**RĪGAS VALSTS TEHNIKUMS**

DATORIKAS NODAĻA

Izglītības programma: Programmēšana

**KVALIFIKĀCIJAS DARBS**

**“Sociālā tīkla sistēma publikāciju un ziņojumu kopīgošanai”**

Paskaidrojošais raksts 94 lpp.

Audzēknis: Ivans Aļeškevičs

Prakses vadītājs: Ilona Demčenko

Nodaļas vadītājs: Normunds Barbāns

**Rīga 2023**

**ANOTĀCIJA**

Kvalifikācijas darba tēma ir sociālā tīkla sistēmas izstrāde publikāciju un ziņojumu kopīgošanai. Šī sistēma nodrošinās lietotājiem iespēju publicēt un kopīgot attēlus un video saturu, sekot citiem lietotājiem un sazināties savā starpā. Sistēmas izstrādei izmantoti sekojoši rīki: Visual Studio Code, MySQL, PHP, Laravel, HTML, CSS, Bootstrap un Vue.js. Šie rīki ir izvēlēti, jo tie piedāvā plašu funkcionalitāti, kas nepieciešama mūsdienīgas un efektīvas sociālā tīkla sistēmas izveidei.

Kvalifikācijas darbs sastāv no vairākām nodaļām, kurās detalizēti aprakstīti visi projekta aspekti. Ievadā sniegts projekta pārskats un mērķi, lai lasītāji varētu saprast darba nozīmīgumu un mērķus. Pirmajā nodaļā, "Uzdevuma nostādne", izklāstīta projekta prasība un izvirzītie mērķi, kas nosaka, kādai jābūt izstrādājamai sistēmai. Otrajā nodaļā, "Prasību specifikācija", aprakstītas sistēmas ieejas un izejas informācijas prasības, kā arī funkcionālās un nefunkcionālās prasības, kas ir būtiskas, lai sistēma darbotos efektīvi un lietotājiem draudzīgi. Trešajā nodaļā sniegts uzdevuma risināšanas līdzekļu izvēles pamatojums, kas paskaidro izmantoto tehnoloģiju izvēli, ņemot vērā to piemērotību un efektivitāti. Ceturtā nodaļa aptver programmatūras produkta modelēšanu un projektēšanu, ietverot sistēmas arhitektūru un ER modeli, kas parāda, kā dažādas sistēmas komponentes ir savstarpēji saistītas. Piektajā nodaļā aprakstītas datu struktūras, kas veido sistēmas pamatu, nodrošinot stabilu un efektīvu datu pārvaldību.

Kvalifikācijas darbs sastāv no 94 lappusēm, 7 tabulām, 42 attēliem un 1 pielikuma. Šī struktūra nodrošina visaptverošu ieskatu sistēmas izstrādes procesā, sākot no sākotnējām prasībām līdz gala rezultātiem, nodrošinot pilnīgu pārskatu par visu izstrādes procesu un sasniegtajiem rezultātiem.

**ANNOTATION**

The topic of this qualification work is the development of a social network system for sharing posts and messages. This system will enable users to publish and share images and video content, follow other users, and communicate with each other. The tools used for system development include Visual Studio Code, MySQL, PHP, Laravel, HTML, CSS, Bootstrap, and Vue.js. These tools were chosen because they offer a wide range of functionalities necessary for creating a modern and efficient social network system.

The qualification work consists of several chapters, each detailing different aspects of the project. The introduction provides an overview of the project and its objectives, allowing readers to understand the importance and goals of the work. The first chapter, "Task Definition," outlines the project's requirements and objectives, specifying what the developed system should achieve. The second chapter, "Requirements Specification," describes the system's input and output information requirements, as well as the functional and non-functional requirements essential for the system to operate effectively and user-friendly. The third chapter presents the rationale for choosing the tools used to solve the task, explaining the selection of technologies based on their suitability and efficiency. The fourth chapter covers the modeling and design of the software product, including the system architecture and ER model, illustrating how various system components are interconnected. The fifth chapter describes the data structures that form the foundation of the system, ensuring stable and efficient data management.

The qualification work consists of 94 pages, 7 tables, 42 images, and 1 appendice. This structure provides a comprehensive insight into the system development process, from initial requirements to final results, offering a complete overview of the entire development process and the outcomes achieved.

**Saturs**

[ievads 5](#_Toc169142595)

[1. UZDEVUMA NOSTĀDNE 6](#_Toc169142596)

[2. PRASĪBU SPECIFIKĀCIJA 8](#_Toc169142597)

[2.1. Ieejas un izejas informācijas apraksts 8](#_Toc169142598)

[2.1.1. Ieejas informācijas apraksts 8](#_Toc169142599)

[2.1.2. Izejas informācijas apraksts 9](#_Toc169142600)

[2.2. Funkcionālās prasības 10](#_Toc169142601)

[2.3. Nefunkcionālās prasības 12](#_Toc169142602)

[3. UZDEVUMA RISINĀŠANAS LĪDZEKĻU IZVĒLES PAMATOJUMS 15](#_Toc169142603)

[4. PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA 17](#_Toc169142604)

[4.1. Sistēmas struktūras modelis 17](#_Toc169142605)

[4.1.1. Sistēmas arhitektūra 17](#_Toc169142606)

[4.1.2. Sistēmas ER modelis 19](#_Toc169142607)

[4.2. Funkcionālais sistēmas modelis 21](#_Toc169142608)

[4.2.1. Datu plūsmu modelis 21](#_Toc169142609)

[5. DATU STRUKTŪRU APRAKSTS 24](#_Toc169142610)

[6. LIETOTĀJA CEĻVEDIS 29](#_Toc169142611)

[6.1. Sistēmas prasības aparatūrai un programmatūrai 29](#_Toc169142612)

[6.2. Sistēmas instalācija un palaišana 30](#_Toc169142613)

[6.3. Programmas apraksts 31](#_Toc169142614)

[6.4. Testa piemērs 46](#_Toc169142615)

[NOBEIGUMS 49](#_Toc169142616)

[INFORMĀCIJAS AVOTI 50](#_Toc169142617)

[PIELIKUMI 51](#_Toc169142618)

[1. pielikums. Programmas pirmteksts 52](#_Toc169142619)

# ievads

Mūsdienās ir ļoti svarīgi izmantot tehnoloģijas, lai palīdzētu cilvēkiem savienoties, dalīties ar savām emocijām un pieredzēm, un uzzināt par notikumiem pasaulē. Sociālās tīmekļu platformas ir kļuvušas svarīga ikdienas dzīves sastāvdaļa. Tāpat ir skaidrs, ka pastāv iespējas inovācijām un jaunām pieejām sociālo tīmekļu platformu izstrādē.

Šajā ievadā tiks apspriesta nepieciešamība un lietderība izstrādātajai tīmekļa vietnei, kura ļauj dalīties ar attēliem, video un stāstiem tiešsaistē, veidojot vizuāli bagātas stāstu kolekcijas. Šis rīks ir būtisks šodienas sabiedrībai, atbalstot saziņu, iedvesmojot un veicinot radošumu.

Šī platforma tiks izstrādāta, ņemot vērā dažādu vecumu cilvēku vajadzības, kuri vēlas dalīties ar ikdienas piedzīvojumiem un svarīgiem brīžiem. Tas ir pielāgojams gan jauniešiem, kuri vēlas izteikt sevi, gan uzņēmumiem, kuri meklē jaunus klientus.

Salīdzinot šo platformu ar esošajiem analogiem tirgū, var izcelt tās elastību, lietotājam draudzīgo saskarni un plašas satura dalīšanas iespējas, kas to atšķir no citiem līdzīgiem produktiem. Tas padara to par vērtīgu risinājumu, kas uzlabos lietotāju digitālo pieredzi.

Biznesa analīze parādīja, ka cilvēki un organizācijas varētu iegūt no šīs platformas, palielinot savu redzamību, piesaistot jaunus sekotājus un sniedzot saturu, kas interesē to mērķauditoriju. Tas var stiprināt gan personīgo zīmolu atpazīstamību, gan uzņēmumu panākumus sociālajos tīklos.

Viens no konkurentiem šai sistēmai ir Instagram, un ir svarīgi izcelt, kā izstrādātā platforma atšķiras labvēlīgā nozīmē. Šī sistēma piedāvās plašākas un uzlabotas satura dalīšanas iespējas, ļaujot lietotājiem izveidot vizuāli bagātas publikāciju kolekcijas, kurās var iekļaut gan attēlus, gan video, gan stāstus. Tāpat platforma būs pielāgojama dažādām vecuma grupām, piedāvājot funkcionalitāti, kas apmierinās gan jauniešu izteikšanās vajadzības, gan uzņēmējdarbības prasības.

Papildus tam, platforma uzsvērs lietotāja draudzīgu saskarni un elastību, nodrošinot lietotājiem vieglu un patīkamu pieredzi. Tas palīdzēs atšķirties no konkurentiem, piedāvājot satura radīšanas un dalīšanas procesu, kas ir intuitīvs un viegli lietojams. Šī pieeja uzsvērs ne tikai saturu, bet arī lietotāju pieredzi, tādējādi piesaistot gan individuālos lietotājus, gan uzņēmumus.

Šīs diferenciācijas faktori ir būtiski, lai pievērstu potenciālo lietotāju uzmanību un izceltu šo jauno sociālo platformu tirgū, veicinot tās lietderību un pievilcību plašākam auditorijas spektram.

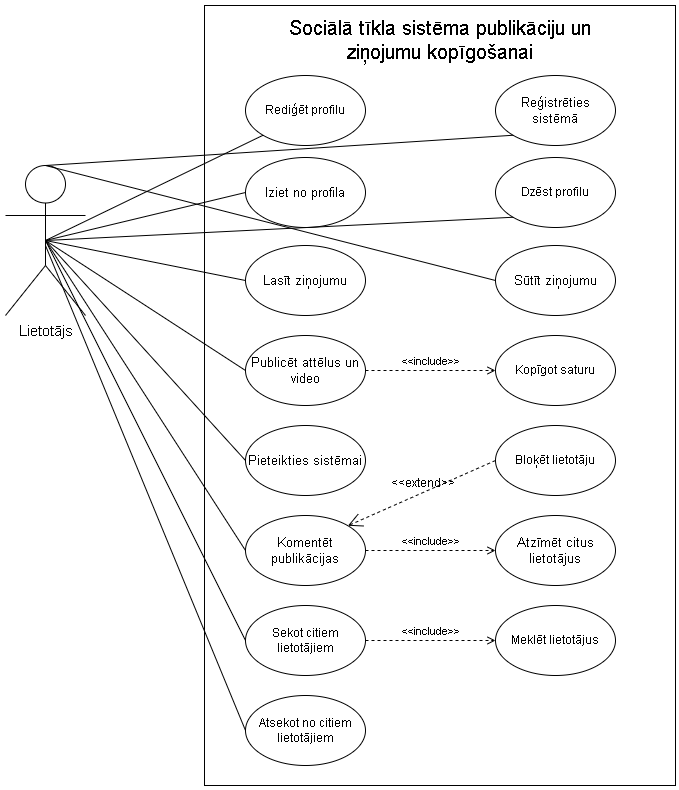
# UZDEVUMA NOSTĀDNE

Kvalifikācijas darba uzdevums ir izveidot sociālā tīkla sistēmu publikāciju un ziņojumu kopīgošanai. Sistēmā nepieciešams realizēt iespēju publicēt attēlus un video saturu, kopīgot tos, sekot citiem lietotājiem, kā arī sazināties ar citiem lietotājiem.

