosaukumu, aprakstu un *tags*, šo rakstu būs iespējams redzēt citiem kas ir šajā saitē. Izmantojot visu *tags* sarakstu būs iespēja filtrēt visus rakstus. Būs iespēja ielogoties, kas dos iespēju rakstīt komentārus zem katra raksta, kurus arī būs iespējams redzēt visiem.

1.1. Līdzīgie produkti:

(skatīt 1. attēlu)Šajā saitē ir iespējams dot savas mantas, lietotas un jaunas, kuras tiks pārdotas lietotājiem, un daļā no cenas tiks dota tam kurš iedeva savu mantu. Komentēt nav iespējas, kas atņem iespēju sazināties ar citiem par mantas un saites kvalitāti.

- Utrupe[18]



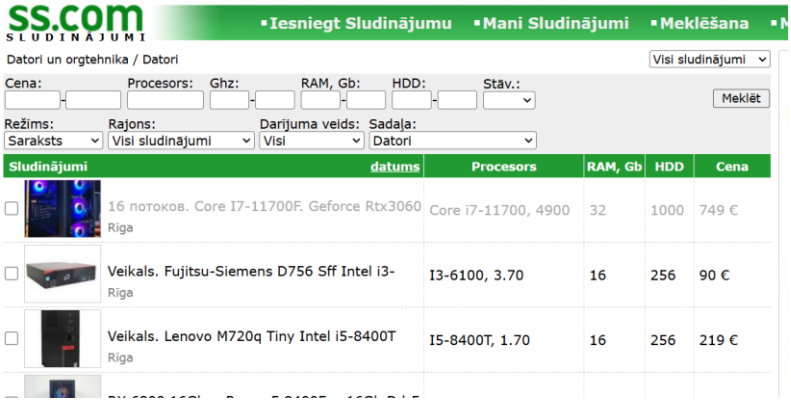
1.attēls. **Utrupe**

Kopīgs: var pārdot savas mantas citiem

Atšķirīgs: nav komentāri

(skatīt 2. attēlu)Šajā saitē jebkurš var pārdot ko vēlas, paši izvēlas cenu un mantas aprakstu, un paši arī piegādā mantu, vai arī tas kurš vēlas mantu dabūt pats to paņem. Šajā saitē nav komentāri.

- ss.com[17]



2.attēls. **ss.com**

Kopīgs: var pārdot savas mantas citiem ar pašizvēlētu cenu

Atšķirīgs: nav iespēja komentēt

(skatīt 3. attēlu)Šajā saitē var apskatīt lietotas mēbeles kuras var nopirkt viņu veikalā. Mēbeles tiek pārbaudītas uz kvalitāti, un visas problēmas tiek pieminētas.

- zviedru mebeles [19]

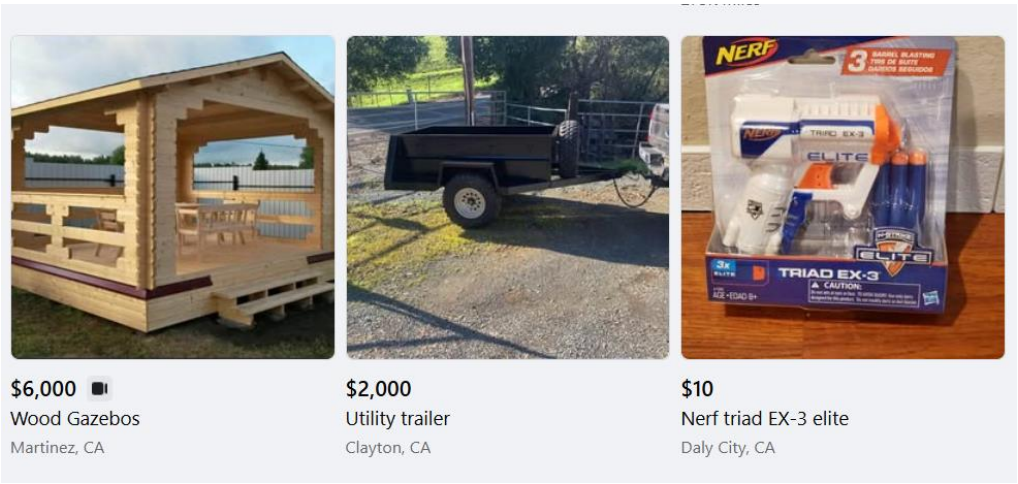
3.attēls. **Zviedrumebeles**

Kopīgs: var nopirkt lietotas mēbeles

Atšķirīgs: nav komentāri, pārdot var tikai uz vietas

(skatīt 4. attēlu)Šajā saitē var apskatīt mantas kas tiek pārdotas, no krēsliem līdz mašīnām. Katrs var pārdot mantas par pašizvēlētu cenu, un jebkurš to var nopirkt. Visas pirkšanas notiek starp lietotājiem.

- *Facebook marketplace* [3]



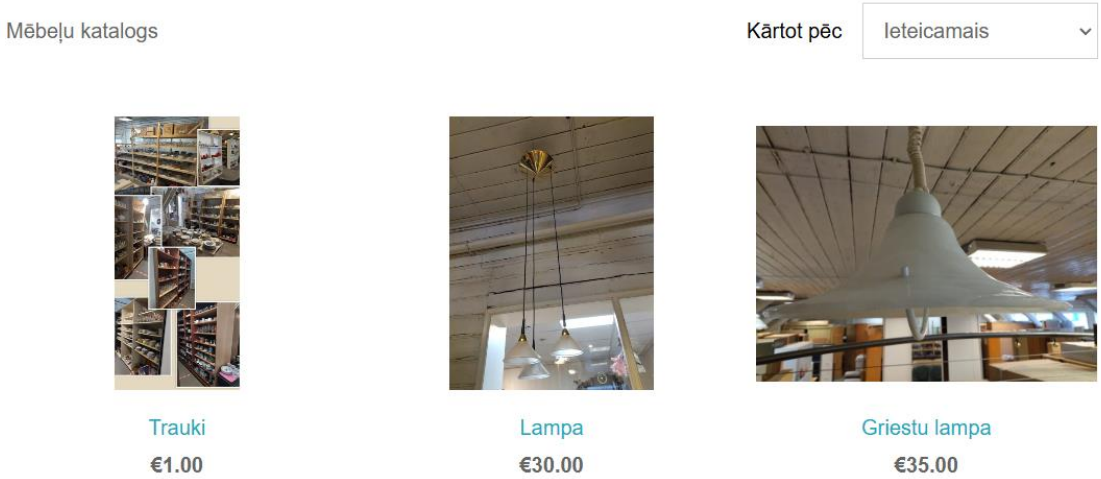
4.attēls. **Marketplace**

Kopīgs: var pārdot ko vēlas, izvēlas pats savu cenu

Atšķirīgs: nav iespējas komentēt.

(skatīt 5. attēlu)Šajā saitē ir iespējams nopirkt lietotas mēbeles, pa lielākam mēbeles kuras var izmantot mājā. Jebkurš var pārdot savas mēbeles ja tās nav salauztas.

- ptmebeles [16]



5.attēls. **Ptmebeles**

Kopīgs: var pārdot mājas mēbeles par savu cenu

Atšķirīgs: nav iespējams komentēt

1.2. Kopīgie secinājumi:

Pa lielākam visas saites kas pārdod lietotas mantas ir ļoti līdzīgas, lielāka atšķirība ir tā, ka dažreiz mantas pārdod pats lietotājs, dažreiz kompānija kas uztur saiti. Bet visbiežāk mantas pārdod pats lietotājs, un tās arī piegādā pircējam vai pircējs tās pats paņem. Galvenā problēma šajās saitēs ir tā, ka nevar komentēt, un pārdot pa lielākam var tikai mēbeles, paštaisītus produktus nevar, piemēram, malku, konservus u.c.

1.3. Mana projekta unikalitāte:

Manā projektā lietotājiem būs vairāk iespējas kontaktēties ar citiem lietotājiem, kas dos iespēju labāk izvēlēties mantu, un izvairīties no sliktām vai bojātām mantām, vēl būs iespēja pārdot sevis ražotus produktus, piemēram, malku un rotaslietas, kas dos vairāk cilvēkiem iespēju izmantot šo saiti.

1.4. Projekta mērķauditorija

Šīs projekts būs viss vairāk mērķēts uz tiem, kas paši veido un ražo produktus, piemēram, paštaisītas rotas lietas vai kurināmā malka, un kā ekstra lietotāji varēs pārdot lietotās mantas.

1.5. Gala rezultāta izskats:

Mana projekta galējais izskats varētu būt līdzīgs dažām no pieminētajām saitēm, bet ar vairāk funkcijām , piemēram, komentēšana kas palīdzēs lietotājam.

1.6. Projekta ierobežojumi:

1. Viss lielākais ierobežojums būs laiks, jo ar savām spējām šo projektu ātri nesanāks pabeigt, un visu laiku arī nevar iesaistīt šajā projektā, kas to tikai palēlinās.
2. Budžets nebūs problēma, jo šai saite vajadzētu tikai vienu nelielu datubāzi, kas daudz nemaksā, un pa lielākam būs viena cilvēka projekts.

2. PROGRAMMATŪRAS IZSTRĀDES VIDE

2.1. Teksta redaktors

2.1.1. *Visual Studio Code*:

Visual Studio Code ir bezmaksas, viegls, bet jaudīgs pirmkoda redaktors, kas darbojas gan datorā, gan tīmeklī un ir pieejams operētājsistēmām *Windows, macOS* un *Linux*. Tam ir iebūvēts *JS, TypeScript* un *Node.js* atbalsts, un tam ir bagātīga paplašinājumu ekosistēma citām programmēšanas valodām (piemēram, *C++, C#, Java, Python, PHP* un *Go*), izpildlaikiem (piemēram, *.NET* un *Unity*), vidēm (piemēram, *Docker* un *Kubernetes*) un mākoņiem (piemēram, *Amazon Web Services, Microsoft Azure* un *Google Cloud Platform*).[5]

Es izvēlējos šo teksta redaktoru, jo tas ir bez maksas, dod plašas iespējas programmēt, ir populārs un ērti izmantojams.

2.2. Datubāze un tās rīki

2.2.1. *MySQL*:

MySQL ir relāciju datubāzes pārvaldības sistēma (*RDBMS*), kas ir bezmaksas, atvērtā koda un izmanto dažādas patentētas licences, tostarp *GNU* vispārējo publisko licenci (*GPL*). Kā *RDBMS, MySQL* izmanto *SQL*, lai pārvaldītu datus datubāzē. Tā organizē korelētus datus vienā vai vairākās datu tabulās, un šī korelācija palīdz strukturēt datus.[13]

Es izvēlējos *MySQL* savai datubāzei, jo šī sistēma man ir visvairāk pazīstama, tā ir bezmaksas un atbalsta daudzas sistēmas.

2.2.2. *HeidiSQL*:

HeidiSQL ir bezmaksas un atvērtā koda programmatūra cilvēkiem, kas strādā ar datubāzēm, un tās mērķis ir būt intuitīvi lietojamai. *HeidiSQL* ļauj izveidot savienojumu ar dažādām datubāzes sistēmām, ieskaitot *MySQL*. [4]

Es izvelējos *HeidiSQL*, jo sen jau izmantoju, man pašam labi zināms, un tas strādā ar *MySQL* relāciju datubāzes pārvaldības sistēmu.

2.3. Programmēšanas valodas

2.3.1. *HTML*:

Hiperteksta iezīmēšanas valoda (*HTML*) ir standarta iezīmēšanas valoda dokumentiem, kas paredzēti attēlošanai tīmekļa pārlūkprogrammā. Tā nosaka tīmekļa satura saturu un struktūru. Tīmekļa pārlūkprogrammas saņem *HTML* dokumentus no tīmekļa servera vai lokālās

krātuves un atveido šos dokumentus multimediju tīmekļa lapās. *HTML* semantiski apraksta tīmekļa lapas struktūru un sākotnēji ietvēra norādes tās izskatam.[7]

Es izvēlējos *HTML*, jo tā ir standarta tīmekļa lapu iezīmēšanas valoda, kas no nozīmē to, ka to atbalsta gandrīz katra sistēma un programma, ka nodrošina tās strādāšanu uz jebkuras sistēmas.

2.3.2. *JS*:

JS(JavaScript) ir augsta līmeņa, bieži vien tieši laikā kompilēta valoda, kas atbilst *ECMAScript* standartam. Tai ir dinamiska tipizēšana, uz prototipiem balstīta objekt orientācija un pirmās klases funkcijas. Tā ir daudz paradigma, kas atbalsta notikumu vadītus, funkcionālus un imperatīvus programmēšanas stilus. Tai ir lietojumprogrammu programmēšanas saskarnes (*API*) darbam ar tekstu, datumiem, regulārām izteiksmēm, standarta datu struktūrām un dokumentu objektu modeli (*DOM*). Tīmekļa pārlūkprogrammām ir īpaša *JavaScript* programma, kas izpilda klienta kodu.[12]

JS tika izvēlēts, jo tas ir cieši saistīts ar tīmekļa lapām, kas samazina risku par programmas nestrādāšanu un dod daudz kontroli pār lapām. *JS* vēl ir pieejamas daudz bibliotēkas, kas var dod vēl vairāk kontroli pār lapu un iespēju veidot ļoti īpašas darbības kas atšķiras no citām tīmekļa lapām

2.3.3. CSS:

Kaskādes stila lapas (*CSS*) ir stila lapu valoda, ko izmanto, lai norādītu dokumenta, kas rakstīts iezīmēšanas valodā, piemēram, *HTML* vai *XML*, noformējumu un stilu. *CSS* ir globālā tīmekļa stūrakmens tehnoloģija līdzās *HTML* un *JavaScript*.[15]

Izmantoju *CSS*, jo tas ir viens no labākajiem veidiem, kā uztaisīt stilu *HTML*, un tas arī ir izmantojams jebkurā sistēmā.

2.3.4. *PHP*:

Populāra vispārējas nozīmes skriptvaloda, kas ir īpaši piemērota tīmekļa izstrādei. Ātra, elastīga un pragmatiska. PHP kodu tīmekļa serverī parasti apstrādā PHP interpretētājs, kas ieviests kā modulis vai *Common Gateway Interface* (*CGI*) izpildāmais fails.[6]

Izvēlējos *PHP*, jo tas ļauj vieglāk savienot saiti ar datubāzi, un tas arī satur rīkus lai veidotu saiti.

2.4. Ietvari

2.4.1. *CodeIgniter*

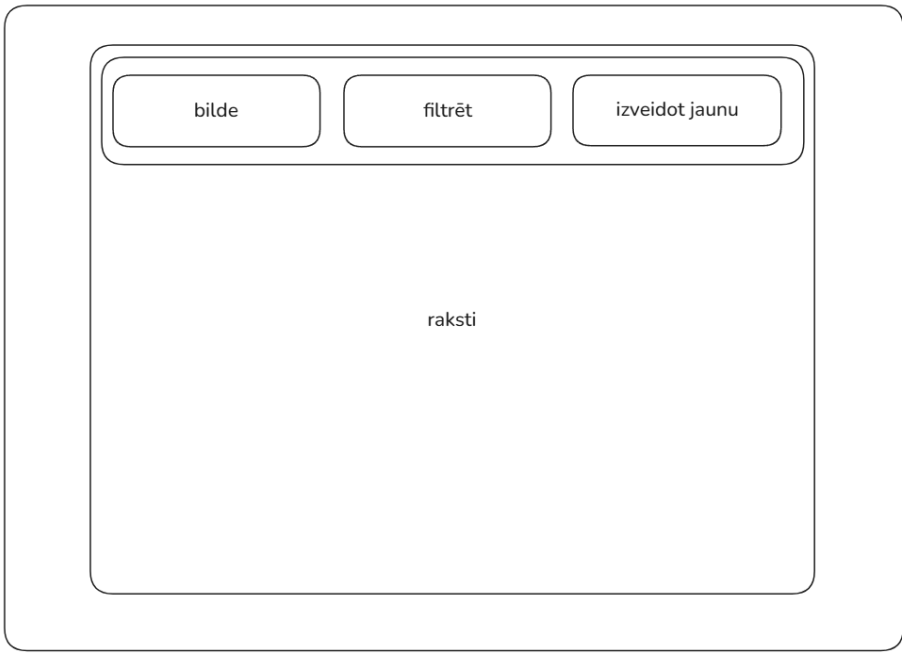
Tas ir lietojumprogrammu izstrādes ietvars — rīku komplekts — cilvēkiem, kas veido tīmekļa vietnes. Tā mērķis ir nodrošināt daudz ātrāku projektu izstrādi nekā rakstot kodu no nulles, nodrošinot bagātīgu bibliotēku komplektu bieži nepieciešamajiem uzdevumiem, kā arī vienkāršu saskarni un loģisku struktūru piekļuvei šīm bibliotēkām. [1]

Es izvēlējos *CodeIgniter* šim projektam, jo tas atvieglo tīmekļa lapu veidošanu, samazinot vajadzīgo laiku lai to veidotu, nodrošina lielāku drošību, ātrumu un autentifikāciju.

3. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

3.1. Sākotnēja ideja

Pirmajā prototipā man pa lielākam bija viena lapa, kurā var izveidot, redzēt un filtrēt savus rakstus.



6.attēls. **Sākotnējā struktūrskice**

3.2. Nefunkcionālās prasības

3.2.1. Navigācijas josla:

Logo – (skatīt 7. attēlu) šeit būs redzama mājas lapas galvenā bilde

Uz sākumu – (skatīt 7. attēlu) poga uz sākuma lapu

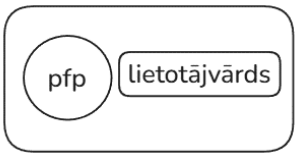
Par mums – (skatīt 7. attēlu) poga uz lapu "par mums"

Profils – (skatīt 7. attēlu) logs kas rāda informāciju par kontu (skatīt 8. attēlu)

7.attēls. **Navigācijas joslas struktūrskice**

3.2.2. Profila logs

Profils – (skatīt 8. attēlu) logā būs redzama lietotāja profila bilde un lietotājvārds



8.attēls. **Profila loga struktūrskice**

3.2.3. Mājas lapa:

Poga filtrēt – atradīsies rakstu loga augšējā kreisajā pusē(skatīt 9. attēlu).

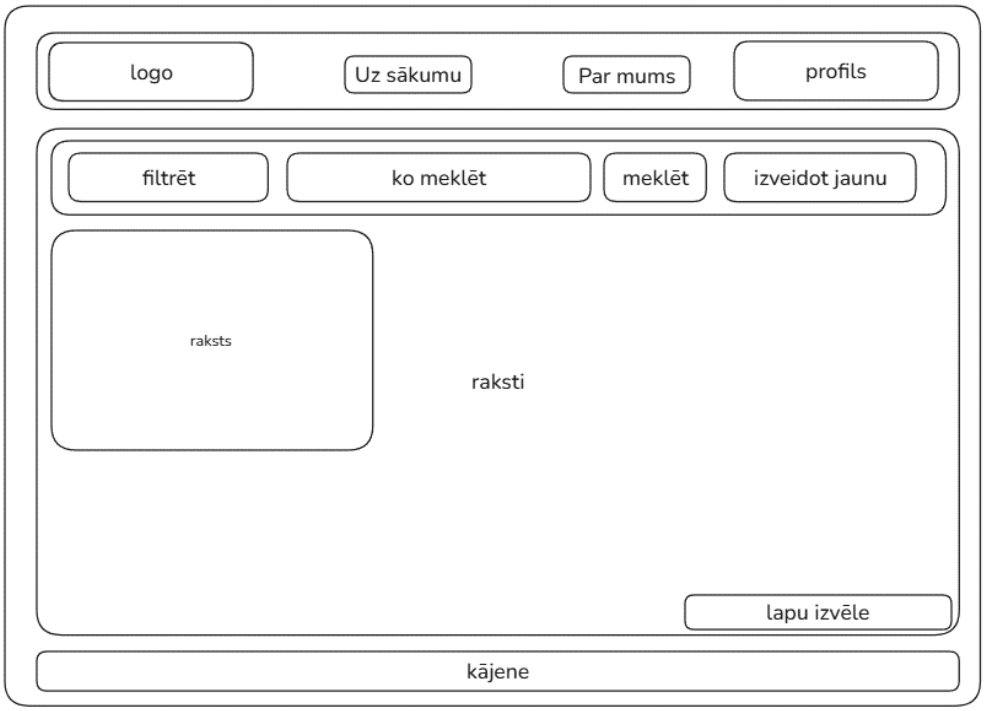
Poga izveidot jaunu – atradīsies rakstu loga augšējā labajā pusē (skatīt 9. attēlu).

Logs ko meklēt – atradīsies rakstu loga augšā(skatīt 9. attēlu).

Logs raksti – atradīsies mājas lapas vidū (skatīt 9. attēlu), zem navigācijas joslas un šajā logā būs redzami visi raksti, kurus lietotāji ir izveidojuši

Lapu izvēlne – (skatīt 9. attēlu) saturēs pogas (skatīt 10. attēls. **Mājas lapu izvēlne** attēlu) , ar kurām var nomainīt mājas lapas lapu

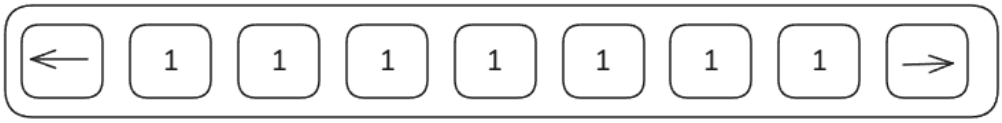
Kājene – (skatīt 9. attēlu) atradīsies pašā lapas apakšpusē, saturēs kontakt informāciju.



9.attēls. **Mājas lapas struktūrskice**

3.2.4. Mājas lapu izvēlne

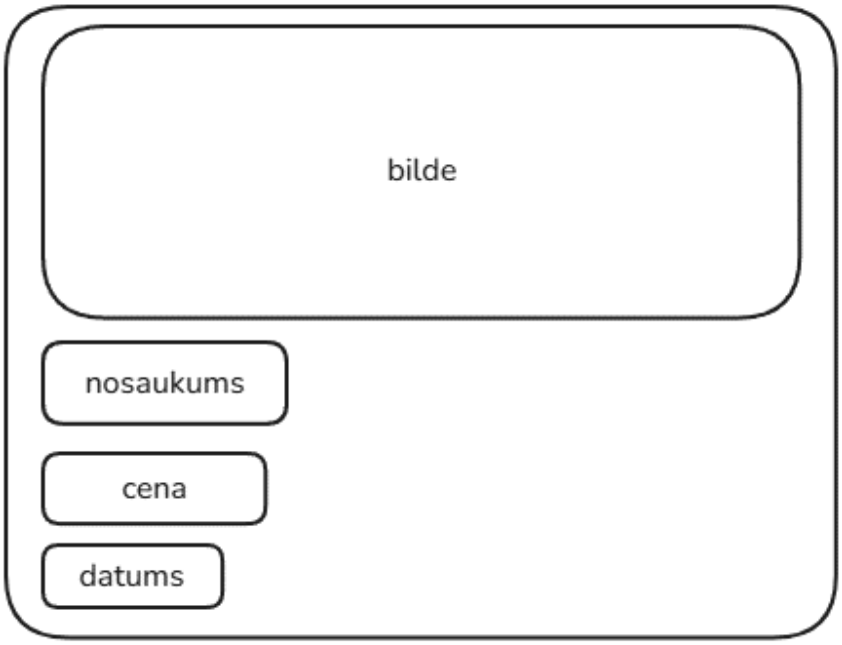
Satur pogas ar kuram var nomainīt rakstus kurus rāda, katra lapa satur 12 rakstus kurus rāda.



10. attēls. **Mājas lapu izvēlnes struktūrskice**

3.2.5. Raksta logs:

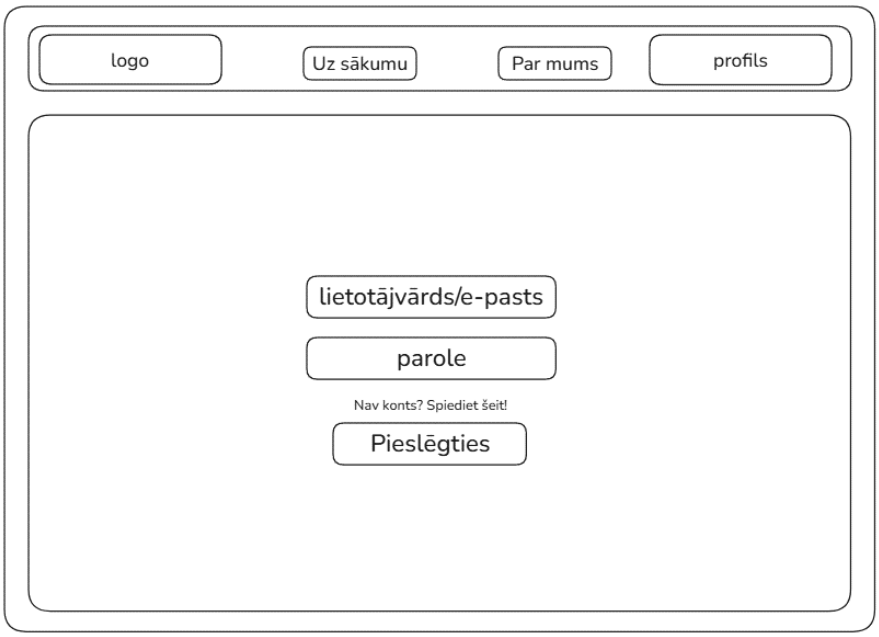
Uz raksta loga (skatīt 11. attēlu) būs redzama bilde, nosaukums, apraksts, datums kad izveidoja. Visi raksti būs sakārtoti viens aiz otra laukā “raksti” (skatīt 9. attēlu).



11.attēls. **Raksta struktūrskice**

3.2.6. Ielogošanās lapa:

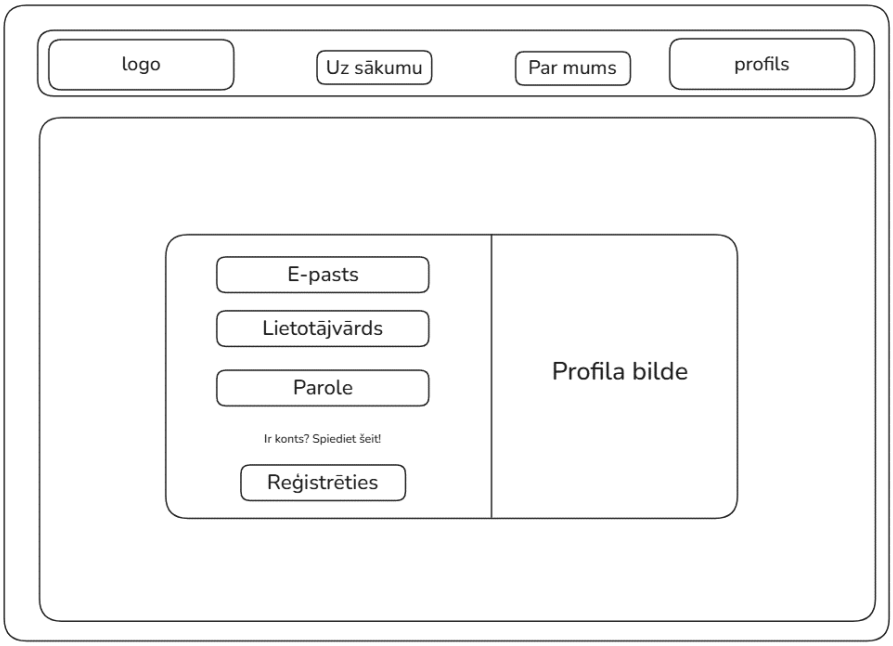
Ielogošanās lapā (skatīt 12. attēlu) būs e-pasta/lietotājvārda un paroles ievada lauki, ielogošanās poga, kā arī poga uz konta veidošanas lapu(skatīt 13.attēls. **Konta izveidošanas loga** attēlu).



12.attēls. **Ielogošanās lapas struktūrskice**

3.2.7. Konta izveidošanas lapa:

Konta izveidošanas lapā (skatīt 13. attēlu) būs e-pasta, lietotājvārda un paroles ievada lauki, Reģistrēšanās lapas poga, kā arī poga uz ielogošanās lapu(skatīt 12. attēlu).

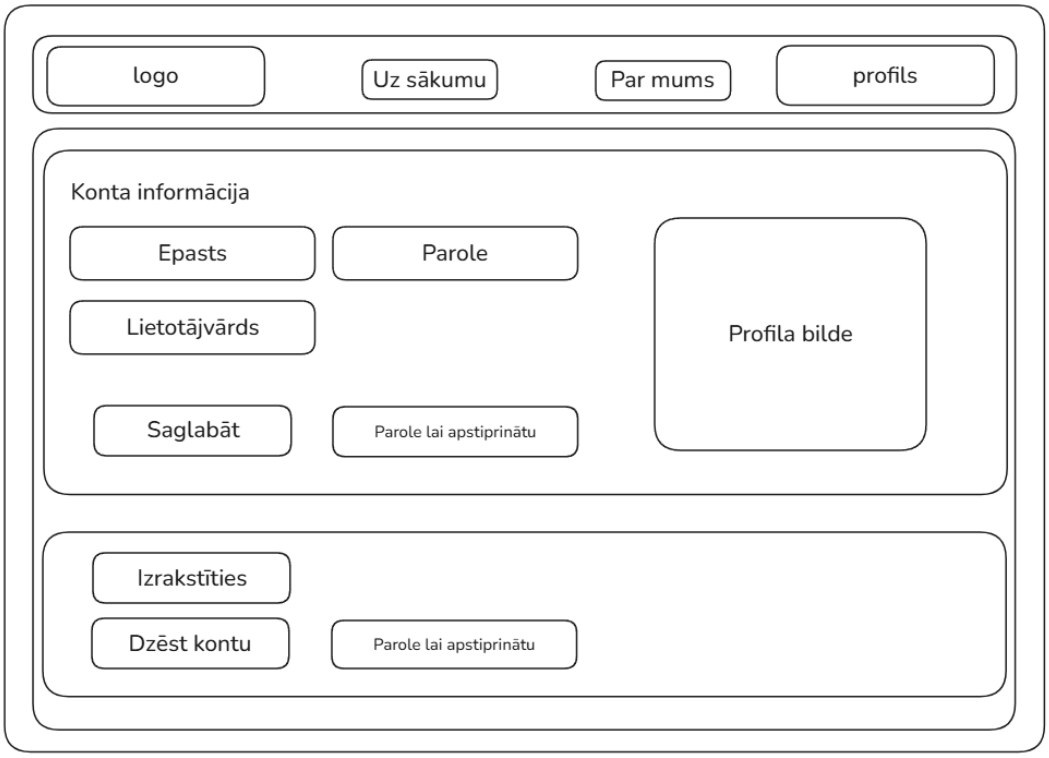


13.attēls. **Konta izveidošanas loga struktūrskice**

3.2.8. Lietotāja konta lapa:

Lietotāja konta lapas (skatīt 14. attēlu) augšējais logs saturēs e-pasta, lietotājvārda un paroles laukus, kur var ierakstīt, ja vēlas to mainīt, profila bildes lauks, kur var redzēt un nomainīt sava profila bildi, kā arī apstiprināšanas lauks, kur ievada savu tagadējo paroli, lai apstiprinātu izmaiņas nospiežot pogu "saglabāt".

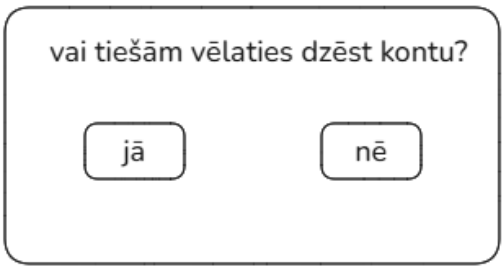
Apakšējais logs satur pogas "izrakstīties" un "dzēst kontu", kā arī paroles lauku, kur ievada tagadējo paroli lai apstiprinātu konta dzēšanu.