Šī sistēma ir ārkārtīgi aktuāla šodienas digitālajā pasaulē, kur sociālie tīkli ir kļuvuši par ikdienas sastāvdaļu un uzņēmumiem ir nepieciešams efektīvi sasniegt savu mērķauditoriju tiešsaistē. Sociālais tīkls būs vērsts uz plašu auditoriju, kurā būs gan cilvēki, kas meklē vizuālo saturu un vēlas sazināties ar cilvēkiem sociālajos tīklos, gan uzņēmumi, kas vēlas izmantot sociālos tīklus, lai tiešsaistē reklamētu savus produktus un zīmolus. Šī sistēma var piedāvāt jaunas iespējas, lai apmierinātu lietotāju vajadzības pēc augstas kvalitātes vizuālā satura un mārketinga, ņemot vērā pieaugošo pieprasījumu pēc sociālajiem medijiem. Viens no konkurentiem šai sistēmai ir Instagram.

Ir plānotas (skat. 1.1. att.) vairākas funkcijas:

* attēlu un video publicēšana;
* saziņa ar citiem lietotājiem un interakcija (iespēja komentēt publikācijas);
* sekošana citiem lietotājiem;
* meklēšana;
* lietotāju bloķēšana;
* reģistrācija/pieteikšanās.



1.1. att. Lietojumgadījuma diagramma

# PRASĪBU SPECIFIKĀCIJA

## Ieejas un izejas informācijas apraksts

### Ieejas informācijas apraksts

Sistēmā tiks nodrošināta šādas ieejas informācijas apstrāde.

1. Informācija par **lietotājiem** sastāvēs no šādiem datiem.

* Vārds – profila vārds - burtu teksts ar izmēru līdz 30 rakstzīmēm (piem., “Ivans”).
* Lietotājvārds – profila lietotājvārds - burtu teksts ar izmēru līdz 30 rakstzīmēm. (piem., “ivans\_a”).
* E-pasts – profila e-pasts - burtu teksts ar izmēru līdz 255 rakstzīmēm (piem., [ivans.a@example.com](mailto:ivans.a@example.com)).
* Parole – profila parole - burtu teksts ar izmēru līdz 30 rakstzīmēm (piem., “Passw0rd123!”).

2. Informācija par **publikācijām** sastāvēs no šādiem datiem.

* Attēli un video – lietotāji var augšupielādēt attēlus un video ar noteikumiem attiecībā uz faila formātu un lielumu – attēla vai video nosaukums (piem., “fotografija.jpg”, “video.mp4”).
* Apraksts – publikācijas apraksts - burtu teksts ar ierobežojumu līdz 5000 rakstzīmēm, kas pavada publicēto saturu (piem., “Mana pirmā publikācija!”).

3. Informācija par **saziņu** sastāvēs no šādiem datiem.

* Ziņojumi – privāts ziņojums citam lietotājam - lietotāji var nosūtīt ziņojumus citiem lietotājiem, kas ietver tekstu un var būt ierobežots pēc garuma (piem., “Čau, kā Tev iet?”).
* Ziņojuma datums un laiks – laiks un datums, kad tika atsūtīts ziņojums – gan sūtītājs gan saņēmējs redzēs datumu un laiku, kad ziņojums tika atsūtīts (piem., “16.11.2023”).

4. Informācija par **komentāriem** sastāvēs no šadiem datiem.

* Komentāri – publikācijas komentārs - lietotāji var pievienot komentārus citu lietotāju publicētajiem attēliem un videoklipiem (piem., “Skaista fotogrāfija!”).
* Atzīmētie lietotāji - lietotāji var atzīmēt citus lietotājus savos komentāros, lai piesaistītu viņu uzmanību konkrētam saturam – lietotāja lietotājvārds (piem., “@ivans\_a”).

5. Informācija par **profila saturu** sastāvēs no šādiem datiem.

* Profila attēls - lietotāji var augšupielādēt attēlu, kas parādīsies viņu profila sadaļā. Profila attēlam jābūt jpeg vai png formātā – attēla nosaukums (piemērs: profila\_foto.jpg).
* Profila virsraksts - lietotāji var ievadīt virsrakstu, kas parādīsies viņu profila augšpusē - burtu teksts ar ierobežojumu līdz 100 rakstzīmēm (piem., “Lapni lūgti manā profilā!”).
* Profila apraksts - lietotāji var sniegt aprakstu par sevi, kas būs redzams viņu profilā - burtu teksts līdz 500 rakstzīmēm (piem., “Laipni lūgti manā profilā, šeit varēsiet apskatīt manas publikācijas”).
* Profila links - lietotāji var norādīt hipersaiti uz citu vietni vai sociālo tīmekļu profilu, lai dalītos ar papildus informāciju par sevi. Šis lauks ir iespēja, un lietotāji varēs to aizpildīt pēc vēlēšanās (piem., “[https://youtube.com](https://youtube.com/)”, “[http://example.com](http://example.com/)”).

### Izejas informācijas apraksts

1. **Lietotāja profila attēls, virsraksts, apraksts un links** - kad lietotājs veiks izmaiņas savā profilā, jaunais profila attēls, virsraksts, apraksts un links tiks saglabāti un parādīti viņa profila sadaļā. Visu šo informāciju izskats, formāts un atrašanās vieta tiks precizēti, lai nodrošinātu vienotu un pievilcīgu dizainu.
2. **E-pasta paziņojums** - sistēma nosūtīs e-pasta paziņojumu lietotājiem pēc reģistrācijas. E-pasts sastāvēs no sveiciena, neliela teksta ar pateicību no sociālā tīkla un beigu frāzes.
3. **Publikācija, tās apraksts, komentāri un atzīmētie lietotāji** - kad lietotājs izveidos jaunu publikāciju, tās attēls un apraksts tiks parādīti publiski citiem lietotājiem. Papildus tam, kad lietotāji pievienos komentārus šai publikācijai vai atzīmēs citus lietotājus komentāros, šī informācija tiks parādīta attiecīgajā publikācijas sadaļā, nodrošinot saturam pievienoto vērtību un mijiedarbību starp lietotājiem.

## Funkcionālās prasības

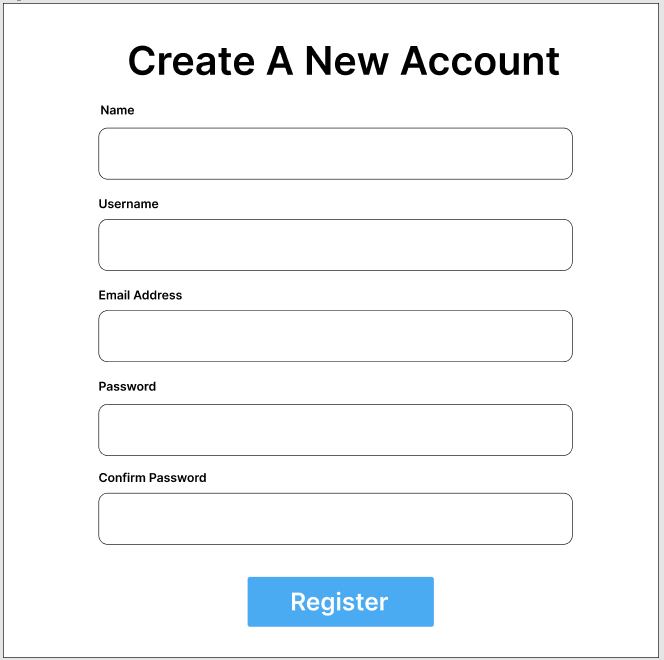
1. Jānodrošina iespēja reģistrēt jaunu lietotāju.
   1. Jāparedz ieejas informācijas par lietotāju (skat. Ieejas informācijas apraksts 1. punkts) ievadīšana un pārbaude uz formāta pareizību.
   2. Ja kāds no obligātajiem laukiem nav ievadīts, sistēma izvada kļūdas paziņojumu, piemēram, "The username field is required".
   3. Pārbaudīt ievadīto lietotājvārdu, lai pārliecinātos, ka tas nesakrīt ar jau eksistējošiem lietotājvārdiem. Ja lietotājvārds jau pastāv, izvadīt paziņojumu “This username is taken” .Pārbaudīt paroli uz pietiekošo drošības pakāpi. Ja tas neatbilst, izvadīt paziņojumu.
   4. Pārbaudīt paroli uz pietiekošo drošības pakāpi. Ja parole neatbilst drošības prasībām, izvadīt paziņojumu, piemēram, “Password must be at least 8 characters long and contain uppercase letters, lowercase letters and numbers”.
2. Jānodrošina lietotāja autorizācija.
   1. Ja ievadītā informācija ir pareiza, lietotājs tiek autentificēts un nonāk savā profilā.
   2. Ja kāds no laukiem nav ievadīts, sistēma izvada paziņojumu, piemēram, “The password field is required”.
   3. Sistēma pārbauda ievadīto lietotājvārdu, e-pastu un paroli. Ja dati neatbilst sistēmas datiem, izvadīt paziņojumu, piemēram, “These credentials do not match our records” un neļauj veikt autorizāciju.
3. Jānodrošina iespēja publicēt attēlus un video.
   1. Lietotāji var publicēt jaunu saturu, iekļaujot attēlus un video materiālus.
   2. Publikācijas virsraksts var būt tukšs.
   3. Ja nav izvēlēts fails publikācijas laukā, sistēma izvada paziņojumu “The media field is required”.
   4. Ja izvēlētais attēla vai video formāts nav atbalstīts, sistēma izvada paziņojumu “The media field must be a file of type: image/jpeg, image/png, video/mp4, video/mpeg, video/quicktime”.
4. Jānodrošina profila rediģēšana.
   1. Lietotāji var mainīt profila attēlu no noklusējuma uz citu attēlu. Atbalstītie formāti: .jpeg vai .png. Attēla izmērs nedrīkst pārsniegt 50MB.
   2. Ja kāds no laukiem ir tukšs un lietotājs mēģina saglabāt izmaiņas, sistēma izvada paziņojumu, piemēram “The media field is required”.
   3. Ja izvēlētais attēla vai video formāts nav atbalstīts, sistēma izvada paziņojumu “The media field must be a file of type: image/jpeg, image/png, video/mp4, video/mpeg, video/quicktime”.
   4. Lietotāji var labot savu profila virsrakstu, aprakstu un URL lauku. Poga “Edit Profile” ļauj lietotājam apskatīt iepriekšējos profila datus.
   5. Ja ievadītais URL nav formātā http://example.com vai https://example.com, sistēma izvada paziņojumu "Nederīgs URL formāts".
5. Jānodrošina iespēja kopīgot publikācijas.
   1. Jānodrošina iespēja kopīgot gan savas publikācijas, gan citu lietotāju publikācijas kā privāts ziņojums citiem lietotājiem.
6. Jānodrošina iespēja komentēt publikācijas.
   1. Jānodrošina iespēja lietotājiem komentēt publikācijas.
   2. Lietotāji var pievienot komentārus savām un citu lietotāju publicētajām publikācijām, veicinot diskusiju un mijiedarbību.
7. Jānodrošina iespēja rādīt publikācijas.
   1. Jānodrošina iespēja lietotājiem apskatīt citu lietotāju publicētos attēlus un video.
   2. Publikācijas teksts jāparāda kopā ar attēlu vai video, lai lietotāji varētu saprast publikācijas kontekstu.
   3. Zem katra publicētā satura jāparādās lietotāju komentāriem, lai veicinātu diskusiju un interakciju.
8. Jānodrošina iespēja sūtīt privātos ziņojumus.
   1. Jānodrošina iespēja lietotājiem sūtīt privātos ziņojumus citiem lietotājiem sistēmā, lai veicinātu personīgu saziņu.
9. Jānodrošina iespēja lasīt privātos ziņojumus.
   1. Jānodrošina iespēja lietotājiem skatīt saņemtos privātos ziņojumus no citiem lietotājiem, nodrošinot drošu un efektīvu komunikāciju.
10. Jānodrošina iespēja bloķēt citus lietotājus.
    1. Administrators var bloķēt citus lietotājus, lai nodrošinātu platformas drošību un uzturētu kopienas vadlīnijas.
11. Jānodrošina iespēja parādīt statistiku administratora panelī.
    1. Jānodrošina statistikas rīki administratora panelī, lai pārvaldītu un analizētu sistēmas lietotājus, komentārus, ziņojumus un publikācijas.
    2. Administratoram jābūt iespējai eksportēt statistiku PDF formātā, lai nodrošinātu datu pārskatāmību un pārskatus.
12. Jānodrošina iespēja parādīt lietotāju sarakstu administratora panelī.
    1. Jānodrošina administratoram iespēja apskatīt visu piereģistrēto lietotāju sarakstu.
    2. Administratoram jābūt iespējai eksportēt lietotāju sarakstu PDF formātā, nodrošinot datu pārvaldību un analīzi.
13. Jānodrošina iespēja meklēt citus lietotājus.
    1. Jānodrošina iespēja lietotājiem meklēt citus lietotājus, ievadot lietotājvārdu meklēšanas laukā.
    2. Sistēma parāda atbilstošus profilus vai līdzīgus profilus pēc ievadītā lietotājvārda. Ja profils netiek atrasts, sistēma izvada paziņojumu "Profils netika atrasts".
14. Jānodrošina iespēja iziet no profila.
    1. Jānodrošina iespēja lietotājiem iziet no sava profila, kad viņi vēlas beigt savu sesiju, nodrošinot drošu iziešanu no sistēmas.

## Nefunkcionālās prasības

1. Sistēmas saskarnes valodai ir jābūt latviešu, angļu un krievu valodām.
2. Jānodrošina tīmekļa lietojumprogrammas pielāgošanas ekrāna izmēriem, kas mūsdienās tiek lietoti, lai to varētu izmantot uz dažādiem monitora izmēriem.
3. Dizainam ir jābūt izstrādātam tā, lai tas būtu pievilcīgs un viegli lasāms arī uz mazākiem ekrāniem.
4. Tekstam jābūt Open Sans fontā un logotipa krāsās, lai nodrošinātu vienotu izskatu un lasāmību.
5. Sistēmas darbībai jābūt efektīvai un resursu ziņā efektīvai, lai nodrošinātu labu veiktspēju pat lielos datu apjomos vai lielā lietotāju skaitā.
6. Sistēmā jānodrošina datu drošība un privātums, izmantojot pietiekaami spēcīgaas šifrēšanas metodes un piekļuves kontroles līdzekļus.
7. Sistēmu jāspēj viegli atjaunināt un paplašināt, lai pielāgotu jaunām funkcijām vai izmaiņām vides.
8. Sistēmas darbībai jābūt uzticamai un jānovērš neplānoti traucējumi vai datu zudumi.
9. Sistēma jāpielāgo cilvēkiem ar ierobežotām fiziskajāam vai sensoriskajām spējām, nodrošinot pieejamību cilvēkiem ar dažādām vajadzībām.
10. Lietotāja saskarnei jābūt saderīgai ar populārākajiem interneta pārlūkiem (piemēram, Chrome, Firefox, Safari), nodrošinot vienmērīgu pieredzi visiem lietotājiem.
11. Sistēmai jābūt ātri reaģējošai un jānodrošina īss atbildes laiks, lai lietotājiem būtu ērti lietot platformu bez gaidīšanas.

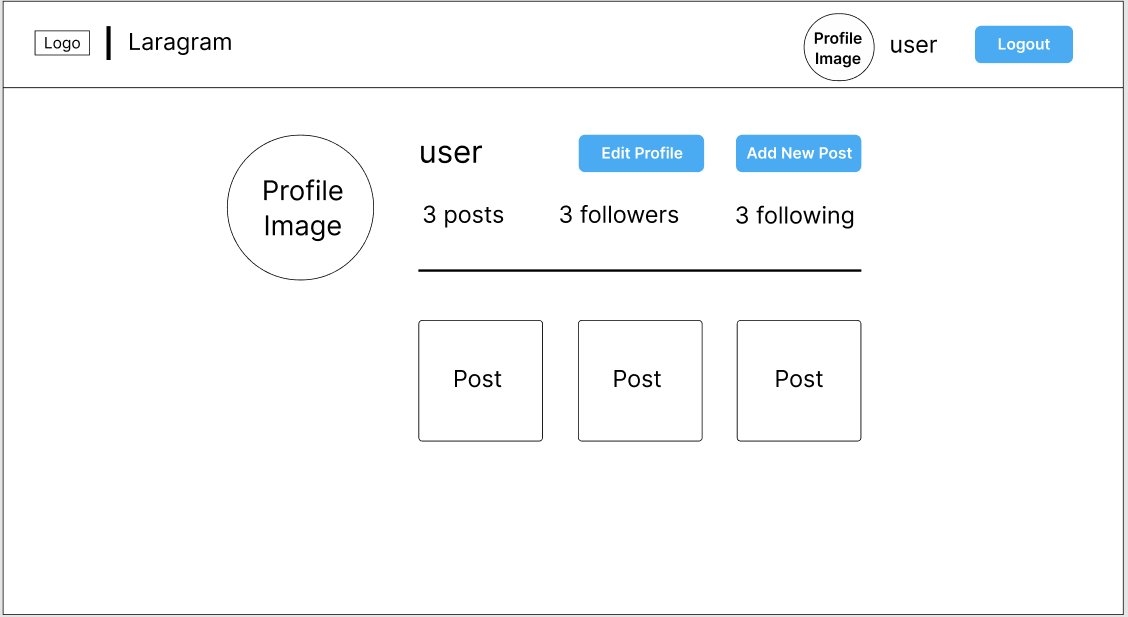
Sistēmas ekrānu skices:

* Sistēmas reģistrācijas skice (skat. 2.1. att)



2.1. att. Reģistrācijas formas ekrāna skice

Šajā skicē tiek attēlots sistēmas interfeiss, kurā lietotāji var reģistrēties. Tajā ir 5 lauki un poga “Register”, kura ļauj reģistrēt jauno lietotāju.

* Sistēmas profila ekrāna skice (skat. 2.2. att.)

2.2. att. Sistēmas profila ekrāna skice

Šajā skicē tiek attēlots sistēmas interfeiss, kurā redzams lietotāja profils, kurš pierakstījās sistēmā. Ir 3 pogas: lai rediģētu profilu, pievienotu jaunu publikāciju un izrakstītos no profila. Kā arī redzams publikāciju, sekotāju un sekošanu skaits.

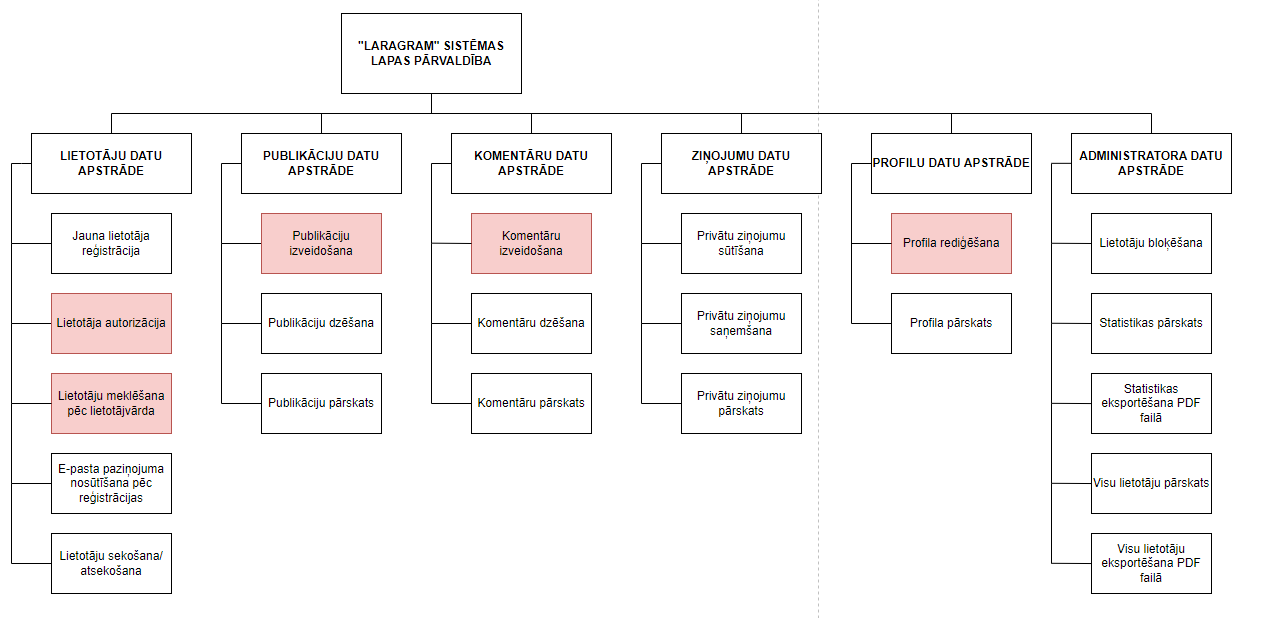
1. **UZDEVUMA RISINĀŠANAS LĪDZEKĻU IZVĒLES PAMATOJUMS**

Sistēma paredzēta plašai pieejamībai un vieglai lietošanai pārlūkprogrammās. Sistēma tiks veidota, izmantojot divu daļu struktūru: lietotāja daļu (frontend) un servera pusi (backend). Lietotāja daļa nodrošina vizuālo interfeisu un lietotāja pieredzi, savukārt servera puse nodrošina datu apstrādi, biznesa loģiku un mijiedarbību ar datu bāzi. Sistēma ir optimizēta un piemērota pārlūkprogrammām, radot lietotājiem ērtu piekļuvi un lietošanas pieredzi tieši caur tīmekli. Šāds lēmums ļauj nodrošināt konsistentu un viegli pielāgojamu interfeisu, kas ir saskanīgs dažādās pārlūkprogrammās, tādējādi sniedzot vienlīdz labu lietojamību neatkarīgi no izmantotās platformas vai ierīces veida.