14. attēls. **Lietotāja konta lapas struktūrskice**

3.2.9. Konta izdzēšanas apstiprināšanas logs:

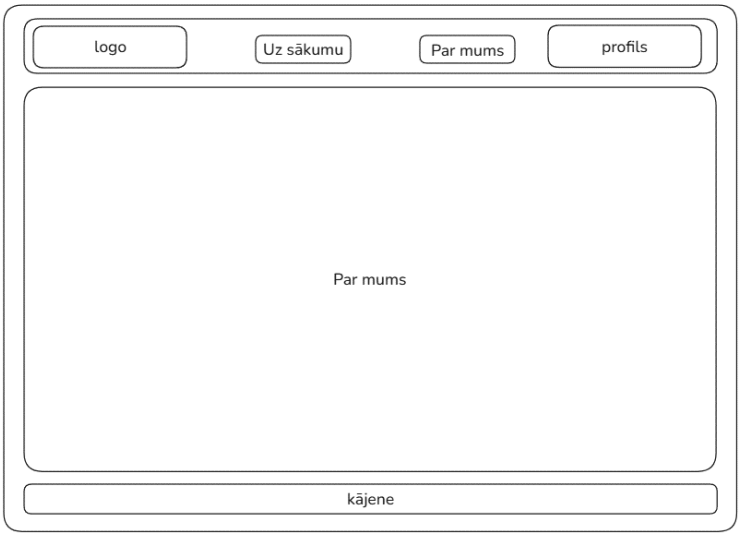
Šajā logā (skatīt 15. attēlu) vajadzēs apstiprināt konta izdzēšanu.



15.attēls. **Konta dzēšanas struktūrskice**

3.2.10. Par mums lapa:

Šajā lapā (skatīt 16. attēlu) būs šī projekta apraksts, kad to sāka veidot, kāds bija progress, kādus rīkus izmantoja, kurš to veidoja u.c.

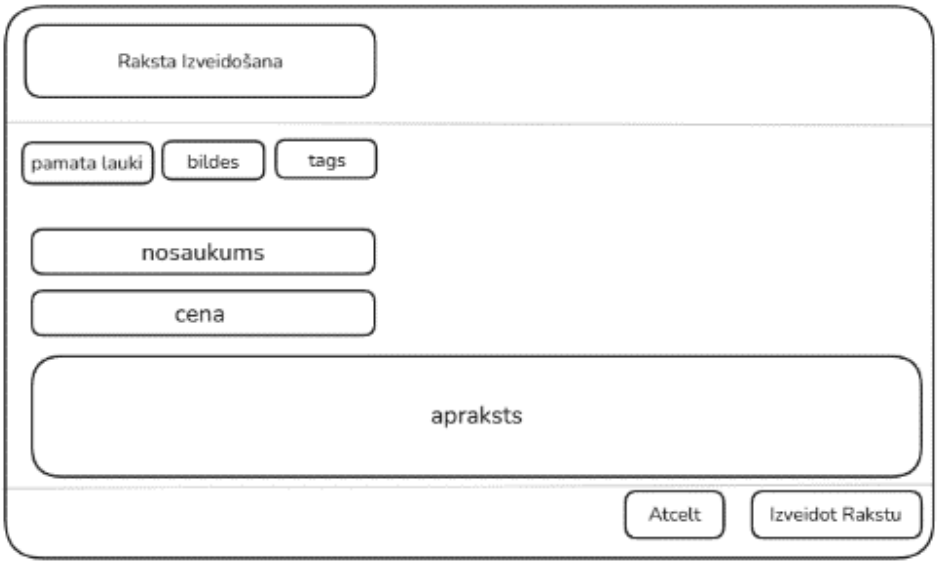


16.attēls. **Par mums lapas struktūrskice**

3.2.11. Rakstu veidošana:

3.2.11.1. Raksta formas pamata lauki:

Šajā laukā (skatīt 17. attēlu) lietotājs ievada raksta nosaukumu, kategoriju, aprakstu.



17.attēls. **Raksta formas pamata lauki struktūrskice**

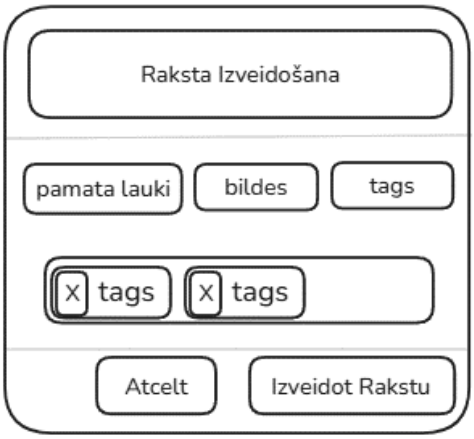
3.2.11.2. Raksta formas bildes lauks:

Šajā laukā (skatīt 18. attēlu) lietotājs izvēlas bildes izmantojot failu meklētāju.

18.attēls. **Raksta formas bildes struktūrskice**

3.2.11.3. Raksta formas *tags*:

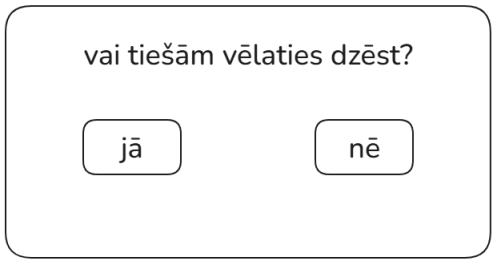
Šajā laukā (skatīt 19. attēlu) lietotājs var pievienot un noņemt *tags* kurus saglabās ar rakstu.



19.attēls. **Raksta formas *tags* struktūrskice**

3.2.12. Raksta dzēšanas apstiprinājums:

(skatīt 20. attēlu)Lietotājs apstiprina vai atceļ raksta dzēšanu



20.attēls. **Raksta dzēšanas apstiprinājums struktūrskice**

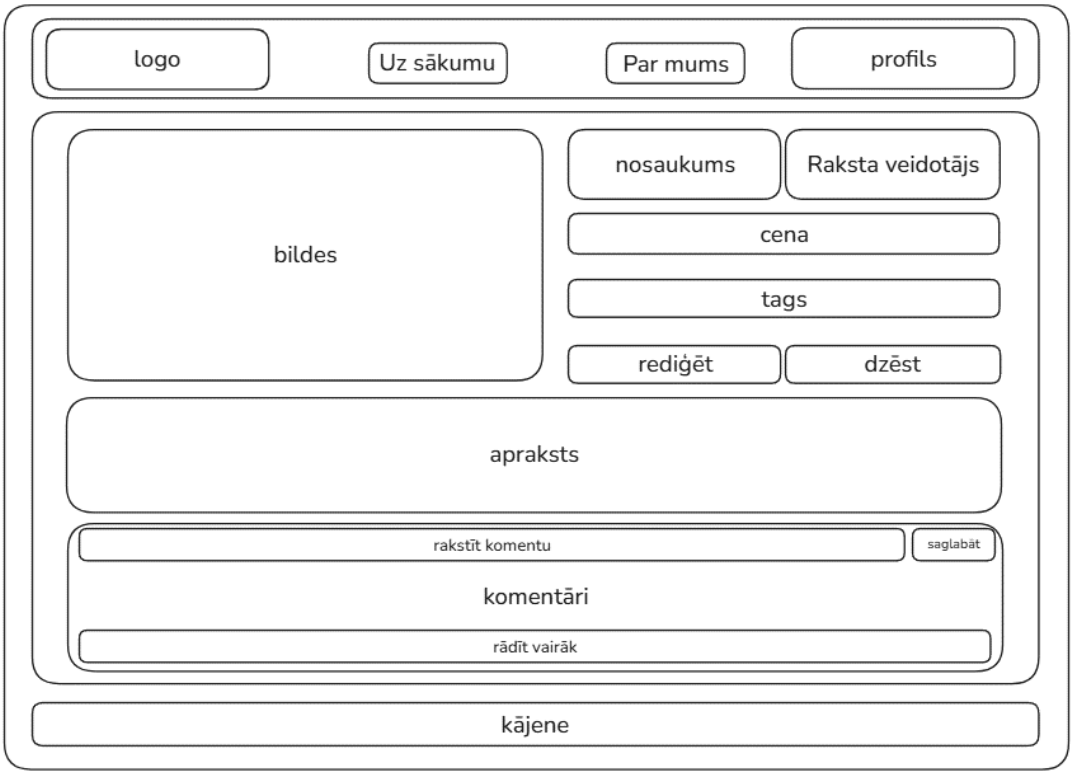
3.2.13. Raksta filtrēšanas logs

(skatīt 21. attēlu) Šajā logā tiks rādīti visi *tags* kuri ir pieejami, apakšā būs poga "ātcelt" un poga "filtrēt".

21.attēls. **Raksta filtrēšanas logs struktūrskice**

3.2.14. Raksta detalizēta informācija

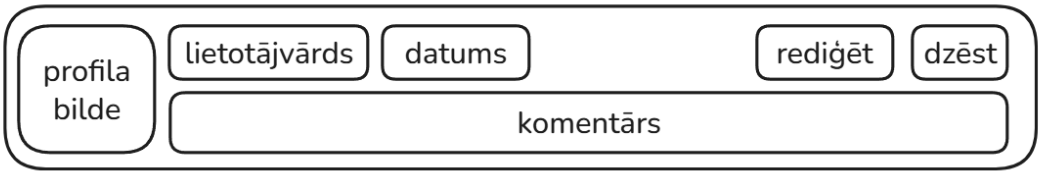
(skatīt 22. attēlu) Būs redzama visa informācija par rakstu – bildes, nosaukums, raksta veidotāja lietotājvārds un profila bilde, cena, *tags*, apraksts, komentāri, kā arī dzēšanas un rediģēšanas pogas, ja apskata raksta lietotājs. Lapas apakšā, virs kājenes, būs komentāru logs, kur varēs redzēt komentārus, un poga "rādīt vairāk" (skatīt 23. attēlu).



22.attēls. **Raksta detalizēta informācija struktūrskice**

3.2.15. Rakstu komentāri

Komentāru logos (skatīt 23. attēlu) būs redzams komentētāja profila bilde, lietotājvārds, datums kad komentāru veidoja, komentāra teksts, un rediģēšanas, dzēšanas pogas ja apskata tas, kurš veidoja komentāru.



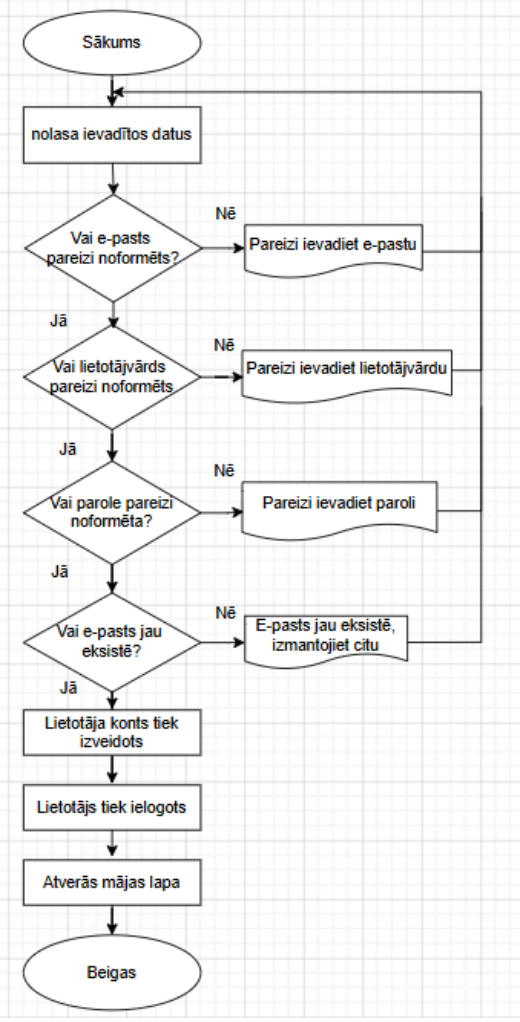
23. attēls. **Komentāru struktūrskice**

3.3. Funkcionālas prasības

3.3.1. Galvenes pogu funkcijas

- (skatīt 9. attēlu) uzspiežot pogu “par mums” tiks atvērta cita mājas lapa (skatīt 16. attēlu). Konts kuru lietotājs izmanto nemainās.
- (skatīt 9. attēlu) Uzspiežot uz lapas logo vai uz pogu “Uz sākumu”, atvērsies mājas lapa.

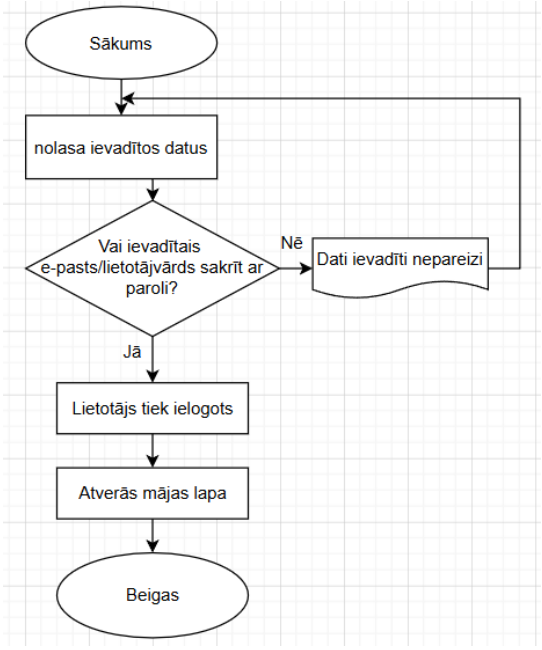
3.3.2. Konta izveidošana



24.attēls. **Konta veidošanas blokshēma**

- (skatīt 12. attēlu) uzspiežot pogu “jauns lietotājs?” atvērsies konta izveidošanas logs(skatīt 13. attēlu). Lietotājs ievada lietotājvārdu, e-pastu, paroli(skatīt 24. attēlu). Nospiežot pogu “izveidot”, šie dati tiek šifrēti un sūtīti uz datubāzi, kur tie tiek noformēti un saglabāti.

3.3.3. Lietotāja ielogošana



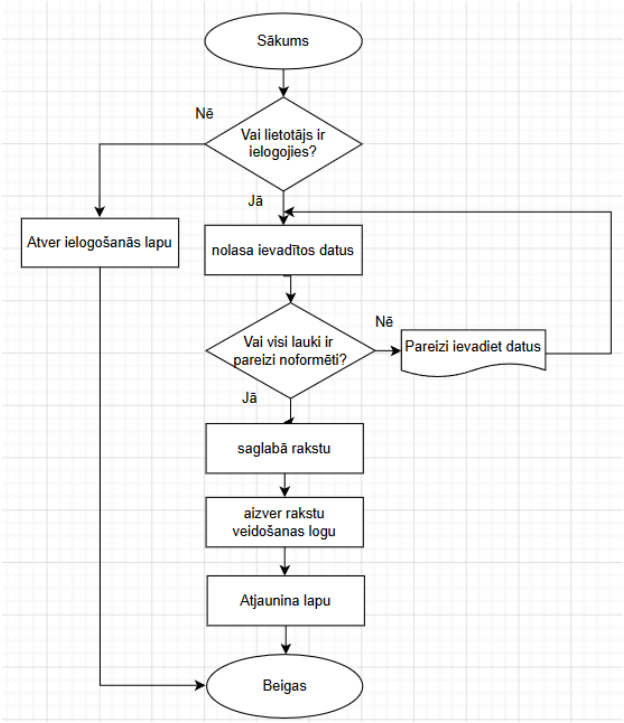
25.attēls. **Ielogošanās blokshēma**

- (skatīt 9. attēlu) uzspiežot uz profila atvērsies pieteikšanās logs(skatīt 12. attēlu). šajā logā lietotājs ievada e-pastu/lietotājvārdu, paroli, nospiežot pogu "pieslēgties", e-pasts\lietotājvārds, parole tiek aizsūtīti uz datubāzi, kur tiek pārbaudīts vai tāds lietotājs eksistē, ja tāds ir, tad lietotājs tiek ielaists savā kontā, ja tāds nav, tad paziņo ka dati ievadīti nepareizi.

3.3.4. Profila bildes mainīšana

- (skatīt 13.attēls. **Konta izveidošanas loga** attēlu) uzspiežot uz profila bildes, atvērsies datora failu pārlūks, kur lietotājs var izvēlēties bildi kuru izmantot savā profilā.