Izvēlēto līdzekļu pamatojums:

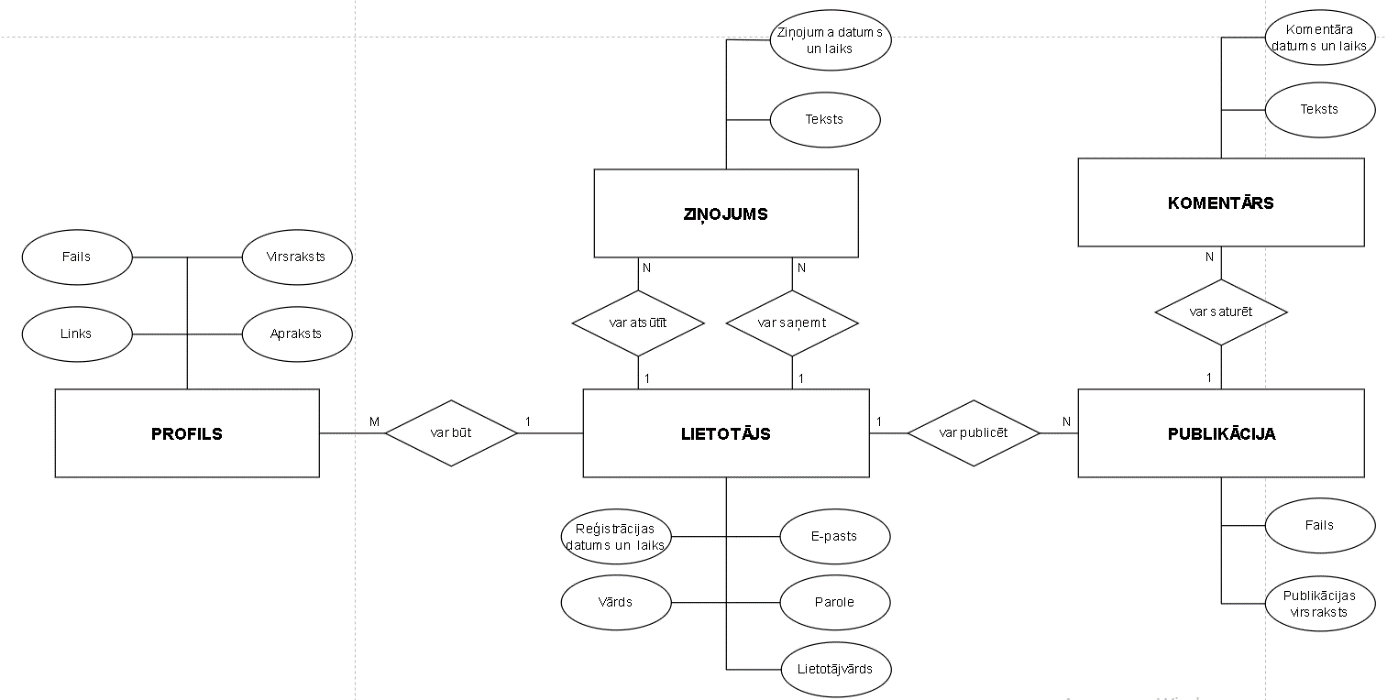
* **Visual Studio Code (versija 1.84.2)** ir populārs koda redaktors un integrēta izstrādes vide (IDE), kas tiek izmantota koda rakstīšanai, labošanai un izstrādei. Tas piedāvā daudzas funkcijas, tai skaitā bagātīgu paplašinājumu klāstu, integrētu versiju vadību, atbalstu dažādām programmēšanas valodām un labu atbalstu krājumiem.
* **MySQL (versija 8.0)** ir atvērtā koda relāciju datu bāzu pārvaldības sistēma, kas piedāvā uzticamu un efektīvu datu uzglabāšanu. MySQL tiek izmantots kā servera puses datu bāze, lai nodrošinātu datu apstrādi, biznesa loģiku un mijiedarbību ar datu bāzi. Šī datu bāzes sistēma ir optimizēta darbam ar lielu datu apjomu, tā piedāvā augstu veiktspēju un drošību, padarot to par piemērotu risinājumu sistēmām, kuras tiek pielāgotas plašai pieejamībai un lietotāju pieredzes uzlabošanai tiešsaistē.
* **PHP jeb Hypertext Preprocessor (versija 8.2.6)** ir PHP ir servera puses skriptu valoda, kas izmantota dinamisku tīmekļa vietņu izstrādē un datu apstrādē. Tā ļauj iegult koda fragmentus HTML failos, ļaujot veidot dinamisku satura, piemēram, lietotāja ievades formas, un mijiedarboties ar datu bāzēm.
* **Laravel (versija 10.44.0)** ir moderns PHP programmēšanas valodas framework, kas piedāvā ērtu veidu, kā izstrādāt efektīvas un strukturētas web lietojumprogrammas. Tas nodrošina iebūvētu atbalstu dažādām funkcijām, piemēram, datu bāzēm, lietotāja autentifikācijai, koda migrācijām un daudz ko citu, paātrinot izstrādes procesu.
* **HTML jeb Hypertext Markup Language (versija 5.3)** ir standarta valoda, ko izmanto, lai veidotu un strukturētu saturu interneta vietnēs. Tā sastāv no dažādām iezīmēm, kas definē, kāds ir konkrēta daļa no teksta vai kāda elementa izskats un funkcionalitāte.
* **CSS jeb Cascading Style Sheets (versija 3)** ir tehnoloģija, kas nodrošina veidu, kā vizuāli noformēt un stilizēt HTML dokumentus. Tā ļauj definēt, kā izskatīsies HTML elementi, nosakot to izmērus, krāsas, fontus un citus stilizācijas parametrus.
* **Bootstrap (versija 5)** ir populāra atvērtā koda HTML, CSS un JavaScript ietvari (framework), kas nodrošina iespēju veikt ātru un vienkāršu tīmekļa lapu un lietotņu izstrādi. Tas piedāvā gatavas stilu klases, komponentes un interaktīvus elementus, kas palīdz veidot responsīvas un modernas tīmekļa lapas, izmantojot vienkāršu un vienotu kodu.
* **Vue.js (versija 3)** ir moderns JavaScript ietvars (framework), kas izstrādāts ar mērķi atvieglot vienādas un elastīgas lietotāja saskarnes (UI) izstrādi tīmekļa lietotnēm. Tas ir lietotājam draudzīgs un viegli iemācāms, un to var integrēt ar citiem tehnoloģiju risinājumiem.

1. **PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA**
   1. **Sistēmas struktūras modelis**
      1. *Sistēmas arhitektūra*

Sistēmai (skat. 4.1. att.) būs piecas apakšistēmas: lietotāju datu apstrāde, publikāciju datu apstrāde, komentāru datu apstrāde, ziņojumu datu apstrāde un profilu date apstrāde.

4.1. att. Funkcionālās dekompozīcijas diagramma

* **Lietotāju modulis.** Nereģistrētiem lietotājiem ir iespēja izveidot jaunu kontu sistēmā, nodrošinot vienkāršu un intuitīvu reģistrācijas procesu. Autorizācijas dati, piemēram, lietotājvārds un parole, sniedz iespēju droši un personificēti piekļūt lietotāja kontam. Sistēma arī nodrošina iespēju meklēt lietotājus pēc lietotājvārda, atvieglojot savstarpējo saziņu un kontaktu uzturēšanu starp lietotājiem. Pēc veiksmīgas reģistrācijas sistēma automātiski nosūta e-pasta pazinājumu, informējot par veiksmīgu reģistrāciju. Turklāt, lietotājiem ir iespēja sekot līdzi vai atsekot citus lietotājus, tā veicinot sociālo mijiedarbību un savstarpēju saziņu.
* **Publikāciju modulis.** Lietotājiem ir iespēja izveidot jaunas publikācijas, kurās tie var dalīties ar attēliem. Pēc publikācijas izveidošanas lietotājiem ir pieejams pārskats par esošajām publikācijām, kur tie var pārskatīt savas publikācijas vai apskatīt citas lietotāju izveidoto saturu. Turklāt, sistēma nodrošina iespēju dzēst publikācijas, ja tās vairs nav aktuālas vai nepieciešamas.
* **Komentāru modulis.** Lietotājiem ir iespēja izveidot jaunus komentārus, atstāt savas domas vai atsauksmes par esošo saturu. Pēc komentāru izveidošanas lietotājiem ir pieejams pārskats par esošajiem komentāriem, kur tie var pārskatīt savus komentārus vai apskatīt citu lietotāju komentārus. Turklāt, sistēma nodrošina iespēju dzēst komentārus, ja tie vairs nav aktuāli vai nepieciešami.
* **Ziņojumu modulis.** Lietotājiem ir iespēja sūtīt privātus ziņojumus, nodrošinot privātu un drošu komunikāciju ar citiem lietotājiem. Lietotāji var arī saņemt privātus ziņojumus no citiem lietotājiem, nodrošinot iespēju sazināties un komunicēt individuāli. Turklāt, sistēma nodrošina iespēju pārskatīt esošos privātos ziņojumus, kur lietotāji var pārskatīt un atbildēt uz saņemtajiem ziņojumiem.
* **Profilu modulis.** Lietotājiem ir iespēja rediģēt savu profilu, mainot informāciju, piemēram, virsrakstu, aprakstu, saiti vai profila attēlu. Pēc profila rediģēšanas lietotājiem ir pieejams pārskats par savu profilu, kur tie var pārskatīt un rediģēt savus personiskos datus un publisko informāciju.
* **Administratora modulis.** Lietotāju bloķēšana nodrošina iespēju administratoram bloķēt lietotājus gadījumos, ja tas ir nepieciešams. Statistikas pārskats sniedz iespēju administratoram pārvaldīt un uzraudzīt sistēmas darbību, pamatojoties uz svarīgiem statistikas datiem. Turklāt, ir pieejama funkcija eksportēt statistikas pārskatus PDF formātā, kas nodrošina iespēju lejupielādēt un saglabāt statistikas datus vai veikt turpmāku analīzi vai apstrādi. Ir pieejams visu lietotāju pārskats un to eksportēšana PDF formātā, kas nodrošina iespēju skatīt visu sistēmas lietotāju informāciju un veikt vēlāku analīzi.
  + 1. *Sistēmas ER modelis*

Sistēmas ER-modelis sastāv no 5 entitijām (skat. 4.2. att.), kas ļauj uzglabāt un apstrādāt svarīgu informāciju. Katra entitāte atspoguļo noteiktu datu kategoriju un attiecības starp tām.

4.2. att. Sistēmas ER modelis

Šeit ir apraksts par svarīgākajām entitātēm:

* Lietotājs:

Lietotājs ir galvenā entitija, kas satur informāciju par reģistrētiem platformas lietotājiem. Katram lietotājam ir unikāls lietotājvārds, e-pasts, parole un profilu saistīta informācija – vārds un reģistrācijas laiks un datums.

* Publikācija:

Šī entitija satur informāciju par lietotāju veidotajām publikācijām. Šeit iekļauti dati par tekstu, publicēšanas datumu un laiku un attēla/video URL. Publikācijas entitija ļauj lietotājiem dalīties ar saturu un veidot savas publikācijas.

* Komentārs:

Šī entitija atspoguļo komentārus, ko lietotāji pievieno publikācijām. Tā satur informāciju par tekstu, komentēšanas datumu un laiku. Tas ļauj lietotājiem dalīties ar savām domām un reakcijām uz citu saturu.

* Ziņojums:

Šī entitija nodrošina privātās ziņas starp lietotājiem. To raksturo ziņojuma teksts, nosūtīšanas laiks un datums. Šī entitija nodrošina privātas saziņas iespēju lietotājiem platformā.

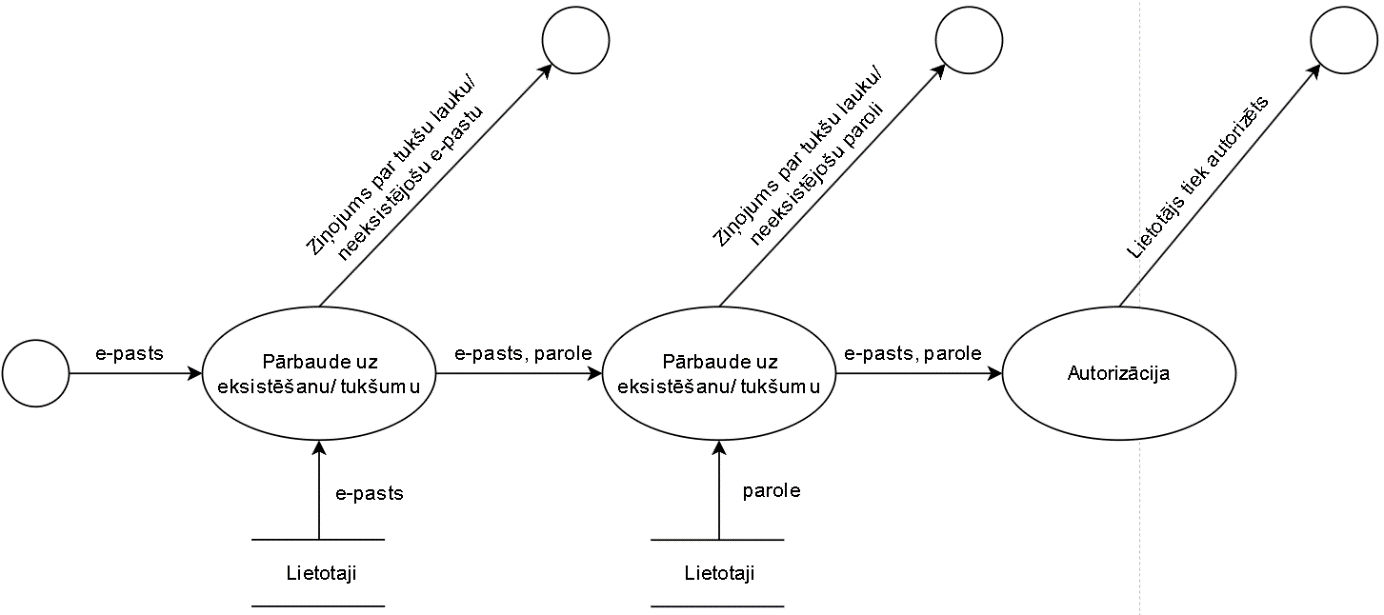
* Profils:

Šī entitija sniedz iespēju lietotājiem personificēt savu klātbūtni platformā.. Šī entitāte spēlē būtisku lomu lietotāja pieredzē, ļaujot tiem izcelt savas personīgās īpašības un dalīties ar papildu informāciju ar kopienas biedriem. To raksturo virsraksts, apraksts, links un fails(attēls).

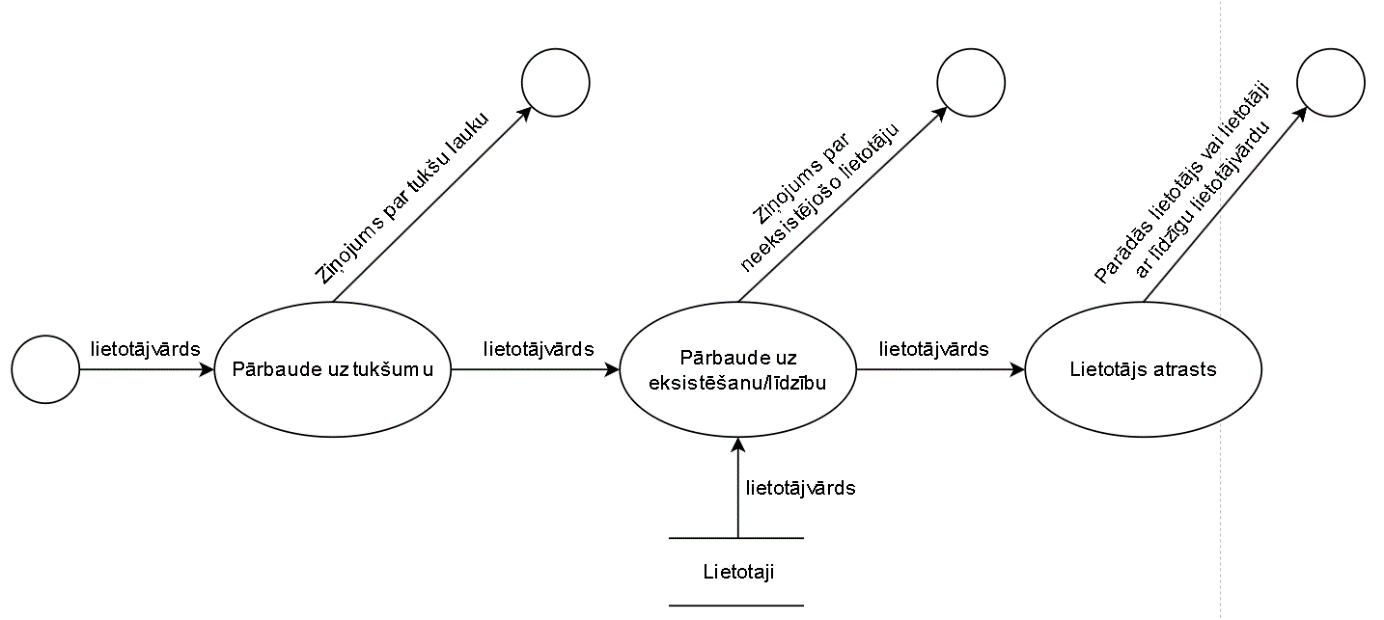
Starp šīm entītijām pastāv arī vairākas attiecības:

* **Lietotājs (1) – (N) Profils.** Katram profilam ir saistīts tieši viens lietotājs, bet katram lietotājam var būt vairāki profili. Šī attiecība nodrošina, ka katram profila ierakstam atbilst tikai viens lietotājs, bet lietotājs var būt saistīts ar vairākiem profilu ierakstiem. Piemēram, viens lietotājs var pārvaldīt personīgo un biznesa kontu, saglabājot atsevišķu informāciju katrā profilā.
* **Lietotājs (1) – (N) Publikācija.** Katrs lietotājs var izveidot vairākas publikācijas, bet katrai publikācijai ir tikai viens saistītais lietotājs. Šī attiecība norāda, ka lietotājs var būt autors vairākām publikācijām, piemēram, vienā dienā var tikt publicēti vairāki ziņojumi vai fotogrāfijas, bet katrai publikācijai ir tikai viens autors.
* **Publikācijas (1) – (N) Komentārs.** Katrai publikācijai (Publikācija) var būt vairāki komentāri (Komentārs), bet katrs komentārs ir saistīts tikai ar vienu publikāciju. Šī attiecība nodrošina iespēju pievienot vairākus komentārus katrai publikācijai, kas veicina aktīvu lietotāju mijiedarbību un diskusiju platformā. Piemēram, vienai publikācijai var būt simtiem komentāru, kas atspoguļo dažādus viedokļus un reakcijas.
* **Lietotājs (1) – (N) Ziņojums (Saņēmējs).** Katrs lietotājs var saņemt vairākus ziņojumus, bet katrs ziņojums ir saistīts tikai ar vienu lietotāju. Šī attiecība norāda, ka lietotājs var saņemt vairākus ziņojumus, bet katrs ziņojums ir adresēts tikai vienam lietotājam.
* **Lietotājs (1) – (N) Ziņojums (Sūtītājs).** Katrs lietotājs var sūtīt vairākus ziņojumus, bet katrs ziņojums ir saistīts tikai ar vienu lietotāju. Šī attiecība norāda, ka lietotājs var saņemt vairākus ziņojumus, bet katrs ziņojums ir adresēts tikai vienam lietotājam.
  1. **Funkcionālais sistēmas modelis**
     1. *Datu plūsmu modelis*

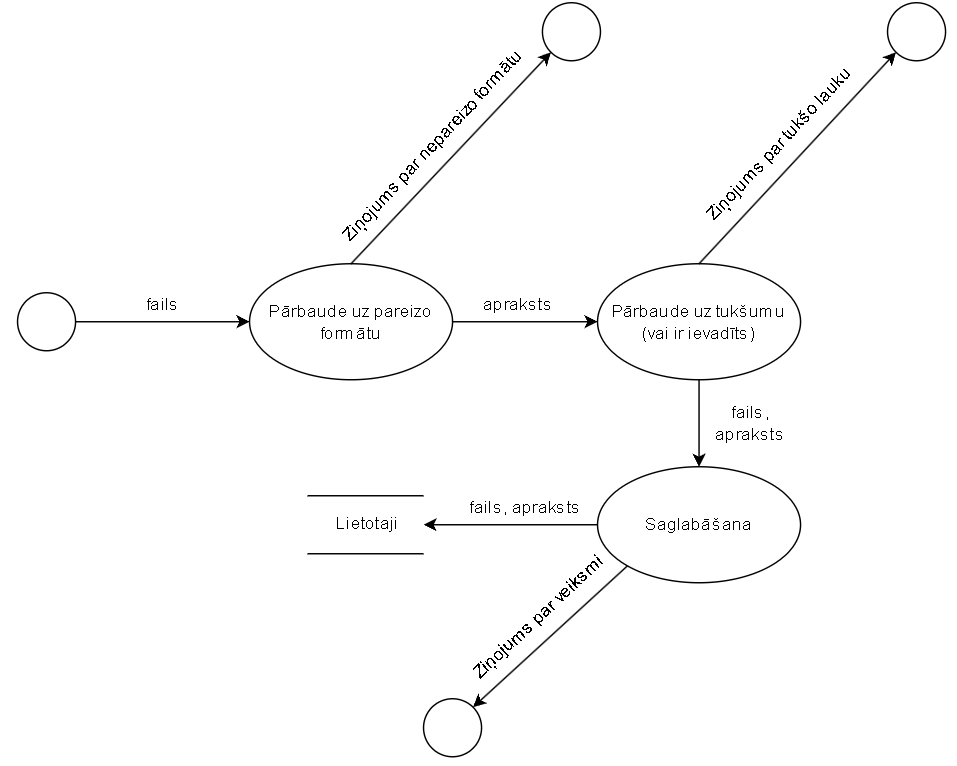
Datu apstrādes procesi:

1. **Lietotāja autorizācija (skat. 4.3. att.):** pārbauda un apstiprina lietotāja identitāti, atļaujot piekļuvi sistēmas funkcijām. Lietotājs sāk autorizācijas procesu, iesniedzot savu lietotājvārdu un paroli. Sistēma pārbauda, vai ievadītie lietotājvārds un parole atbilst sistēmā esošajiem datiem. Ja dati nav pareizi, sistēma paziņo, ka ievadītie dati nesakrīt un dod lietotājam iespēju autorizēties vēlreiz. Ja lauki ir tukši, sistēma par to paziņo. Ja dati ir pareizi, sistēma apstiprina autorizāciju, un lietotājam tiek piešķirts piekļuves līmenis.