3.3.5. Raksta veidošana



26.attēls. **Rakstu veidošanas blokshēma**

- (skatīt 9. attēlu) uzspiežot uz pogas "izveidot jaunu", atvērsies logs ar vairākām sadaļām, pamata lauki, bildes, *tags* (skatīt 17. attēlu). Pamata lauka nodaļā lietotājs ievada raksta nosaukumu, cenu, aprakstu. Bildes logā var izvēlēties 0-15 bildēm, izmantojot failu meklētāju. *Tags* logā var izvedot un izdzēst savus *tags* kurus saglabās ar šo rakstu (skatīt 26. attēlu). Spiežot pogu "atcelt" šis logs aizvērsies, un atverot šo logu atkal neatjauninot lapu, visa ievadītā informācija nepazudīs.

3.3.6. Raksta rediģēšana

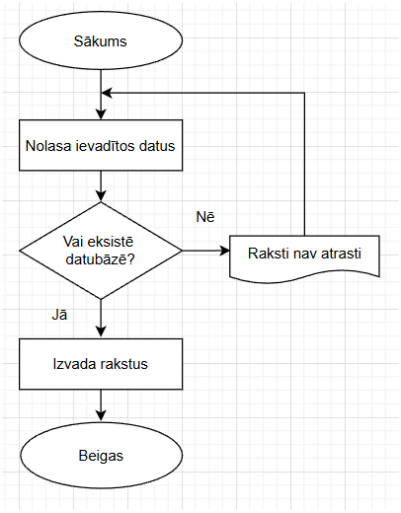
- (skatīt 22.attēls. **Raksta detalizēta informācija struktūrskice** attēlu) uzspiežot uz pogas "rediģēt", atvērsies logs ar vairākām formu logiem – pamata lauki, bilde, *tags* (skatīt 17. att.), katrs lauks būs aizpildīts ar tiem datiem kas šajā rakstā ir saglabāti. Raktu rediģēšanai izmanto kodu lai vajadzīgo informāciju aizsūtītu uz aizmugursistēmu (skatīt 6. pielikumu), kur to apstrādā un saglabā.

3.3.7. Raksta dzēšana

- (skatīt 22.attēls. **Raksta detalizēta informācija struktūrskice** attēlu) uzspiežot uz pogas "dzēst", atvērsies logs(skatīt 20. attēlu). spiežot pogu "nē", logs aizvērsies un nekas nenotiks, spiežot pogu "jā", logs aizvērsies un izvēlētais raksts tiks izdzēsts

3.3.8. Raksta filtrēšana

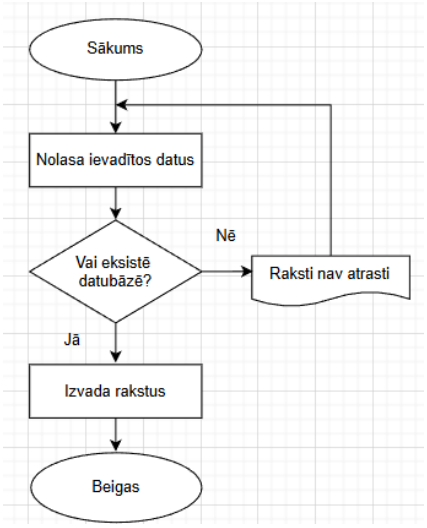
- (skatīt 21. attēlu) *tags* logā visi *tags* tiks rādīti kā pogas, uzspiežot (izvēlējoties) to krāsa nomainās uz tumši zilu, uzspiežot atkal (atceļot izvēli) to krāsa nomainās uz zilu. Uzspiežot pogu "filtrēt", šis logs aizvērās un visi raksti kas satur visus izvelētos *tags* tiks parādīti logā "raksti" (skatīt 9. un 27. attēlu).



27.attēls. **Rakstu filtrēšanas blokshēma**

3.3.9. Raksta meklēšana

- (skatīt 9. attēlu) Uz loga "ko meklēt" var uzspiest un ierakstīt raksta nosaukumu, un uzspiežot pogu "meklēt", visi raksti pazudīs no loga "raksti", un parādīsies tie raksti, kuru nosaukums satur to, kas ir ierakstīts logā "ko meklēt".



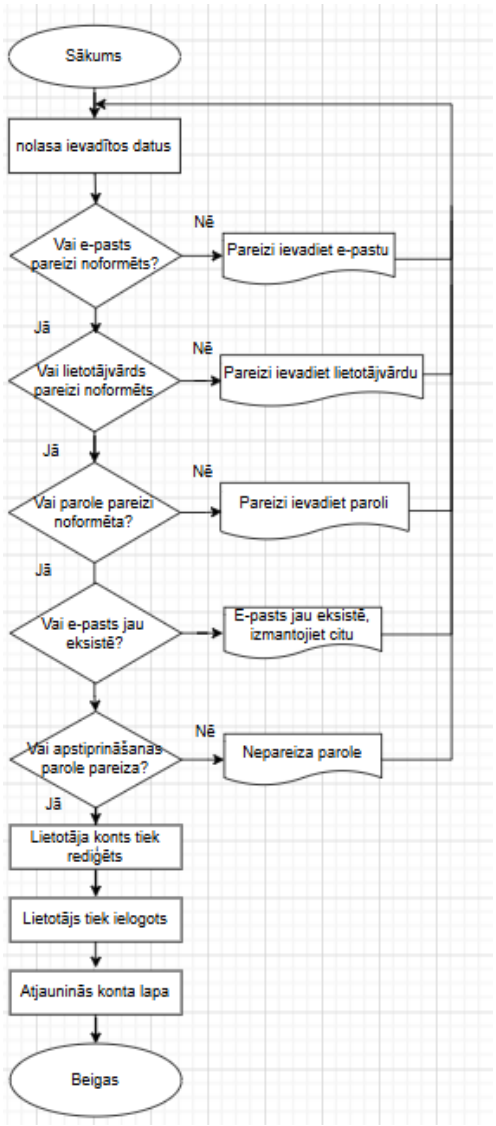
28.attēls. **Rakstu meklēšanas bl okshēma**

3.3.10. Raksta informācija

- Uzspiežot uz pašu rakstu(skatīt 9. attēlu), atvērsies lapa, kas saturēs raksta detalizētu informāciju(skatīt 22. attēlu).

3.3.11. Konta rediģēšana

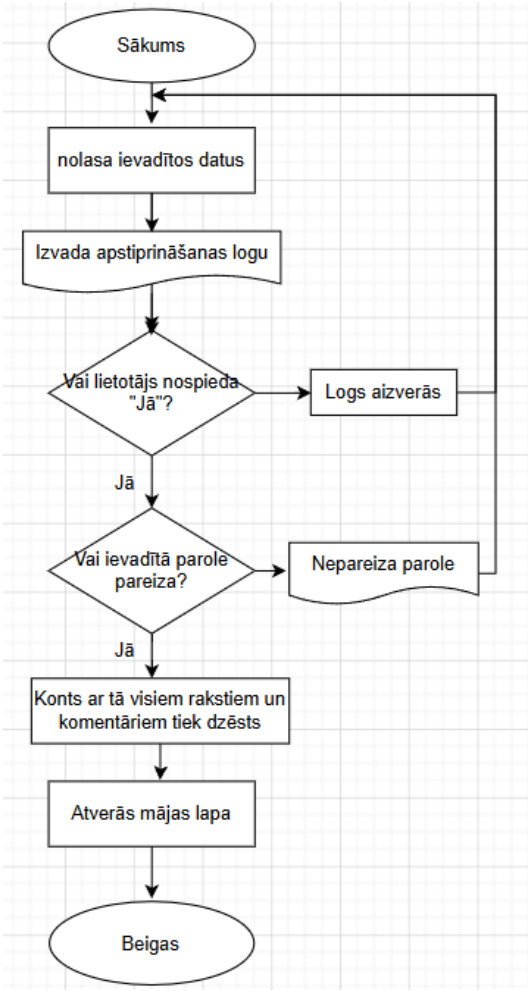
- Uzspiežot uz Profila loga (skatīt 8. attēlu), atverās konta lapa (skatīt 14. attēlu). E-pasta, lietotājvārda un bildes lauki jau būs aizpildīti ar saglabātajiem datiem. Visus šos laukus var nomainīt, lai mainītu konta informāciju. Lai saglabātu, lietotājam jāaizpilda "parole apstiprināšanai" lauks ar tagadējo paroli, lai apstiprinātu konta rediģēšanu, un saglabāto izmainīto informāciju (skatīt 29. attēlu).



29. attēls. **Konta rediģēšanas blokshēma**

3.3.12. Konta dzēšana

- Nospiežot pogu “Dzēst kontu”, atvēries apstiprināšanas logs (skatīt 15. attēlu). Nospiežot “Nē”, notiks nekas un logs aizvērsies, nospiežot pogu “Jā”, tiks pārbaudīta parole, ja tā ir nepareiza, tad paziņos ka tā ir nepareiza, ja tā ir pareiza, tad konts un tā visi raksti un komentāri tiks dzēsti (skatīt 30. attēlu).



30. attēls. **Konta dzēšanas blokshēma**

3.4. Sistēmas struktūras modelis

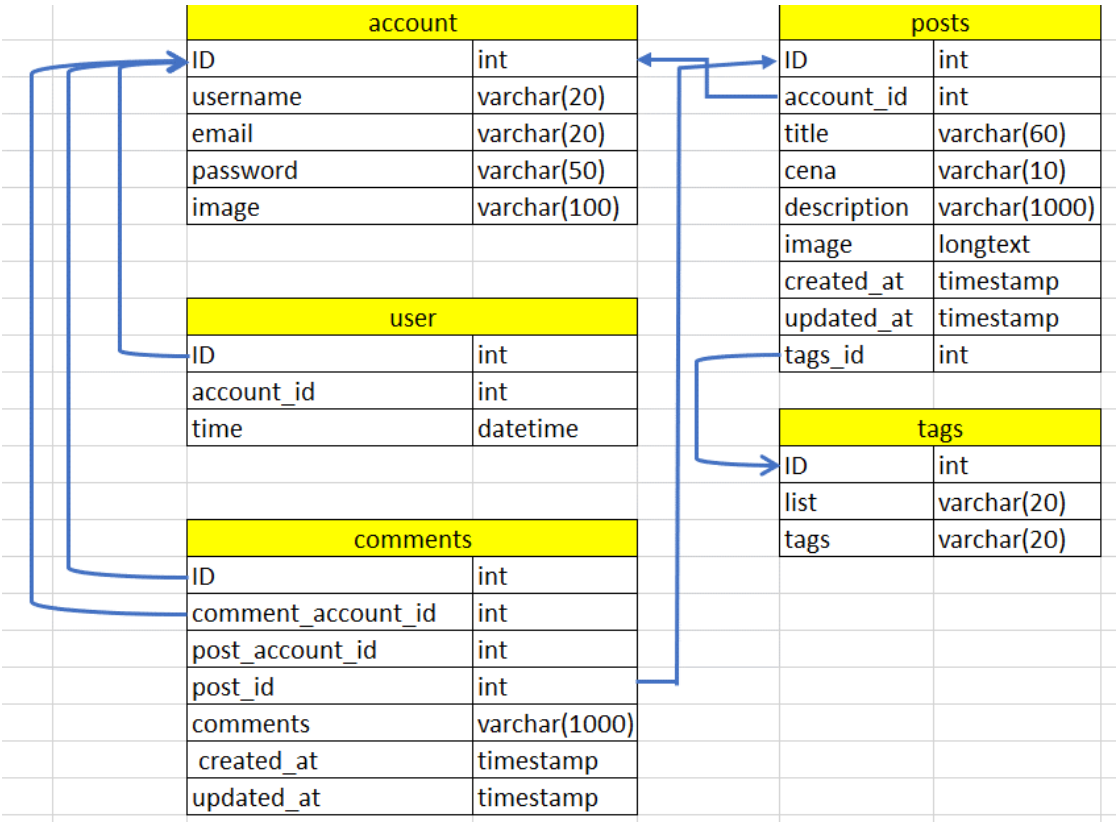
3.4.1. Datubāzes apraksts

Datubāzes nosaukums ir “norokas_db”, to pārvalda ar rīku “*HeidiSQL*” un tās dzinējs ir “*MySQL*”.

Šajā datubāzē atrodas 4 tabulas:

- “account”(skatīt 1. tabula)glabā informāciju par izveidotajiem kontiem
- “user”(skatīt 2. tabula), glabā informāciju par lietotājiem, kas pašlaik ir ielogojušies

- ”posts”(skatīt 3. tabula), glabā informāciju par veidotajiem rakstiem
- “tags”(skatīt 4. tabula), glabā katra raksta *tags*



31.attēls. **Datubāzes struktūras diagramma**

Noklusējuma rakstzīmju kopa datubāzē ir *utf8mb4*.

3.4.2. Tabula “account”

Šī tabula (skatīt 1. tabula tabulu) glabā informāciju par sistēmas lietotāju kontiem. Tā satur lietotāju unikālo identifikatoru, lietotājvārdu, e-pastu, mobilo numuru, paroli un lietotāja profila bildes nosaukumu.

1. tabula **“account”**

Nosaukums	Tips	Atribūti	Komentāri
ID	int	NOT NULL, PRIMARY KEY, AUTO INCREMENT	Unikāls identifikators kontam
username	varchar(20)	NOT NULL	Lietotāja lietotājvārds
email	varchar(20)	NOT NULL	Lietotāja e-pasta adrese
password	varchar(150)	NOT NULL	Šifrēta lietotāja parole
image	varchar(100)	NOT NULL	Lietotāja profila bilde

3.4.3. Tabula “user”

Šī tabula (skatīt 2. tabula tabulu) glabā informāciju par lietotājiem, kas pašlaik ir ielogojušies un izmanto saiti. Tā satur lietotāja *ID* un laiku, kad lietotājs ielogojās, kas tiek atjaunināts pēc katras lietotāja darbības(atver citu lapu vai logu, atjaunina lapu u.c.).

2. tabula **“user”**

Nosaukums	Tips	Atribūti	Komentāri
ID	Int	NOT NULL, PRIMARY KEY, AUTO INCREMENT	Unikāls identifikators lietotājam
account_id	Int	NOT NULL	Lietotāja konta ID
time	datetime	NOT NULL	Laiks, kad lietotājas ielogojās

3.4.4. Tabula “posts”

Šī tabula (skatīt 3. tabula tabulu) glabā visu informāciju par pašu rakstu. Tā satur veidotāja konta *ID*, nosaukums, cena, apraksts, bilde/es, laiks kad izveidoja un pēdējo reizi izmainīja, un *tags ID*, ar ko var atrast raksta *tags* no citas tabulas (skatīt 3. tabula tabulu).

3. tabula **“post”**

Nosaukums	Tips	Atribūti	Komentāri
ID	int	NOT NULL, PRIMARY KEY, AUTO INCREMENT	Unikāls identifikators rakstam
account_id	int	NOT NULL	Lietotāja konta ID
title	varchar(50)	NOT NULL	Raksta nosaukums
price	varchar(10)	NOT NULL	Raksta cena
description	varchar(5000)	NOT NULL	Raksta apraksts
image	varchar(5000)	NOT NULL	Saraksts ar raksta bildēm
created_at	timestamp	NOT NULL	Datums un laiks, kad rakstu izveidoja
updated_at	timestamp		Datums un laiks, kad rakstu pēdējo reizi izmainīja
tags_id	varchar(50)	NOT NULL	Saraksts ar *tag* ID, kurus šis raksts izmanto

3.4.5. Tabula “tags”

Šī tabula (skatīt 4. tabula. tabulu) glabā visus *tags* kas ir veidoti. Katram *tag* ir savs unikāls *ID*.

4. tabula **“tags”**

Nosaukumi	Tips	Atribūti	Komentāri
ID	int	NOT NULL, PRIMARY KEY, AUTO INCREMENT	Unikāls identifikators priekš *tag*
list	varchar(20)	NOT NULL	Saraksts, pie kura pieder *tags*
tags	varchar(20)	NOT NULL	*Tags* kuru raksts izmantos

3.4.6. Tabula “comments”

Šī tabula (skatīt 5. tabula **“comments”** tabulu) glabā visu vajadzīgo par komentāru vienā laukā, tā satur komentāra ID, konta ID tam, kurš to rakstīja, konta ID tam, kurš veidoja rakstu, paša raksta ID, paša komenta saturs, kad tas ir bijis rakstīts un pēdējo reizi rediģēts.

5. tabula **“comments”**

Nosaukumi	Tips	Atribūti	Komentāri
ID	int	NOT NULL, PRIMARY KEY, AUTO INCREMENT	Unikāls identifikators priekš *tag*
comment_account_id	int	NOT NULL	Id tam kontam, kas rakstīja komentu
post_account_id	int	NOT NULL	Id tam kontam, kas veidoja rakstu
post_id	int	NOT NULL	Raksta id
comment	varchar(1000)	NOT NULL	Komenta saturs
created_at	timestamp	NOT NULL	Kad veidots
updated_at	timestamp		Kad pēdējo reizi rediģēts

4. TESTĒŠANA

4.1. Lietotāja reģistrēšanās funkcijas testēšana

Testa mērķis: pārbaudīt reģistrēšanos sistēmā.

Testa gadījums 1:

Situācija: Lietotājs aizpilda visus ievadlaukus, ievada e-pastu, kas neeksistē datu bāzē un parole ievadīta pēc parādītajiem noteikumiem.

Gaidītais rezultāts: Lietotājs tiek veiksmīgi reģistrēts un iegūst piekļuvi savam kontam.

Testa gadījums 2:

Situācija: Lietotājs mēģina reģistrēties, neaizpildot visus laukus.

Gaidītais rezultāts: WEB lapa parāda kļūdas ziņojumu pie neaizpildītajiem laukiem "Lūdzu aizpildiet visus laukus".

Testa gadījums 3:

Situācija: Lietotājs mēģina reģistrēties, izmantojot jau eksistējošu e-pastu.

Gaidītais rezultāts: WEB lapa parāda kļūdas ziņojumu "E-pasts jau eksistē!".

Testa rezultāts: Reālais testa rezultāts atbilst paredzētajam, tika veiksmīgi izvadīts kļūdas paziņojums, ja tāds ir paredzēts, un notika veiksmīga reģistrēšanās sistēmā, ja visi ievadlauki aizpildīti pareizi.

6. tabula **reģistrācijas testpiemēri**

N.p.k.	Situācija	Gaidāmais rezultāts	Testa rezultāts
1.	Lietotājs aizpilda visus ievadlaukus, ievada e-pastu, kas neeksistē datu bāzē un parole ievadīta pēc parādītajiem noteikumiem.	Lietotājs tiek veiksmīgi reģistrēts un iegūst piekļuvi savam kontam.	Veiksmīga reģistrācija, konts izveidots
2.	Lietotājs mēģina reģistrēties, neaizpildot visus laukus.	WEB lapa parāda kļūdas ziņojumu pie neaizpildītajiem laukiem "Lūdzu aizpildiet visus laukus".	Kļūdas paziņojums pie neaizpildītiem laukiem parādās
3.	Lietotājs mēģina reģistrēties, izmantojot jau eksistējošu e-pastu.	WEB lapa parāda kļūdas ziņojumu "E-pasts jau eksistē!".	Kļūdas ziņojums par pastāvošu e-pastu parādīts

4.2. Lietotāja pieteikšanās funkcijas testēšana

Testa gadījums 1:

Situācija: Lietotājs veiksmīgi ievada savu e-pastu/lietotājvārdu un paroli.

Gaidītais rezultāts: Lietotājs ir veiksmīgi pieslēdzies savam kontam.

Testa gadījums 2:

Situācija: Lietotājs mēģina pieteikties, ievadot paroli, kas neatbilst konkrētajam pastam/lietotājvārdam

Gaidītais rezultāts: Sistēma parāda kļūdas ziņojumu "E-pasts vai parole nepareiza!".

Testa gadījums 3:

Situācija: Lietotājs mēģina pieteikties, ievadot e-pastu/lietotājvārdu, kas neeksistē sistēmā

Gaidītais rezultāts: Sistēma parāda kļūdas ziņojumu " E-pasts vai parole nepareiza!".

Testa rezultāts: Gaidāmie testa rezultāti atbilst paredzētajiem, tika veiksmīgi izvadīts kļūdas paziņojums, ja tāds ir paredzēts, un notika veiksmīga pieteikšanās sistēmā, ja visi ievadlauki aizpildīti pareizi.

7. tabula **pieteikšanās testpiemēri**

N.p.k.	Situācija	Gaidāmais rezultāts	Testa rezultāts
1.	Lietotājs veiksmīgi ievada savu e-pastu/lietotājvārdu un paroli.	Lietotājs ir veiksmīgi pieslēdzies savam kontam.	Veiksmīga pieteikšanās, konts atvērts
2.	Lietotājs mēģina pieteikties, ievadot paroli, kas neatbilst konkrētajam e-pastam/lietotājvārdam	Sistēma parāda kļūdas ziņojumu "E-pasts vai parole nepareiza!".	Kļūdas paziņojums parādīts
3.	Lietotājs mēģina pieteikties, ievadot e-pastu/lietotājvārdu, kas neeksistē sistēmā	Sistēma parāda kļūdas ziņojumu " E-pasts vai parole nepareiza!".	Kļūdas ziņojums parādīts

4.3. Raksta veidošanas funkcijas testēšana

Testa gadījums 1:

Situācija: lietotājs aizpilda visus laukus un uzspiež pogu “izveidot rakstu”.

Gaidītais rezultāts: raksts tiek izveidots un lapa atsvaidzinās.

Testa gadījums 2:

Situācija: lietotājs izvēlas vairāk nekā 10 bildes.

Gaidāmais rezultāts: parādās paziņojums par to, ka nevar izvelēties vairāk par 10 bildēm.

Testa gadījums 3:

Situācija: lietotājs neizvēlas nevienu *tag*, un uzspiež uz pogu "izveidot rakstu".

Gaidāmais rezultāts: parādās paziņojums par to, ka vajag izvēlēties vismaz vienu *tag*.

Testa rezultāts: Gaidāmie testa rezultāti atbilst paredzētajiem, tika veiksmīgi izvadīts kļūdas paziņojums, ja tāds ir paredzēts, un notika veiksmīga raksta veidošana, ja visi ievadlauki aizpildīti pareizi.

8. tabula **Raksta veidošanas funkcijas testēšana**

N.p.k.	Situācija	Gaidāmais rezultāts	Testa rezultāts
1.	lietotājs aizpilda visus laukus un uzspiež pogu "izveidot rakstu".	raksts tiek izveidots un lapa atsvaidzinās.	Veiksmīga raksta izveidošana un lapa atsvaidzinās.
2.	lietotājs izvēlas vairāk nekā 10 bildes.	parādās paziņojums par to, ka nevar izvelēties vairāk par 10 bildēm.	Kļūdas paziņojums parādīts
3.	lietotājs neizvēlas nevienu tag, un uzspiež uz pogu "izveidot rakstu".	parādās paziņojums par to, ka vajag izvēlēties vismaz vienu tag.	Kļūdas paziņojums parādīts

4.4. Konta rediģēšanas funkcijas testēšana

Testa gadījums 1:

Situācija: lietotājs ievada visus laukus pareizi un nospiež pogu "saglabāt".

Gaidāmais rezultāts: konts tiek rediģēts un atvērās mājas lapa.

Testa gadījums 2:

Situācija: lietotājs ievada nomainītos datus, bet neievada apstiprināšanas paroli pareizi.

Gaidāmais rezultāts: parādās ziņojums par to, ka parole ievadīta nepareizi.

Testa gadījums 3:

Situācija: lietotājs nav ievadījis jauno paroli pēc parādītajiem noteikumiem.

Gaidāmais rezultāts: parādās ziņojums par to, ka parole ievadīta nepareizi.

Testa rezultāts: Gaidāmie testa rezultāti atbilst paredzētajiem, tika veiksmīgi izvadīts kļūdas paziņojums, ja tāds ir paredzēts, un notika veiksmīga konta rediģēšana, ja visi ievadlauki aizpildīti pareizi.

9. tabula **Konta rediģēšanas funkcijas testēšana**

N.p.k.	Situācija	Gaidāmais rezultāts	Testa rezultāts
1.	lietotājs ievada visus laukus pareizi un nospiež pogu "saglabāt".	konts tiek rediģēts un atvērās mājas lapa.	Veiksmīga konta rediģēšana
2.	lietotājs ievada nomainītos datus, bet neievada apstiprināšanas paroli pareizi.	parādās ziņojums par to, ka parole ievadīta nepareizi.	Kļūdas paziņojums parādīts
3.	lietotājs nav ievadījis jauno paroli pēc parādītajiem noteikumiem.	parādās ziņojums par to, ka parole ievadīta nepareizi.	Kļūdas paziņojums parādīts

4.5. Konta dzēšanas funkcijas testēšana

Testa gadījums 1:

Situācija: lietotājs ievada apstiprināšanas paroli pareizi un nospiež pogu "izdzēst kontu".

Gaidāmais rezultāts: konts un visi rakti, komentāri ko konts veidoja, tiek dzēsts.

Testa gadījums 2:

Situācija: lietotājs ievada apstiprināšanas paroli nepareizi.

Gaidāmais rezultāts: parādās ziņojums par to, ka parole ievadīta nepareizi.

Testa rezultāts: Gaidāmie testa rezultāti atbilst paredzētajiem, tika veiksmīgi izvadīts kļūdas paziņojums, ja tāds ir paredzēts, un notika veiksmīga konta dzēšana, ja visi ievadlauki aizpildīti pareizi.

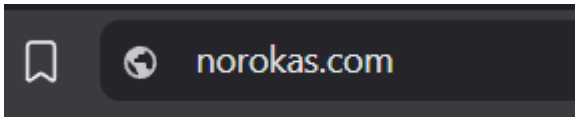
10. tabula **Konta dzēšanas funkcijas testēšana**

N.p.k.	Situācija	Gaidāmais rezultāts	Testa rezultāts
1.	lietotājs ievada apstiprināšanas paroli pareizi un nospiež pogu "izdzēst kontu".	konts un visi rakti, komentāri ko konts veidoja, tiek dzēsts.	Veiksmīga konta dzēšana
2.	lietotājs ievada apstiprināšanas paroli nepareizi.	parādās ziņojums par to, ka parole ievadīta nepareizi.	Kļūdas paziņojums parādīts

5. LIETOTĀJA CEĻVEDIS

5.1. Interneta saites piekļūšana

Interneta saite ir pieejama šajā domēnā: https://norokas.com/. Lai atvērtu interneta veikalu, ir nepieciešams jebkura pārlūkprogramma, piemēram Google Chrome vai Microsoft Edge, kuras meklēšanas joslā ievadīt iepriekš minēto saiti. (skat. 32. attēls **Pārlūkprogrammas meklēšanas josla** att.).



32. attēls **Pārlūkprogrammas meklēšanas josla**

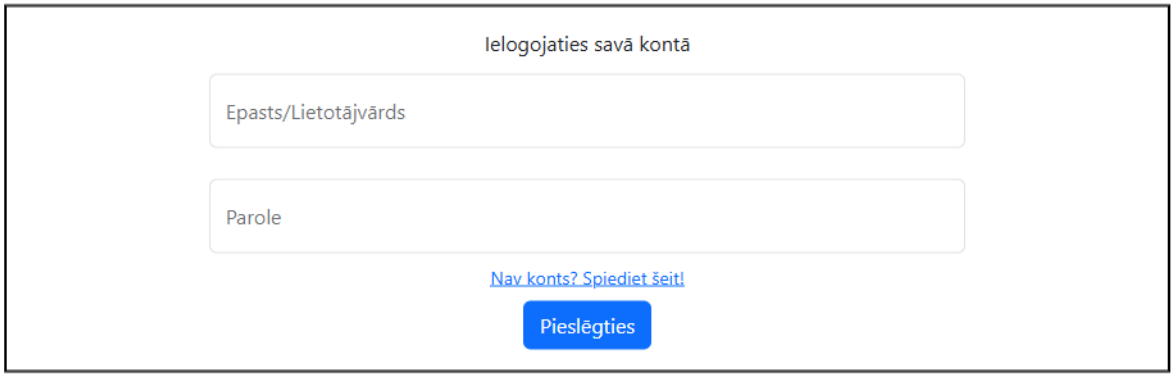
5.2. Navigācija uz konta pieslēgšanos lapu

Lietotājs var tikt pie pieslēgšanās lapu, nospiežot uz navigācijas lauka labajā pusē, kur tiek rādīts, ka lietotājs nav pašlaik ielogojies(skat. 33. attēls **Pieslēgšanās lapas atrašana** att.).

33. attēls **Pieslēgšanās lapas atrašana**

5.3. Pieslēgšanās kontā

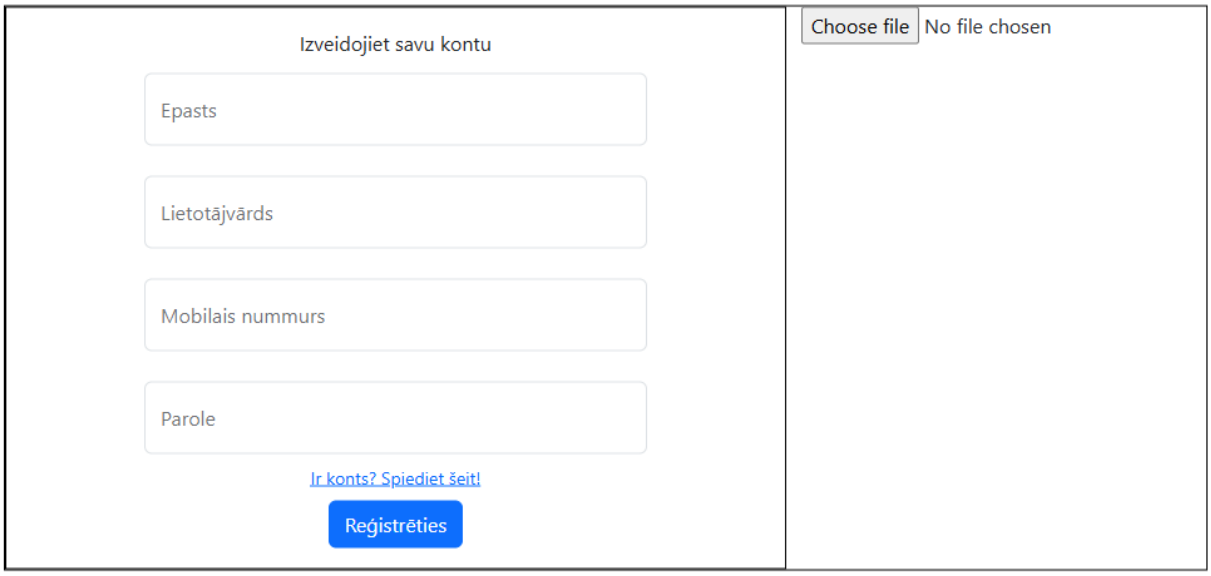
Ielogošanās lapā (skat. 34. attēls **Ielogošanās lapa** att.) lietotājam vajag ievadīt sava konta e-pastu/lietotājvārdu un paroli, un nospiežot pogu “pieslēgties”, lietotājs tiek ielogots savā kontā un pārvietots uz mājas lapu.



34. attēls **Ielogošanās lapa**

5.4. Konta izveidošana

Ja lietotajam nav jau izveidots konts, tad vajag uzspiest uz pogu “Nav konts? Spiediet šeit!” ielogošanās lapā (skat. 34. attēls **Ielogošanās lapa** att.), kas atvērs konta veidošanas lapu (skat. 35. attēls **Konta veidošanas lapa** att.). kurā lietotājam vajag ievadīt savu e-pastu, lietotājvārdu, mobilo numuru, paroli, un profila bildi, ja vēlas savu izvēlētu bildi.