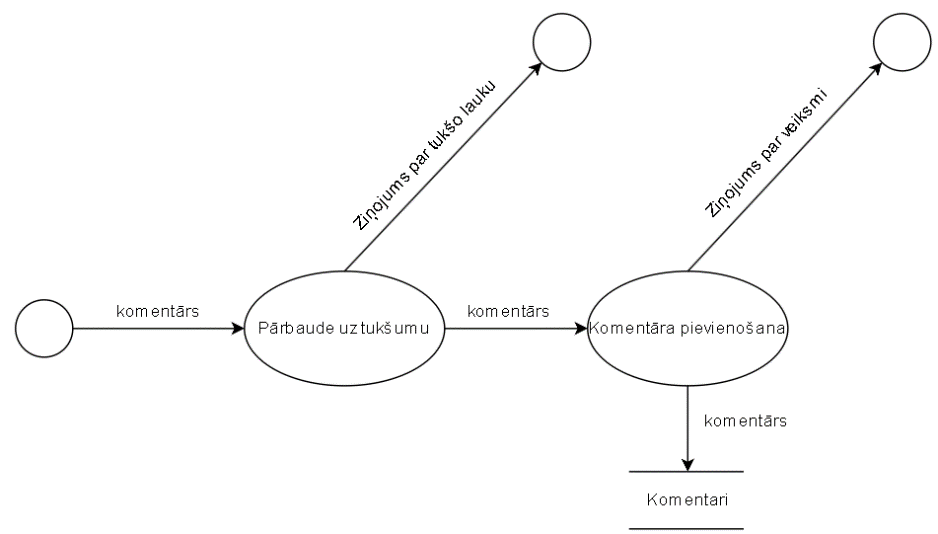
4.3. att. Datu plūsmu diagramma – lietotāja autorizācija

1. **Lietotāju meklēšana pēc lietotājvārda (skat. 4.4. att.):** sistēma atrod un atgriež informāciju par meklētajiem lietotājiem. Lietotājs sāk meklēšanas procesu, ievadot konkrēta lietotāja lietotājvārdu vai daļu no tā. Sistēma pārbauda datu bāzi, lai atrastu lietotāja vārdu, kas atbilst ievadītajam kritērijam. Ja atbilstīgi dati tiek atrasti, sistēma attēlo rezultātus, kas var ietvert atrasto lietotāju sarakstu ar saistītajiem profiliem.Ja lauks ir tukšs, sistēma izvada atbilstošu ziņojumu. Ja lietotājs datu bāzē netiek atrasts, tad sistēma izvada par to paziņojumu, kā arī ja lauks ir tukšs, parādās atbilstošs paziņojums.

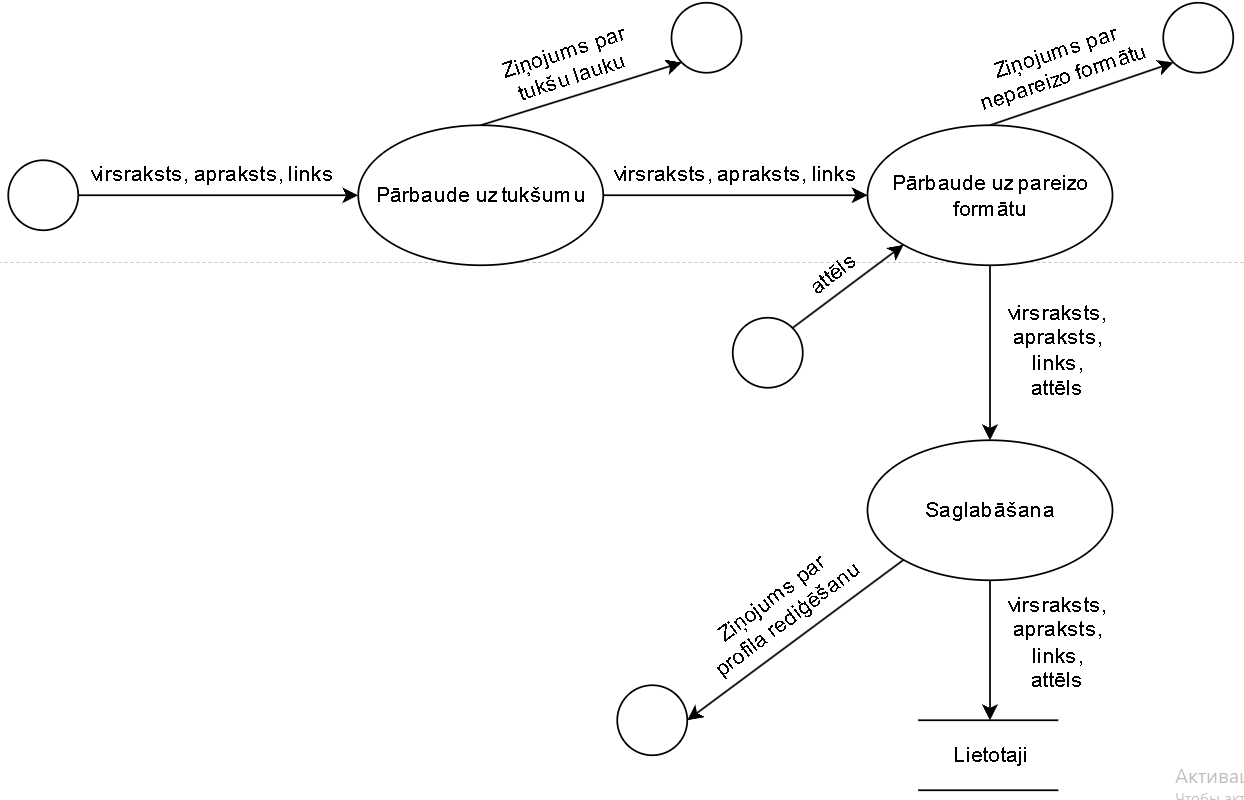
4.4. att. Datu plūsmu diagramma – lietotāju meklēšana pēc lietotājvārda

1. **Publikāciju izveidošana (skat. 4.5. att.):** lietotājiem ir iespēja izveidot jaunas publikācijas, pievienojot tekstu, attēlus vai video. Lietotājs sāk procesu, izvēloties opciju izveidot jaunu publikāciju. Lietotājs izvēlas attēlu vai video un ievada aprakstu. Ja attēla formāts ir pareizs un apraksta lauks nav tukšs, sistēma saglabā jauno publikāciju datu bāzē, piešķirot tai unikālu identifikatoru un informē lietotāju par veiksmīgu publikācijas izveidi.

4.5. att. Datu plūsmu diagramma – publikāciju izveidošana

1. **Komentāru izveidošana (skat. 4.6. att.):** ļauj lietotājiem pievienot komentārus pie esošajām publikācijām. Lietotājs sāk procesu, izvēloties opciju pievienot komentāru pie esošas publikācijas. Lietotājs ievada komentāra tekstu. Sistēma veic datu validāciju, pārbaudot, vai komentāra lauks nav tukšs. Ja lauks nav tukšs, sistēma saglabā jauno komentāru datu.

4.6. att. Datu plūsmu diagramma – komentāru izveidošana

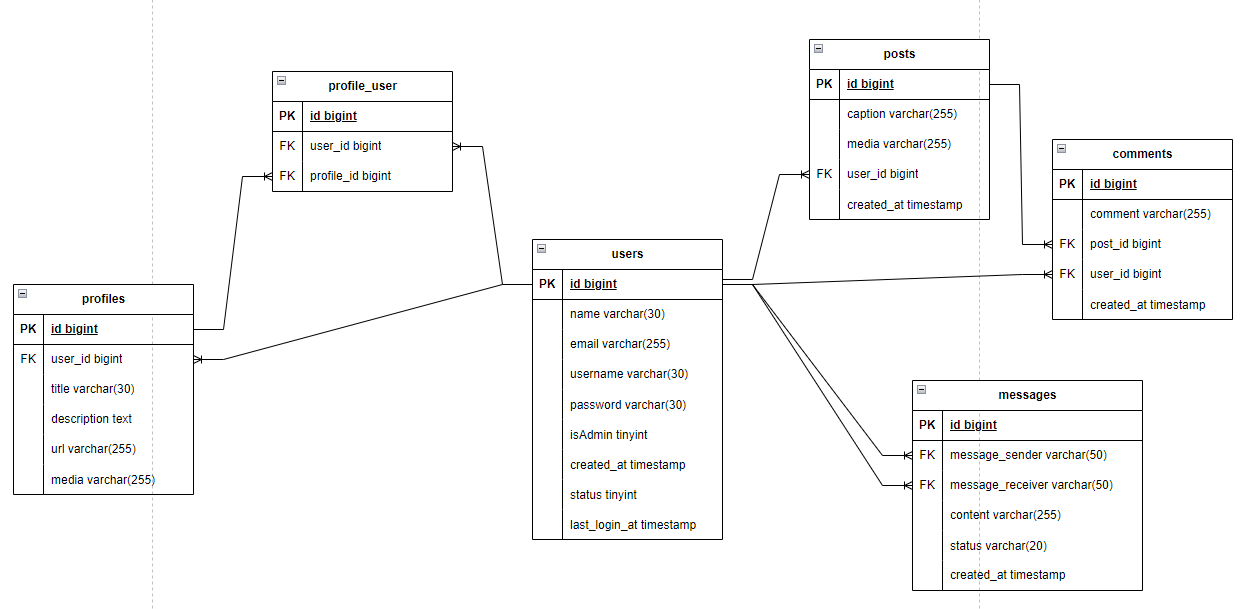
1. **Profila rediģēšana (skat. 4.7. att.):** ļauj lietotājam labot savu profilu. Lietotājs izvēlas opciju rediģēt profilu. Sistēma atgūst esošos profilus un iegūst informāciju par lietotāja profilu, iekļaujot attēlu, virsrakstu, aprakstu un linku. Lietotājs ievada jauno informāciju, piemēram, rediģē profila attēlu, maina virsrakstu, aprakstu vai profilu saiti. Sistēma veic datu validāciju, pārbaudot, vai jaunie dati nav tukši un attēls ir pareizā formātā. Ja dati ir derīgi, sistēma saglabā jauno informāciju datu bāzē, atjauninot lietotāja profilu. Sistēma informē lietotāju par veiksmīgu profilu rediģēšanu.

4.7. att. Datu plūsmu diagramma – profila rediģēšana

1. **DATU STRUKTŪRU APRAKSTS**

Datu bāze sastāv no 6 tabulām (skat. 5.1. att.), kurās ir informācija par lietotājiem, profiliem, publikācijām, komentāriem, ziņojumiem un to saistībām ar lietotājiem.

Tabulu saturs:

1. Tabula **“users”** satur 9 laukus un glabās informāciju par lietotājiem.
2. Tabula **“posts**” satur 4 laukus un glabās informāciju par publikācijām.
3. Tabula **“profiles”** satur 5 laukus un glabās informāciju par profiliem.
4. Tabula **“comments”** satur 4 laukus un glabās informāciju par komentāriem.
5. Tabula **“messages”** satur 6 laukus un glabās informāciju par ziņojumiem.
6. Tabula **“profile\_user”** satur 3 laukus un tieks izmantota kā starptabula starp **“profiles”** un **“users”** tabulām.