35. attēls **Konta veidošanas lapa**

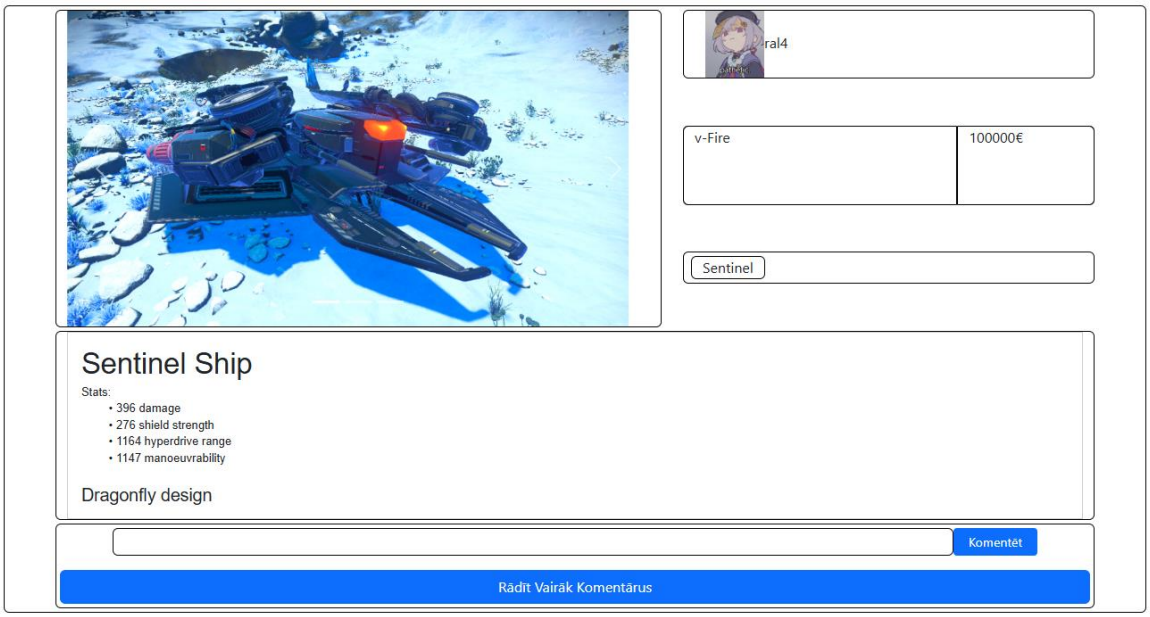
5.5. Rakstu apskatīšana

Visi lietotāju veidotie raksti būs redzami saites mājas lapā (skat. 36. attēls **Rakstu apskatīšana** att.), to nosaukums, izvēlētā cena, un datums kad izveidoja. Nospiežot uz raksta ar peles kreiso pogu atvērsies raksta informācijas lapa (skat. 37. attēls **Raksta informācijas lapa** att.).

36. attēls **Rakstu apskatīšana**

5.6. Raksta informācijas lapa

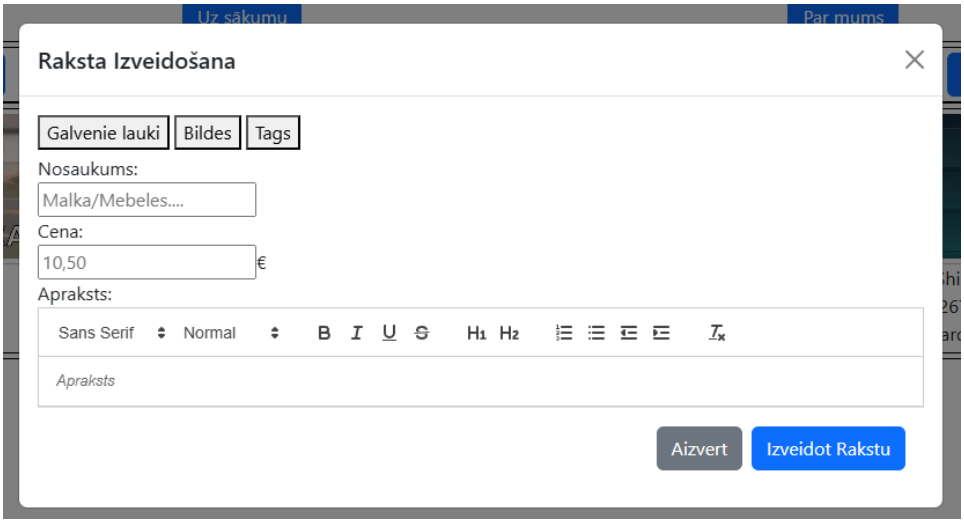
Šajā lapā (skat. 37. attēls **Raksta informācijas lapa** att.) būs redzama visa svarīgā informācija par rakstu, visas bildes, kurš veidoja rakstu, izmantotie *tags*, apraksts un lauks komentēšanai.



37. attēls **Raksta informācijas lapa**

5.7. Raksta veidošana

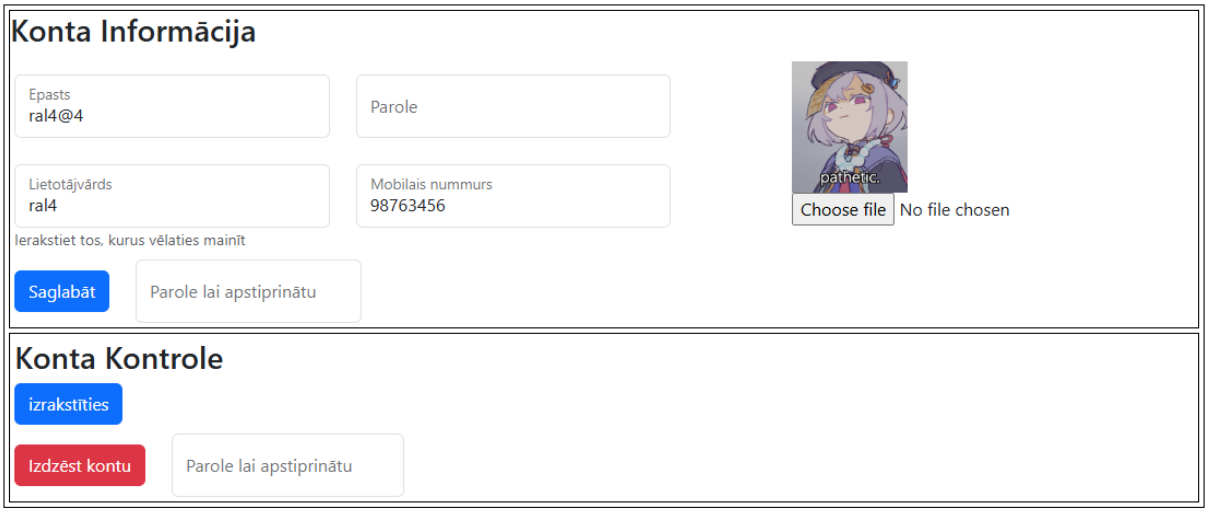
Nospiežot pogu "izveidot" mājas lapā (skat. 36. attēls **Rakstu apskatīšana** att.), atvērsies raksta veidošanas logs (skat 38. attēls **Raksta veidošanas logs** att.), kurā vajag ievadīt visu svarīgo informāciju par rakstu, un ar pogu "izveidot rakstu" tas tiks saglabāts.



38. attēls **Raksta veidošanas logs**

5.8. Darbības ar kontu

Nospiežot uz navigācijas lauka labajā pusē (skat. 33. attēls **Pieslēgšanās lapas atrašana** att.) tad, kad jau lietotājs ir ielogojies savā kontā, atvērsies konta lapa (skat. 39. attēls **Konta lapa** att.), kurā var veikt darbības ar kontu. Lietotājs var nomainīt sava konta e-pastu, lietotājvārdu, paroli un mobilo numuru, vispirms ievadot tagadējo paroli lai apstiprinātu izmaiņas, un tad nospiežot pogu "saglabāt". Lietotājs var jebkurā laikā izrakstīties no sava konta nospiežot pogu "izrakstīties", kā arī dzēst savu kontu, vispirms apstiprinot ievadot tagadējo paroli.



39. attēls **Konta lapa**

SECINĀJUMI

1. WEB lapas izstrāde izdevās veiksmīgi, atbilstoši pasūtītāja vēlmēm.
2. Tika izveidoti visi galvenie komponenti, lietotāju reģistrēšana, konta veidošana un rediģēšana, rakstu veidošana un komentāru veidošana rakstu detalizēta apskatīšana, kas parāda visu svarīgo par rakstu.
3. Nodrošināts, ka visa nepieciešamā informācija ir viegli pieejama, palīdzot lietotājiem pieņemt informētus pirkuma lēmumus
4. Katram raksta ir pieejami tikai 15 *tags*, kas ir mazāk nekā vajadzētu būt pabeigtam projektam, bet tas ir pietiekami prototipa funkcionalitātes parādīšanai.
5. Šis bija pirmais nopietns projekts ar daudzām funkcijām un *codeigniter* ietvaru, kā arī vairākām *JavaScript* bibliotēkām, kas ļoti paplašināja manas zināšanas par WEB palu izstrādi.

IZMANTOTI AVOTI

1. codeignitor.com [Tiešsaiste] Pieejams: https://codeigniter.com/user_guide/intro/index.html [skatīts 18.09.2025.].
2. developer.mozilla.org [Tiešsaiste] Pieejams: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model [skatīts 18.09.2025.].
3. facebook.com [Tiešsaiste] Pieejams: https://www.facebook.com/marketplace/ [skatīts 06.01.2025.].
4. heidisql.com [Tiešsaiste] Pieejams: https://www.heidisql.com/ [skatīts 13.01.2025.].
5. infoworld.com [Tiešsaiste] Pieejams: https://www.infoworld.com/article/2335960/what-is-visual-studio-code-microsofts-extensible-code-editor.html [skatīts 14.01.2025.].
6. ne.wikipedia.org [Tiešsaiste] Pieejams: https://en.wikipedia.org/wiki/PHP [skatīts 13.01.2025.].
7. ne.wikipedia.org [Tiešsaiste] Pieejams: https://en.wikipedia.org/wiki/HTML [skatīts 13.01.2025.].
8. ne.wikipedia.org [Tiešsaiste] Pieejams: https://en.wikipedia.org/wiki/JavaScript [skatīts 13.01.2025..]
9. ne.wikipedia.org [Tiešsaiste] Pieejams: https://en.wikipedia.org/wiki/CSS [skatīts 13.01.2025..]
10. ne.wikipedia.org [Tiešsaiste] Pieejams: https://en.wikipedia.org/wiki/Post [skatīts 14.01.2025.].
11. ne.wikipedia.org [Tiešsaiste] Pieejams: https://en.wikipedia.org/wiki/Tag_(metadata) [skatīts 14.01.2025.].
12. ne.wikipedia.org [Tiešsaiste] Pieejams: https://en.wikipedia.org/wiki/JavaScript [skatīts 18.09.2025.].
13. ne.wikipedia.org [Tiešsaiste] Pieejams: https://en.wikipedia.org/wiki/MySQL [skatīts 18.09.2025.].
14. ne.wikipedia.org [Tiešsaiste] Pieejams: https://en.wikipedia.org/wiki/API [skatīts 18.09.2025.].
15. ne.wikipedia.org [Tiešsaiste] Pieejams: https://en.wikipedia.org/wiki/CSS [skatīts 18.09.2025.].

16. ptmbeles.lv [Tiešsaiste] Pieejams: https://www.ptmebeles.lv/katalogs/ [skatīts 06.01.2025.].
17. ss.com [Tiešsaiste] Pieejams: ss.com [skatīts 06.01.2025.].
18. utrupe.lv [Tiešsaiste] Pieejams: https://www.utrupe.lv/ [skatīts 06.01.2025.].
19. zviedrumebeles.lv [Tiešsaiste] Pieejams: https://zviedrumebeles.lv/ [skatīts 06.01.2025.].

PIELIKUMI

1. Pielikums. ***AccountController.php***

```
public function signup()
    {
        $imagefile = $this->request->getFiles();
        if ($imagefile["image_select"] != "") {
            $newName = $imagefile["image_select"]->getRandomName();
        } else {
            $newName = "default.png";
        };
        $db = db_connect();
        $pass      =      PASSWORD_hash($this->request->getPost('password'),
PASSWORD_ARGON2ID);
        log_message('debug', $db->escape($pass));
        $query = $db->query('SELECT `create_account`(' . $db->escape($this-
>request->getPost('email'))    .    ',    '    .    $db->escape($this->request-
>getPost('username'))   .   ',   '   .   $db->escape($pass)   .   ',   '   .   $db-
>escape($newName) . ') as signup');
        $result = $query->getRow();
        $result = explode(",", $result->{"signup"});
        log_message('debug', json_encode($result));
        if ($result[1] == "account_created") {
            $session = session();
            if ($newName != "default.png") {
                $imagefile["image_select"]->move('uploads/avatar',
$newName);
            }
            $session->set('account_id', $result[0]);
            $session->set('logged_in', '1');
            $session->set('username', $this->request->getPost('username'));
            $session->set('pfp', $newName);
        } else {
            $result = "account_exists";
        }
        return $this->response->setJSON([
            'error' => false,
            'message' => $result
        ]);
    }
```

2. Pielikums. ***AccountController.php***

```
public function login($login)
    {
        $db = db_connect();
        $login = explode(",", $login);

        $email = str_contains($login[0], "@");
        if ($email == true) {
            $query = $db->query('select password from account where email =
"' . $login[0] . '"');
            $result = $query->getRow();
            $query = 'select id, username, image from account where email =
"' . $login[0] . '"';
        } else {
            $query = $db->query('select password from account where username
= "' . $login[0] . '"');
            $result = $query->getRow();
            $query = 'select id, username, image from account where username
= "' . $login[0] . '"';
        }

        if ($result != "") {
            if (password_verify($login[1], $result->{"password"}) == true) {
                $query = $db->query($query);
                $result = $query->getRow();
                log_message('debug',  "result-------------------------"  .
json_encode($result));

                $session = session();

                $session->set('account_id', $result->{"id"});
                $session->set('logged_in', '1');
                $session->set('username', $result->{"username"});
                $session->set('pfp', $result->{"image"});
                $session->set('SessionStart', time());
                $result = "loged in successfully";
            } else {
                $result = "Nepareizi ievadīti dati";
                log_message('debug', "login1");
            }
        } else {
            $result = "Nepareizi ievadīti dati";
            log_message('debug', "login2");
        }
        return $this->response->setJSON([
            'error' => false,
            'message' => $result
        ]);
    }
```

3. Pielikums. ***AccountController.php***

```
public function delete($password)
    {
        $session = session();
        $db = db_connect();
        $query = $db->query("SELECT password, image from account where id =
" . $session->get("account_id"));
        $result = $query->getRow();
        if (password_verify($password, $result->{"password"}) == true) {
            $query = $db->query("SELECT `account_deletion`('" . $session-
>get('account_id') . "') as deleted");
            $result = $query->getRow();
            if ($result->{"deleted"} != "default.png") {
                unlink('uploads/avatar/' . $result->{"deleted"});
            }
            $delete = "account deleted succesfully";
            $session->destroy();
            $session->close();
        } else {
            $delete = "nuh uh";
        }


        return $this->response->setJSON([
            'error' => false,
            'message' => $delete
        ]);
    }
```

```
public function editAccount()
    {
        $session = session();
        $db = db_connect();
        $file = $this->request->getFile('file');
        $fileName = $file->getFilename();
        if ($fileName != '') {
            $fileName = $file->getRandomName();
            $file->move('uploads/avatar', $fileName);
        } else {
            $fileName = "default.png";
        }

        $data = [
            'email' => $this->request->getPost('email'),
            'username' => $this->request->getPost('username'),
            'password' => PASSWORD_hash($this->request->getPost('password'),
PASSWORD_ARGON2ID),
            'image' => $fileName,
        ];
        $query = $db->query("SELECT email, password from account where id =
" . $session->get("account_id"));
        $result = $query->getRow();
        log_message("debug", json_encode($data));
        $email = (object) array();
        if ($result->{"email"} != $data["email"]) {
            $email = $db->query('SELECT `check_user`("' . $data["email"] .
'") as email');
            $email = $email->getRow();
        } else {
            $email->{"email"} = "0";
        }
        log_message("debug", json_encode($email));
        if ($email->{"email"} == "0") {
            if                                (password_verify($this->request-
>getPost('passwordConfirmEdit'), $result->{"password"}) == true) {
                if ($this->request->getPost("password") == "") {
                    $query = $db->query('UPDATE account set email = "' .
$data["email"] . '", username = "' . $data["username"] . '", image = "' .
$data["image"] . '" where id = ' . $session->get("account_id"));
                    $result = "success";
                    $session->set('pfp', $data["image"]);
                    $session->set('username', $data["username"]);
                } else {
                    $result = $query->getRow();
                    $query = $db->query('UPDATE account set email = "' .
$data["email"] . '", username = "' . $data["username"] . '", password = "' .
$data["password"] . '", image = "' . $data["image"] . '" where id = ' .
$session->get("account_id"));
                    $result = "success";
                    $session->set('pfp', $data["image"]);
                    $session->set('username', $data["username"]);
                }
            } else ($result = "incorrect password");
        } else {
            $result = "email_exists";
        }
        log_message("debug", json_encode($result));
```

```
    return $this->response->setJSON([
```

***AccountController.php* pielikuma turpinājums**

```
        'error' => false,
        'message' => $result,
    ]);
}
```

```
public function fetchTags() //ar ajax request dabūj visus tags, jau izveidotus
kā html lai uzreiz izmnatotu
    {
        $tags = "";
        $fullTagsSearch = "";       //veido tags priekš filtrēšanas
        $db = new \App\Models\PostModel();
        $tags = $db->query('select id, list, tags FROM tags');
        $tags = $tags->getResultArray();
        $old_list = "a";
        foreach ($tags as $tags1) {
            if ($tags1["list"] != $old_list) {
                $old_list = $tags1["list"];
                $fullTagsSearch .= '<h1>' . $tags1["list"] . '</h1>';
            }
            $fullTagsSearch   .=   '<button   id="tag_button"   type="button"
class="btn btn-primary me-1" data-bs-toggle="button" value="' . $tags1["id"]
. '">' . $tags1["tags"] . '</button>';
        }
        $fullTagsPost = "";
        $tagsList = [];
        foreach ($tags as $tags1) {     //veido tags priekš takstu veidošanas
            array_push($tagsList, $tags1["list"]);
            $tagsList = array_unique($tagsList);
        }
        $old_list = "a";
        foreach ($tagsList as $tagslist) {
            $fullTagsPost .= '<optgroup label="' . $tagslist . '">';
            foreach ($tags as $tags2) {
                if ($tags2["list"] == $tagslist) {
                    $fullTagsPost .= '<option value="' . $tags2["id"] . '">'
. $tags2["tags"] . '</option>';
                }
            }
            $fullTagsPost .= "</optgroup>";
        }
        return $this->response->setJSON([
            'error' => false,
            'searchTags' => $fullTagsSearch,
            'postTags' => $fullTagsPost
        ]);
    }
```

6. Pielikums. ***PostController.php***

```
public function add()
    {                                                   //saglabā form
        $nameArray = [];
        if ($imagefile = $this->request->getFiles()) {

            foreach      ($imagefile['titleimage']      as      $img)      {
//izveido nosaukumu kas satur uz nejaušību ģenerētus burtus un to saglabā
                $randName = $img->getRandomName();
                $img->move('uploads/avatar', $randName);
                $nameArray[] = [
                    "filename" => $randName,
                ];
            }
            log_message('debug', json_encode($nameArray));
        }

        $session = session();

        $data = [                                                   //saliek
fisu form informāciju masīvā lai to saglabātu
            'account_id' => (int) $session->get('account_id'),
            'title' => $this->request->getPost('title'),
            'price' => $this->request->getPost('price'),
            'description' => "' " . $this->request->getPost('description') .
" '",
            'image' => json_encode($nameArray),
            'created_at' => date('Y-m-d H:i:s'),
            'tags_id' => $this->request->getPost('fullTags')
        ];
        $postModel = new \App\Models\PostModel();
        //log_message('debug',    "full    data    to    send    to    db"    .
json_encode($data));
        //$postModel->save($data);                                  //saglabā
masīvu datubāzē
        $db = new \App\Models\PostModel();
        $query = ('
        insert  into  posts(account_id,  title,  price,  description,  image,
created_at, tags_id)
        values (' . (int) $session->get('account_id') . ',
        ' . $db->escape($this->request->getPost('title')) . ',
        ' . $this->request->getPost('price') . ',
        ' . $db->escape($this->request->getPost('description')) . ',
        ' . $db->escape(json_encode($nameArray)) . ',
        ' . $db->escape(date('Y-m-d H:i:s')) . ',
        ' . $db->escape($this->request->getPost('fullTags')) . ')
        ');
        log_message('debug', $query);
        $db->query($query);
        return $this->response->setJSON([
            'error' => false,
            'message' => 'Successfully added new post!'
        ]);
    }
```

```
public function fetch()
    {
         $condition = explode("-", $_GET["condition1"]);
         log_message('debug', json_encode($condition));
         //log_message('debug', json_encode($_GET[1]));
        $db = new \App\Models\PostModel();
        switch ($condition[0]) {

            case "search":
                $query  =  $db->query('select  ID,  title,  price,  image,
created_at FROM posts WHERE title LIKE "%' . $condition[1] . '%" order by id
desc limit 6');
                $result = $query->getResultArray();
                $count = $db->query('SELECT count(id) FROM posts WHERE title
LIKE "%' . $condition[1] . '%"');
                $count = $count->getResultArray();
                break;
            case "tags":
                array_shift($condition);
                $query = 'select ID, title, price, image, created_at FROM
posts WHERE  ';
                $count = 'SELECT count(id) FROM posts WHERE  ';
                foreach ($condition as $tag_id) {
                    $count .= ' or tags_id like "%' . $tag_id . '%"';
                    $query .= ' or tags_id like "%' . $tag_id . '%"';
                }
                $query .= ' order by id desc limit 6';
                $query = str_replace("  or ", "", $query);
                $query = $db->query($query);
                $result = $query->getResultArray();
                $count = str_replace("  or ", "", $count);
                $count = $db->query($count);
                $count = $count->getResultArray();
                break;
            case "page":
                $result  =  $db->query('select  ID,  title,  price,  image,
created_at FROM posts order by id desc limit 6 offset ' . $condition[1]*6);
                $result = $result->getResultArray();
                $count = $db->query('SELECT count(id) FROM posts');
                $count = $count->getResultArray();
                break;
            default:
            return(view("/pages/stinkystinky.php"));
        }

        $data["posts"] = '';

        if ($result) {                                          //izveido
html ar kuru rādīs visus rakstus
            $session = session();
            foreach ($result as $post) {
                $nameArray = json_decode($post['image'], true);

                $data["posts"] .= '
                <div id="' . $post['ID'] . '"  class="post col-3 p-1 container
row border border-black m-1 p-0 rounded">
                    <img   class="col   p-0"   src="../uploads/avatar/'   .
$nameArray[0]['filename'] . '">
```

```
                    <hr class="d-none d-md-block mt-1 mb-0">
```

***PostController.php* pielikuma turpinajums**

```
                      <div class=""> ' . $post['title'] . ' </div>
                      <div class=""> ' . $post['price'] . '&#8364 </div>
                      <div class="text-secondary" id="date">' . date('d F Y',
strtotime($post['created_at'])) . '</div>
                </div>';
            }
            $data['header'] = view('templates/header');
            $data['footer'] = view('templates/footer');
            return (view("/pages/home", $data));
            /* return $this->response->setJSON([
                'error' => false,
                'posts' => $data,
                'count' => $count[0]["count(id)"]

            ]); */
        } else {
            $data['header'] = view('templates/header');
            $data['footer'] = view('templates/footer');
            $data["posts"] = '<div class="text-secondary text-center fw-bold
my-5">Nav tādi raksti atrasti!</div>';
            return (view("/pages/home", $data));
        }
    }
```

```
public function edit()
    {
        $image_selected = 0;
        $nameArray = [];
        $imagefile = $this->request->getFiles();
        if ($imagefile['titleimage'][0]->isValid() == 1) {
            foreach ($imagefile['titleimage'] as $img) {
//izveido nosaukumu kas satur uz nejaušību ģenerētus burtus un to saglabā
                $randName = $img->getRandomName();
                log_message('debug', 'uploaded images');
                $nameArray[] = [
                    "filename" => $randName,
                ];
            }
            $image_selected = 1;
            log_message('debug', json_encode($nameArray));
        }

        $session = session();

        $db = \Config\Database::connect();
        $query = 'SELECT `edit_post`(' . $this->request->getPost('postID') .
', '
            . $session->get('account_id') . ', '
            . $image_selected . ', "'
            . $db->escape($this->request->getPost("title")) . '", "'
            . $this->request->getPost("price") . '", '
            . $db->escape($this->request->getPost('description')) . ', '
            . $db->escape(json_encode($nameArray)) . ', '
            .  $db->escape($this->request->getPost("fullTags")) . ') as
edit';
        log_message('debug', "full data to send to db" .
json_encode($query));
        $query = $db->query($query);
        $result = $query->getResultArray();



        if ($result[0]["edit"] == "success") {
            if ($imagefile['titleimage'][0]->isValid() == 1) {
                foreach ($imagefile['titleimage'] as $index => $img) {
//izveido nosaukumu kas satur uz nejaušību ģenerētus burtus un to saglabā
                    $img->move('uploads/avatar',
$nameArray[$index]['filename']);
                    log_message('debug', 'file to save' .
$nameArray[$index]['filename']);
                }


                helper('filesystem');
                foreach (json_decode($this->request->getPost("oldimages"))
as $img) {
                    log_message('debug', "removing file - " . $img);
                    try {
                        unlink(APPPATH . "/../public/uploads/avatar/" .
$img);
                    } catch (Exception) {
                        log_message('debug', "file already deleted");
```

***PostController.php* pielikuma turpinājums**

```
                    }
                }
            }
        }

        return $this->response->setJSON([
            'error' => false,
            'message' => 'Successfully added new post!'
        ]);
    }
```

9. Pielikums. ***PostController.php***

```
    public function delete($post_id)
    {                                   //izdzēš izvēlēto post
        $session = session();
        $db = \Config\Database::connect();
        $query = $db->query('SELECT `post_deletion` ("' . $post_id . '", "'
. $session->get('account_id') . '") as removed');
        $result = $query->getResultArray();
        $result = $result[0]["removed"];
        $replace = ['"', "[", "]", "{", "}", "filename:"];
        $result = str_replace($replace, '', $result);
        $result = explode(",", $result);
        log_message('debug', json_encode($result));
        if ($result[0] == "success") {
            array_shift($result);
            helper('filesystem');
            foreach ($result as $img) {
                log_message('debug', "removing file - " . $img);
                try {
                    unlink(APPPATH . "/../public/uploads/avatar/" . $img);
                } catch (Exception) {
                    log_message('debug', "file already deleted");
                }
            }
            return $this->response->setJSON([
                'error' => false,
                'message' => 'success'
            ]);
        } else {
            return $this->response->setJSON([
                'error' => false,
                'message' => 'failed'
            ]);
        }
    }
```

```
public function add($contents)
    {
        $session = session();
        $contents = explode(",", $contents);
        $post_id =  $contents[0];
        $comment = $contents[1];
        $commenter_id = $session->get("account_id");

        $db = \Config\Database::connect();
        $query = $db->query('SELECT `create_comment`(' . $post_id . ', ' .
$db->escape($comment) . ', ' . $commenter_id . ') as comment');
        $result = $query->getResultArray();
        $result = explode(",", $result[0]["comment"]);
        date_default_timezone_set('Europe/Riga');

        $data = '<div id="' . $result[1] . '_comment" class="col-12 row my-
1 mx-0 justify-content-evenly border border-black rounded">
            <div class="row col-1 justify-content-evenly">
            <div class="col-auto p-0">
            <img    class="mt-1"    style="height:    65px"    src="'    .
base_url('uploads/avatar/' . $session->get("pfp")) . '">
            </div>
            </div>

            <div class="col-11 row justify-content-evenly">
            <div class="col-11 row my-1 mb-0 justify-content-between"><span
class="col-4   align-self-center   p-0   text-secondary">'   .   $session-
>get("username") . '    ' . date("Y-m-d H:i:s") . '</span>
            <div  class="col-3 row">
            <button   id="'   .   $result[1]   .   '_edit"     type="button"
class="edit_comment   col       me-1   mb-1   btn   btn-primary   btn-sm"
onclick="commentEdit(this.id)">Rediģēt</button>
            <button   id="'   .   $result[1]   .   '_delete"     type="button"
class="delete_comment col m-0 mb-1 btn btn-danger btn-sm" >Dzēst</button>
            </div>
            </div>
<hr class="d-none d-md-block m-0 mb-1">
            <div id="' . $result[1] . '_comment_text" class="col-11 mb-1">'
. $comment . '</div>
            </div>
            </div>';

        return $this->response->setJSON([
            'error' => false,
            'message' => $data,
        ]);
    }
```

```
public function fetch($contents)
    {
        $data = '';
        $db = \Config\Database::connect();
        $session = session();
        $contents = explode(",", $contents);
        if ($contents[1] == "0") {
            $last_id = 0;
        } else {
            $last_id = $contents[1];
        }
        for ($i = 0; $i < 2; $i++) {
            try {
                $query    =    $db->query('select    `fetch_comments`('    .
$contents[0] . ', ' . $last_id . ') as comment');
                $result = $query->getResultArray();
                $result = explode(",", $result[0]["comment"]);
                $editing = "";
                if ($result[0] == "success") {
                    $last_id = $result[4];
                    $id = $session->get('account_id');
                    if ($id == $result[6] or $id == 10 or $id == 11) {
                        $editing = '<div  class="col-3 row">
            <button   id="'   .   $last_id   .   '_edit"      type="button"
class="edit_comment    col    me-1    mb-1    btn    btn-primary    btn-sm"
onclick="commentEdit(this.id)">Rediģēt</button>
            <button   id="'   .   $last_id   .   '_delete"      type="button"
class="delete_comment col mb-1 btn btn-danger btn-sm" >Dzēst</button>
            </div>';
                    }
                }
                $data .= '<div id="' . $last_id . "_comment" . '" class="col-
12 row mx-0 my-1 justify-content-evenly border border-black rounded">
            <div class="col-1 row justify-content-evenly ">
            <div class="col-auto p-0">
            <img    class="mt-1"    style="height:    65px"    src="'    .
base_url('uploads/avatar/' . $result[1]) . '">
            </div>
            </div>

            <div class="col-11 row justify-content-evenly">
            <div class="col-11 row my-1 mb-0 justify-content-between"><span
class="col-4 align-self-center p-0 text-secondary">' . $result[2] . '    ' .
$result[5] . '</span>
            ' . $editing . '

            </div>
                <hr class="d-none d-md-block m-0 mb-1">
            <div id="' . $last_id . '_comment_text" class="col-11 mb-1 ">' .
$result[3] . '</div>
            </div>
            </div>';
            } catch (Exception) {
                log_message('debug', "man im dead -PHP");
                break;
            }
        }
        return $this->response->setJSON([
```

```
            'error' => false,
```