5.1. att. Tabulu saišu shēma

Lietotāju tabula satur informāciju par lietotājiem, tostarp e-pastu, vārdu, lietotājvārdu, lietotāja izveidošanas datumu un laiku, statusu, pēdējo autorizācijas laiku, lauku, kurš nosaka vai lietotājam ir administratora tiesības un paroli. Tabulā “profiles” ir informācija par lietotāju profiliem, ietverot virsrakstu, aprakstu, linku un failu(attēlu). Tabula “profile\_user” ir starptabula, kura nodrošina saistību starp lietotājiem un viņu profiliem. Tā satur identifikatoru un divas ārejās atslēgas. Tabulā “posts” ir informācija par lietotāju publikācijām, ietverot aprakstu un failu (attēls/video). Tabula “comments” satur informāciju par lietotāju komentāriem publikācijās. Tabulā “messages” ir informācija par ziņojumiem, ietverot tekstu, ziņojuma saņēmēju un sūtītāju, statusu un ziņojuma datumu un laiku.

Attiecības starp tabulām ir šādas:

* **“users” – “profiles”.** Katram lietotājam var būt viens vai vairāki profili, un katram profilam var būt vairāki saistītie lietotāji. Attiecības ir daudz pret daudziem starp lietotājiem un profiliem.
* **“users” – “posts”.** Katrs lietotājs var veikt vienu vai vairākas publikācijas, bet katrā publikācijā piedalās tikai viens lietotājs. Attiecības ir viens pret daudziem starp lietotājiem un publikācijām.
* **“posts” – “comments”.** Katrai publikācijai var būt vairāki komentāri, bet katrs komentārs ir saistīts tikai ar vienu publikāciju. Attiecības ir viens pret daudziem starp publikācijām un komentāriem.
* **“users” – “comments”.** Katram lietotājam var būt viens vai vairāki komentāri, bet katrs komentārs ir saistīts tikai ar vienu lietotāju. Šīs ir attiecība viens pret daudziem starp lietotājiem un komentāriem.
* **“messages” – “users” (Saņēmējs).** Katrs ziņojums ir saistīts ar vienu lietotāju kā saņēmēju, un katrs lietotājs var saņemt daudz ziņojumu. Attiecības ir viens pret daudziem starp ziņojumiem un lietotājiem, izmantojot lauku **“message\_receiver”** kā ārējo atslēgu, kas norāda uz lietotāju tabulas lauku **„username”**. Tas nozīmē, ka katrs saņēmējs (lietotājs) var būt saistīts ar daudziem ziņojumiem.
* **“messages” – “users” (Sūtītājs).** Katrs ziņojums ir saistīts ar vienu lietotāju kā sūtītāju, un katrs lietotājs var nosūtīt daudz ziņojumu. Attiecības ir viens pret daudziem starp ziņojumiem un lietotājiem, izmantojot lauku **“message\_sender”** kā ārējo atslēgu, kas norāda uz lietotāju tabulas lauku **„username”**. Tas nozīmē, ka katrs sūtītājs (lietotājs) var būt saistīts ar daudziem ziņojumiem.

Šādi organizējot datubāzi, ir novērsts datu dublēšanās un datubāze ir nodrošināta trešajā normālā formā. Tas nozīmē, ka katrā tabulā ir informācija, kas ir loģiski saistīta, un starp atribūtiem katrā tabulā nav pārejas atkarību.

5.1. tabula

Tabulas **„users”** struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | - | Lietotāja kārtas numurs |
| 2. | name | varchar | 30 | Lietotāja vārds |
| 3. | email | varchar | 255 | Lietotāja e-pasts |
| 4. | username | varchar | 30 | Lietotāja lietotājvārds |
| 5. | password | varchar | 30 | Lietotāja parole |
| 6. | isAdmin | tinyint | 1 | Nosaka vai lietotājs ir administrators |
| 7. | created\_at | timestamp | - | Lietotāja izveidošanas laiks |
| 8. | status | tinyint | 1 | Nosaka vai lietotājs ir bloķēts |
| 9. | last\_login\_at | timestamp | - | Lietotāja pēdējais autorizācijas laiks |

Tabula **„users”** satur primāro atslēgu **„id”**. Satur arī lauku **„isAdmin”**, kurš nosaka vai lietotājs ir administrators un lauku **„status”**, kurš nosaka vai lietotājs ir bloķēts.

5.2. tabula

Tabulas **„profiles”** struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | int | - | Profila kārtas numurs |
| 2. | user\_id | int | - | Ārējā atslēga no tabulas **„users”** |
| 3. | title | varchar | 30 | Profila virsraksts |
| 4. | description | varchar | 5000 | Profila apraksts |
| 5. | url | varchar | 255 | Profila links |
| 6. | media | varchar | 5000 | Profila attēls |

Tabula **„profiles”** satur primāro atslēgu **„id”** un ārējo atslēgu **„user\_id”** no tabulas **„users”**, ar kuru palīdzību tiek veikta attiecība viens-pret-daudziem starp tabulām **„users”** un **„profiles”**.

5.3. tabula

Tabulas **„profile\_user”** struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | - | Komentāra kārtas numurs |
| 2. | profile\_id | bigint | - | Ārējā atslēga no tabulas **„profiles”** |
| 3. | user\_id | bigint | - | Ārējā atslēga no tabulas **„user”** |

Tabula **„profile\_user”** satur trīs atslēgas: primāro atslēgu **„id”,** ārējo atslēgu **„profile\_id”** no tabulas **„profiles”** un ārējo atslēgu **„user\_id”** no tabulas **„users”.** Ārējās atslēgas tiek izmantotas, lai izveidotu saiti daudz-pret-daudzem starp tabulām **„profiles”** un **„users”** (viens-pret daudziem starp **„profiles”** un **„profile\_user”** un viens pret daudziem starp **„users”** un **„profile\_user”**).

5.4. tabula

Tabulas **„comments”** struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Komentāra kārtas numurs |
| 2. | comment | varchar | 255 | Komentāra teksts |
| 3. | post\_id | bigint | 20 | Ārējā atslēga no tabulas **„posts”** |
| 4. | user\_id | bigint | 20 | Ārējā atslēga no tabulas **„users”** |
| 5. | created\_at | timestamp | - | Komentāra izveidošanas laiks |

Tabula **„comments”** satur trīs atslēgas: primāro atslēgu **„id”,** ārējoatslēgu **„post\_id”** no tabulas **„Publikacijas”** un ārējo atslēgu **„user\_id”** no tabulas **„users”**. Lauks **„post\_id”** tiks izmantots, lai izveidotu saiti viens-pret-daudziem starp tabulām komentāru un publikāciju tabulām. Lauks **„user\_id”** tiks izmantots, lai izveidotu saiti viens-pret-daudziem starp komentāru un lietotāju tabulām.

5.5. tabula

Tabulas **„posts”** struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | post\_id | int | **-** | Publikācijas kārtas numurs |
| 2. | user\_id | int | - | Ārējā atslēga no tabulas **„users”** |
| 3. | caption | varchar | 255 | Publikācijas apraksts |
| 4. | media | varchar | 255 | Publikācijas fails (attēls/video) |

Tabula **„posts”** satur divas atslēgas: primāro atslēgu **„post\_id”** unārējo atslēgu **„user\_id”** no tabulas **„users”**, ar kuru palīdzību tiek veikta attiecība viens-pret-daudziem starp tabulām **„posts”** un **„users”**.

5.6. tabula

Tabulas **„messages”** struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Ziņojuma kārtas numurs |
| 2. | message\_sender | varchar | 50 | Ziņojuma sūtītājs, ārējā atslēga no tabulas **„users”** |
| 3. | message\_receiver | varchar | 50 | Ziņojuma saņēmējs, ārējā atslēga no tabulas **„users”** |
| 4. | content | text | - | Ziņojuma teksts |
| 5. | status | varchar | 20 | Ziņojuma statuss (izlasīts vai nē) |
| 6. | created\_at | timestamp | - | Ziņojuma izveidošanas laiks |

Tabula **„message”** satur primāro atslēgu **„id”** un 2 ārējās atslēgas **„message\_sender”** un **„message\_receiver”** no tabulas **„users”** . Lauks **„message\_sender”** tiks izmantots, lai izveidotu saiti viens-pret-daudziem starp tabulām **„users”** un **„messages”** kā ziņojuma sūtītāju. Tas nozīmē, ka viens lietotājs var nosūtīt daudz ziņojumu, bet katrs ziņojums var būt tikai no viena sūtītāja. Līdzīgi, lauks **„message\_receiver”** tiks izmantots, lai izveidotu saiti viens-pret-daudziem starp tabulām **„users”** un **„messages”** kā ziņojuma saņēmēju. Tas nozīmē, ka viens lietotājs var saņemt daudz ziņojumu, bet katrs ziņojums var būt adresēts tikai vienam saņēmējam. Šīs saites ir būtiskas, lai nodrošinātu privātā čata funkcionalitāti starp lietotājiem. Privātā čata sistēma ļauj lietotājiem nosūtīt un saņemt ziņojumus, kas tiek glabāti tabulā **„messages”**, saglabājot saites starp sūtītāju un saņēmēju.

1. **LIETOTĀJA CEĻVEDIS**
   1. **Sistēmas prasības aparatūrai un programmatūrai**

Lai veiksmīgi palaistu un darbinātu izstrādāto sociālā tīkla sistēmu, kas realizēta kā interneta tīmekļa vietne, nepieciešams atbilstošs aparatūras un programmatūras nodrošinājums. Sistēmas lietošana neprasa specifisku programmu instalāciju, tādēļ to var izmantot no jebkuras ierīces, kurai ir piekļuve internetam un atbilstoša pārlūkprogramma.

Aparatūras prasības:

* **Procesors:** 1 GHz vai ātrāks ar vismaz diviem kodoliem.
* **Operatīvā atmiņa:** Minimums 2 GB RAM, ieteicams 4GB vai vairāk.
* **Cietā diska vieta:** Vismaz 500MB brīvas vietas pārlūkprogrammas kešatmiņai un pagaidu failiem.
* **Displejs:** Izšķirtspēja vismaz 1024 x 768 pikseļi.

Programmatūras prasības:

* Operētājsistēmas:
  + **Windows:** Windows 7 un jaunākas versijas.
  + **macOs:** macOS 10.12 (Sierra) un jaunākas versijas.
  + **Linux:** Modernas Linux izplatīšanas ar jaunākajām drošības un programmatūras atjauninājumiem.
  + **Mobilās ierīces:** iOS 12 un jaunākas versijas, Android 8.0 (Oreo) un jaunākas versijas.
* Pārlūkprogrammas:
  + **Safari:** 13. versija un jaunākas.
  + **Microsoft Edge:** 98.0.1108.43 un jaunākas.
  + **Google Chrome:** 98.0.4758.107 un jaunākas.
  + **Opera:** 84.0.4316.31 un jaunākas.
  + **Mozilla Firefox:** 98.0.1 un jaunākas.

Tīkla prasības:

* **Interneta savienojums:** Stabils un ātrs interneta savienojums. Minimālais ātrums 1 Mbps lejupielādei un 512 Kbps augšupielādei, ieteicamais ātrums 5 Mpbs lejupielādei un 1 Mbps augšupielādei, lai nodrošinātu nevainojamu sistēmas darbību.

Šis prasības nodrošina, ka sistēma darbojas efektīvi un lietotājiem draudzīgi, ļaujot piekļūt visām sociālā tīkla funkcijām, tostarp attēlu un video publicēšanai un kopīgošanai, saziņai ar citiem lietotājiem un sekošanai. Sistēmas optimālā darbība ir atkarīga no atbilstošas aparatūras un programmatūras konfigurācijas, kā arī no stabila interneta savienojuma.