***CommentController.php* pielikuma turpinajums**

```
            'message' => $data,
            'id' => $last_id
        ]);
    }
```

```
public function delete($comment_id)
    {
        $session = session();
        $db = \Config\Database::connect();
        $query = $db->query('SELECT `delete_comment`(' . $comment_id . ', '
. $session->get("account_id") . ') as comment');
        $result = $query->getResultArray();
        if ($result[0]["comment"] = "success") {
            return $this->response->setJSON([
                'error' => false,
                'message' => $result
            ]);
        }
    }
```

```
public function edit($comment)
    {
        $session = session();
        $db = \Config\Database::connect();
        $comment = explode("_", $comment);
        $query = $db->query('SELECT `edit_comment` (' . $comment[0] . ', ' .
$session->get("account_id") . ', ' . $db->escape($comment[1]) . ') as
comment');
        $result = $query->getResultArray();
        log_message('debug', json_encode($result));
        if ($result[0]["comment"] = "success") {
            return $this->response->setJSON([
                'error' => false,
                'message' => "success"
            ]);
        }
    }
```

```
    public function post($post_id)
    {
        $data['header'] = view('templates/header');
        $data['footer'] = view('templates/footer');

        $db = \Config\Database::connect();
        $query = $db->query("select * from posts where ID=" . $post_id . "");
        $posts = $query->getResultArray();

        $posts[0]["id"] = $post_id;

        $replace = ['"', "[", "]", "{", "}", "filename:", "filetitle:"];
        $posts[0]["image"] = str_replace($replace, '', $posts[0]["image"]);
//removes all the unnesecary stuff form the filename
        $posts[0]["image"] = explode(",", $posts[0]["image"]);

        $replace = ['"', '[', ']'];
        $posts[0]["tags_id"]       =       str_replace($replace,       '',
$posts[0]["tags_id"]);
        if ($posts[0]["tags_id"] != "") {
            $tags_id = explode(",", $posts[0]["tags_id"]);
            $posts[0]["tags_id"] = [];
            $posts[0]["tags"] = [];
            log_message('debug',        "aaaaaaaaaaaaaaaaaaaaaaa"        .
json_encode($posts[0]["tags_id"]));

            foreach ($tags_id as $tag_id) {
                array_push($posts[0]["tags_id"], $tag_id);
                $query = $db->query("select tags from tags where ID=" .
$tag_id . "");
                array_push($posts[0]["tags"],                          $query-
>getResultArray()[0]["tags"]);
            }
        } else {
            $posts[0]["tags"] = [];
            array_push($posts[0]["tags"], "");
            $posts[0]["tags_id"] = [];
            array_push($posts[0]["tags_id"], "");
        }


        $query = $db->query("select username, image from account where
id=(select account_id from posts where id=" . $post_id . ")");
        $account = $query->getResultArray();

        $data['post_data'] = $posts;
        $data['account_info'] = $account;
        return view('pages/post', $data);
    }
```

```
    public function view($page)
    {
        if (! is_file(APPPATH . 'Views/pages/' . $page . '.php')) {
            // Whoops, we don't have a page for that!
            throw new PageNotFoundException("YOU SUCK WAAAH WAAH \n'" . $page
. "'   doesn't exist you idiot");
        }
        $data['header'] = view('templates/header');
        $data['footer'] = view('templates/footer');


        //timeout stuff
        $session = session();
        if (null != $session->get('SessionStart')) { //controls user timeout
            if ((time() - $session->get('SessionStart')) > 600) { //set to
timeout after 10 minutes
                $session->destroy();
               return view('pages/' . 'login', $data);
            } else {
                $session->set('SessionStart', time());
                return view('pages/' . $page, $data);
            }
        } else {
            return view('pages/' . $page, $data);
         }
    }
```

```
$("#post_form").submit(function(e) { // ar ajax request saglabā form
        e.preventDefault();
        const form = document.getElementById("post_form");
        const formData = new FormData(this);
        if (!this.checkValidity()) {
          e.preventDefault();
          $(this).addClass('was-validated');
        } else {
          $("#add_post_btn").text("Adding...");
          $.ajax({
            url: '<?= base_url('post/add') ?>',
            method: 'post',
            data: formData,
            contentType: false,
            cache: false,
            processData: false,
            dataType: 'json',
            success: function(response) {
              var $tagselect = $("#tagselect").select2({
                placeholder: 'izvēlieties tags',
                dropdownParent: $('#createModal'),
              });
              $tagselect.val(null).trigger("change");
              location.reload();

            }
          });
        }
      });
```

```
$(document).delegate('#tag_filter_confirm', 'click', function() {
        var condition = "tags"
        var selected_tags = "";
        document.querySelectorAll('#tag_button.btn.btn-
primary.active').forEach(buttonElement => {
          selected_tags += "-" + buttonElement.value;
          bootstrap.Button.getOrCreateInstance(buttonElement).toggle();
        })
        condition += selected_tags;
        if (condition == "tags") {
          condition = "none"
        };
        console.log(condition);
        window.location.assign("?condition1=" + condition);
      });
```

```
$(function() {
      $("#logout-confirm").on("click", function(e) {
        e.preventDefault();
        var status = "log out"
        $.ajax({
          url: '<?= base_url('account/logout/') ?>' + status,
          method: 'post',
          success: function(response) {
            console.log(response.message);
            if (response.message == "loged out successfully") {
              window.location.assign("/home");
            }
          }
        });
      })
      $("#delete-account").on("click", function(e) {
        e.preventDefault();
        var status = "delete account";
        var password = document.getElementById("passwordConfirm").value;
        $.ajax({
          url: '<?= base_url('account/delete/') ?>' + password,
          method: 'post',
          success: function(response) {
            console.log(response.message);
            if (response.message == "account deleted succesfully") {
              window.location.assign("/home");
            } else {
              console.log(response.message);
            }
          }
        });
      })
```

```
var selDiv = "";
      var storedFiles = [];
      $(document).ready(function() {
        $("#file").on("change", handleFileSelect);
        selDiv = $("#selectedImage");
      });

      function handleFileSelect(e) {
        var files = e.target.files;
        var filesArr = Array.prototype.slice.call(files);
        filesArr.forEach(function(f) {
          if (!f.type.match("image.*")) {
            return;
          }
          storedFiles.push(f);

          var reader = new FileReader();
          reader.onload = function(e) {
            var html =
              '<img id="image" name="image" src="' +
              e.target.result +
              "\" data-file='" +
              f.name +
              "'    class='avatar    rounded    lg'    alt='Category    Image'
height='120px' width='120px'>";
            selDiv.html(html);
          };
          reader.readAsDataURL(f);
        });
      }
```

```
$.ajax({
          url: '<?= base_url('account/fetch') ?>',
          method: 'post',
          success: function(response) {
            var response = response.message;
            if (response) {
              document.getElementById("email").value = response["email"];
              document.getElementById("username").value                    =
response["username"];
            }
          }
        });
      }
```

```
$("#editForm").submit(function(e) {
        e.preventDefault();
        var a = email();
        var s = username();
        var f = password();
        if (a && s && f == "correct") {
          const formData = new FormData(this);
          $.ajax({
            url: '<?= base_url('account/edit') ?>',
            method: 'post',
            data: formData,
            contentType: false,
            cache: false,
            processData: false,
            dataType: 'json',
            success: function(response) {
              if (response.message == "success") {
                location.reload();
              } else {
                document.getElementById("mail-warn").setAttribute("style",
"display:block");
                document.getElementById("mail-warn2").setAttribute("style",
"display:block");
              }
            }
          });
        }
      })
```

```
$("#login-confirm").on("click", function(e) {
        e.preventDefault();
          const login = [];
          login[0] = document.getElementById("email").value;
          login[1] = document.getElementById("password").value;

          $.ajax({
            url: '<?= base_url('account/login/') ?>' + login,
            method: 'post',
            success: function(response) {
              console.log(response.message);
              if (response.message == "loged in successfully") {
                window.location.assign("/home");
              } else {
                document.getElementById("backdrop").setAttribute("style",
"position:fixed; width:100%; height:100%; top:0; left:0; padding:0; margin:0;
display:none")
                $("#backdrop").show();
                $("#alert").show();
              }
            }
          });
      })
```

```
function fetchAllTags() { //ar ajax request parāda visus saglabātos tags
                $.ajax({
                    url: '<?= base_url('post/fetchTags') ?>',
                    method: 'get',
                    success: function(response) {
                        $("#tagselect").html(response.postTags);
                    }
                });
            }
            $(document).delegate('#post_comment',  'click',  function(e)  {
//ar ajax saglabā komentārus
                e.preventDefault();
                var post_id = document.getElementsByClassName("post_id");
                var comment = document.getElementById("comment").value;
                var contents = post_id[0].id + "," + comment;
                if (comment != "") {
                    $.ajax({
                        url: '<?= base_url('comment/add/') ?>' + contents,
                        method: 'post',
                        success: function(response) {

$("#comment_container").prepend(response.message);
                            document.getElementById("comment").value = "";
                        }
                    });
                }
            });
```

```
function fetchComments() {
                var post_id = document.getElementsByClassName("post_id");
                var comment_id = "0";
                contents = post_id[0].id + "," + comment_id;
                $.ajax({
                    url: '<?= base_url('comment/fetch/') ?>/' + contents,
                    method: 'get',
                    success: function(response) {
                        last_comment_id = response.id;
                        if (response.message != "") {
                            $("#comment_container").html(response.message);
                        }
                    }
                });
            }
            $(document).delegate('#show_comments',  'click',  function(e)  {
//ar ajax request parāda komentārus
                var post_id = document.getElementsByClassName("post_id");
                var comment_id = "1";
                contents = post_id[0].id + "," + last_comment_id;
                $.ajax({
                    url: '<?= base_url('comment/fetch/') ?>/' + contents,
                    method: 'get',
                    success: function(response) {
                        last_comment_id = response.id

$("#comment_container").add(response.message).appendTo(document.getElementB
yId("comment_container"));
                    }
                });
            })
```

```
$("#post_form").submit(function(e) {
                e.preventDefault();
                const form = document.getElementById("post_form");
                const formData = new FormData(this);
                $.ajax({
                    url: '<?= base_url('post/edit') ?>',
                    method: 'post',
                    data: formData,
                    contentType: false,
                    cache: false,
                    processData: false,
                    dataType: 'json',
                    success: function(response) {
                        location.reload();
                    }
                });
            });
```

```
$(document).delegate('.post_delete_button', 'click', function(e) { //ar ajax
request parāda komentārus
                var post_id = document.getElementsByClassName("post_id");
                post_id = post_id[0].id;
                Swal.fire({
                    title: 'Vai jūs esat pārliecināti?',
                    text: "Jūs šo nevarēsiet atsaukt!",
                    icon: 'warning',
                    showCancelButton: true,
                    confirmButtonColor: '#3085d6',
                    cancelButtonColor: '#d33',
                    cancelButtonText: 'Atcelt',
                    confirmButtonText: 'Piekrītu'
                }).then((result) => {
                    if (result.isConfirmed) {
                        $.ajax({
                            url:   '<?=   base_url('post/delete/')   ?>/'   +
post_id,
                            method: 'get',
                            success: function(response) {
                                if (response.message == "success") {
                                    window.location.assign("/home");
                                }
                            }
                        });
                    }
                })
            })
```

```
$(document).delegate('.delete_comment', 'click', function(e) { //ar ajax
request parāda komentārus
                comment_id = this.id,
                    comment_id = comment_id.split("_")
                comment_id = comment_id[0]
                Swal.fire({
                    title: 'Vai jūs esat pārliecināti?',
                    text: "Jūs šo nevarēsiet atsaukt!",
                    icon: 'warning',
                    showCancelButton: true,
                    confirmButtonColor: '#3085d6',
                    cancelButtonColor: '#d33',
                    cancelButtonText: 'Atcelt',
                    confirmButtonText: 'Piekrītu'
                }).then((result) => {
                    if (result.isConfirmed) {
                        $.ajax({
                            url: '<?= base_url('comment/delete/') ?>' +
comment_id,
                            method: 'post',
                            success: function(response) {
                                if (response.message[0]["comment"] ==
"success") {
                                    document.getElementById(comment_id +
"_comment").remove();
                                }
                            }
                        });
                    }
                })
            })
```

```
$(document).delegate('.edit_comment_confirm', 'click', function(e) { //ar
ajax request parāda komentārus
                var comment_id = this.id;
                comment_id = comment_id.split("_")
                comment_id = comment_id[0]
                var   comment   =   document.getElementById(comment_id   +
"_comment_text").value;
                full_comment = comment_id + "_" + comment;
                $.ajax({
                    url: '<?= base_url('comment/edit/') ?>' + full_comment,
                    method: 'post',
                    success: function(response) {
                        if (response.message == "success") {
                            const input = document.createElement("div");
                            input.setAttribute("id",      comment_id      +
"_comment_text");
                            input.setAttribute("class", "col-11 mb-1 border
border-black rounded");
                            input.setAttribute("value", comment);
                            input.innerText = comment;
                            document.getElementById(comment_id            +
"_comment_text").replaceWith(input);

                            const                deletebtn                =
document.createElement("button");
                            deletebtn.setAttribute("id",    comment_id    +
"_delete");
                            deletebtn.setAttribute("class",  "delete_comment
col m-1 btn btn-primary btn-sm");
                            deletebtn.innerText = "Dzēst";
                            document.getElementById(comment_id            +
"_cancel").replaceWith(deletebtn);

                            const edit = document.createElement("button");
                            edit.setAttribute("id", comment_id + "_edit");
                            edit.setAttribute("class", "edit_comment col m-
1 btn btn-primary btn-sm ");
                            edit.setAttribute("onclick",
"commentEdit(this.id)")
                            edit.innerText = "edit";
                            document.getElementById(comment_id            +
"_edit").replaceWith(edit);
                        }
                    }
                });

            })
```

```
$("#signup_form").submit(function(e) {
                e.preventDefault();
                var a = email();
                var s = username();
                var f = password();
                if (a && s && f == "correct") {
                    const formData = new FormData(this);
                    $.ajax({
                        url: '<?= base_url('account/signup') ?>',
                        method: 'post',
                        data: formData,
                        contentType: false,
                        cache: false,
                        processData: false,
                        dataType: 'json',
                        success: function(response) {
                            if (response.message[1] == "account_created") {
                                window.location.assign("/home");
                            }else{
                                document.getElementById("mail-
warn").setAttribute("style", "display:block");
                                document.getElementById("mail-
warn2").setAttribute("style", "display:block");
                            }
                        }
                    });
                }
            })
```

```
let selectedFiles = [];
function displaySelectedFiles(input) {
  //parāda izvēlētās bildes formā ar checkbox pie tām
  const fileListDiv = document.getElementById("fileList");
  fileListDiv.innerHTML = "";
  selectedFiles = [];
  if (input.files.length > 0) {
    selectedFiles = Array.from(input.files);
    const list = document.createElement("ul");

    for (let i = 0; i < selectedFiles.length; i++) {
      const file = selectedFiles[i];
      const listItem = document.createElement("li");
      listItem.style.display = "flex";
      listItem.style.alignItems = "center";

      const img = document.createElement("img");
      img.style.Width = "125px";
      img.style.height = "100px";
      img.style.objectFit = "cover";
      listItem.appendChild(img);

      list.appendChild(listItem);

      const reader = new FileReader();
      reader.onload = function (e) {
        img.src = e.target.result;
      };
      reader.readAsDataURL(file);
    }

    fileListDiv.appendChild(list);
  } else {
    fileListDiv.innerHTML = "No files selected.";
  }
}
```

```
var names = "undefined";
function combineData() {
  fullTags = [];
  $("#tagselect").select2("data").forEach(CombineTags);
  tag = document.createElement("input");
  tag.id = "fullTags";
  tag.name = "fullTags";
  tag.type = "hidden";
  tag.value = JSON.stringify(fullTags);
  document.getElementById("post_form").appendChild(tag);

  var description = "";
  description = quill.getContents();
  desc = document.createElement("input");
  desc.id = "description";
  desc.name = "description";
  desc.type = "hidden";
  desc.value = JSON.stringify(description);
  document.getElementById("post_form").appendChild(desc);
}
```

32. Pielikums. ***js1.js***

```
orig_comment = "";
function commentEdit(comment_id){
  comment_id = comment_id.split("_");
  comment_id = comment_id[0];
  orig_comment                                                              =
document.getElementById(comment_id+"_comment_text").innerText;

  const input = document.createElement("input");
  input.setAttribute("id", comment_id+"_comment_text");
  input.setAttribute("class", "col-11 mb-1");
  input.setAttribute("style", "background-color:#dee2e6");
  input.setAttribute("value",
document.getElementById(comment_id+"_comment_text").innerText);
  input.innerText                                                           =
document.getElementById(comment_id+"_comment_text").innerText;
  document.getElementById(comment_id+"_comment_text").replaceWith(input);
  input.focus();

  const cancel = document.createElement("button");
  cancel.setAttribute("id", comment_id+"_cancel");
  cancel.setAttribute("class", "cancel_comment col m-1 btn btn-primary btn-
sm");
  cancel.setAttribute("onclick", "commentCancel(this.id)");
  cancel.innerText = "Atcelt";
  document.getElementById(comment_id+"_delete").replaceWith(cancel);

  const edit = document.createElement("button");
  edit.setAttribute("id", comment_id+"_edit");
  edit.setAttribute("class", "edit_comment_confirm col m-1 btn btn-primary
btn-sm");
  edit.innerText = "Saglabāt";
  document.getElementById(comment_id+"_edit").replaceWith(edit);

}
```

33. Pielikums. ***js1.js***

```
function commentCancel(comment_id){
  comment_id = comment_id.split("_");
  comment_id = comment_id[0];

    const input = document.createElement("div");
  input.setAttribute("id", comment_id+"_comment_text");
  input.setAttribute("class", "col-11 mb-1");
  input.setAttribute("value", orig_comment);
  input.innerText = orig_comment;
  document.getElementById(comment_id+"_comment_text").replaceWith(input);

const deletebtn = document.createElement("button");
  deletebtn.setAttribute("id", comment_id+"_delete");
  deletebtn.setAttribute("class", "delete_comment col m-1 btn btn-primary
btn-sm");
  deletebtn.innerText = "Dzēst";
  document.getElementById(comment_id+"_cancel").replaceWith(deletebtn);

  const edit = document.createElement("button");
  edit.setAttribute("id", comment_id+"_edit");
  edit.setAttribute("class", "edit_comment col m-1 btn btn-primary btn-sm ");
  edit.setAttribute("onclick", "commentEdit(this.id)")
  edit.innerText = "edit";
  document.getElementById(comment_id+"_edit").replaceWith(edit);

  orig_comment = "";
}
```