* 1. **Sistēmas instalācija un palaišana**

Lai instalētu un palaistu sociālā tīkla sistēmu, jāizpild sekojošas darbības:

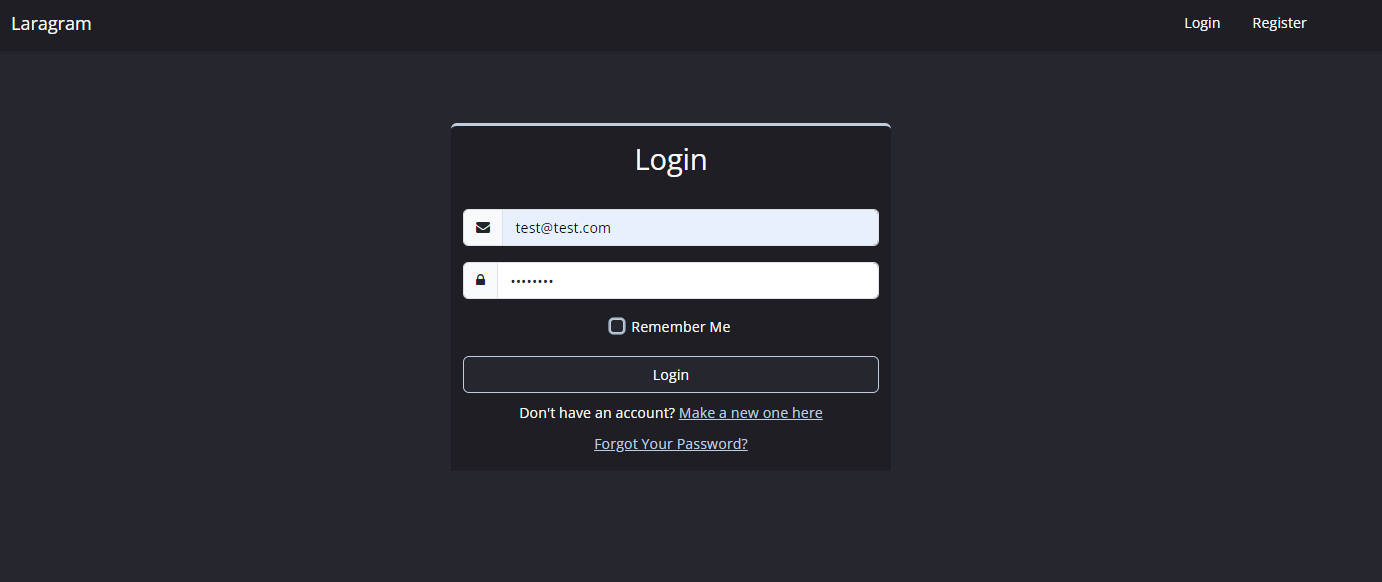
* Priekšnosacījumi:
  + Instalēts un konfigurēts tīmekļa serveris (Laragon).
  + Instalēta PHP (versija 8.2.6 vai jaunāka).
  + Instalēta MySQL datubāze (versija 8.0 vai jaunāka).
  + Instalēts Composer (PHP atkarību pārvaldnieks).
  + Instalēts Laravel, izmantojot “composer global require laravel/installer”.
* Koda lejupielāde un izvietošana:
  + Klonēts projekta kods no GitHub repozitorija servera publiskajā direktorijā (D:/laragon/www/).
* Atkarību instalēšana:
  + Izpildīta komanda “composer install”, lai instalētu visas nepieciešamās atkarības priekš PHP.
  + Instalētas Node.js atkarības un npm (Node Package Manager), izmantojot “npm install”.
  + Instalēta papildu PHP pakete attēlu apstrādei, izmantojot “composer require intervention/image”.
* Saišu izveide:
  + Izveidota simboliska saite uz uzglabāšanas direktoriju, izmantojot “php artisan storage:link”.
* Konfigurācijas failu iestatīšana:
  + Kopēts “.env.example” fails un nosaukts “.env” izmantojot komandu “cp .env.example .env”.
  + Rediģēts “.env” fails, ievadīti pareizi datubāzes un citi nepieciešamie parametru iestatījumi.
* Datubāzes migrācijas:
  + Izpildīta komanda “php artisan migrate”, lai uztaisītu datubāzes tabulas.
* Palaišana:
  + Palaists Laravel iebūvētais serveris, izmantojot komandu “php artisan serve”.
  + Palaists npm, lai kompilētu un optimizētu front-end aktīvus, izmantojot “npm run dev”.

Lietotāja puse:

* Piekļuve tīmekļa vietnei:
  + Atvērta pārlūkprogramma un ievadīts servera URL <http://127.0.0.1:8000>
* Reģistrācija un pieteikšanās:
  + Japiereģistrējas sistēmā, ievadōt nepieciešamos datus. Lietotāja profila e-pasts – [test@test.com](mailto:test@test.com), parole – Password123
  + Lai piekļūtu administratora profilam, jāpiereģistrējas kā lietotājs un pēc tam datubāzē tabulā “users” jāsamaina lauks “isAdmin” uz 1. Pēc tam jaēj uz login lapu un jāievada tā lietotāja datus. Administratora e-pasts – [admin@test.com](mailto:admin@test.com), parole – Admin123
  1. **Programmas apraksts**

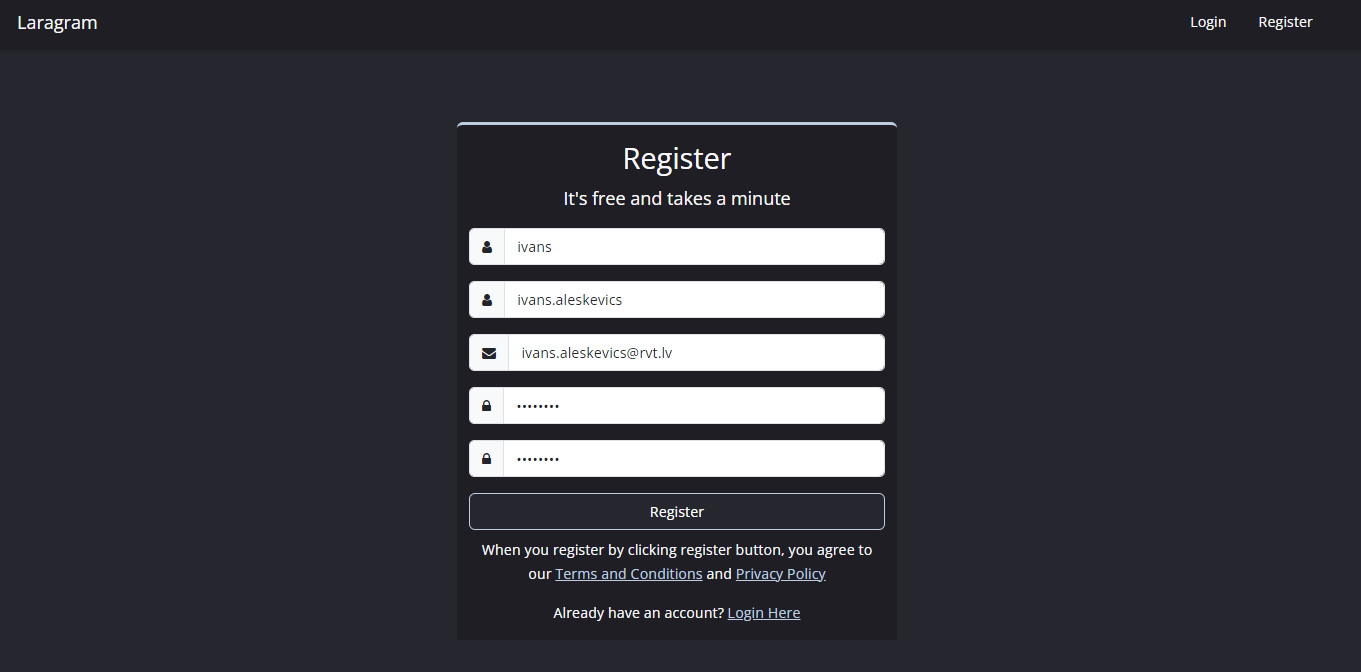
**Lietotāju reģistrācija un autorizācija**

Vispirms atvēram mājaslapu: <http://127.0.0.1:8000/>. Ievadam e-pastu un anela. Spiežam uz Login un ienākam profilā (skat. 6.1. att.).

****

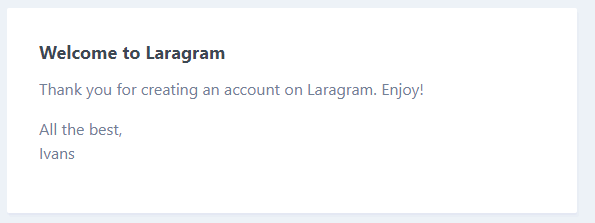
6.1. att. Sākuma lapa

Ja profila nav, tad spiežam Register uz navigācijas anela vai “Make a new one here” login formas lejā, lai pārietu uz reģistrācijas lapu. Ierakstam vārdu, lietotājvārdu, e-pastu, paroli un otrreiz paroli, lai to apstiprinātu. Kad visi lauki aizpildīti, spiežam Register. Ja profils jau ir, tad spiežam Login navigācijas paneļa labajā stūrī vai “Login here” reģistrācijas formas lejā (skat. 6.2. att.).

****

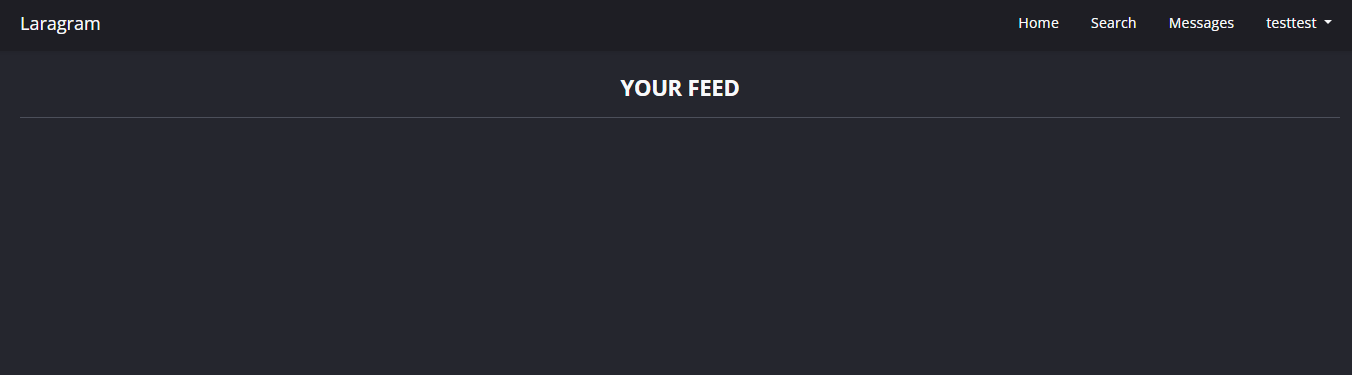
6.2. att. Reģistrācijas lapa

Pēc veiksmīgas reģistrācijas uz e-pastu atnāks paziņojums (skat. 6.3. att.).



6.3. att. E-pasta paziņojums par reģistrāciju

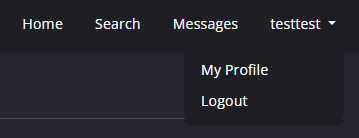
Pēc veiksmīgas reģistrācijas vai logina, atvērsies galvenā lapa (skat. 6.4. att).

******

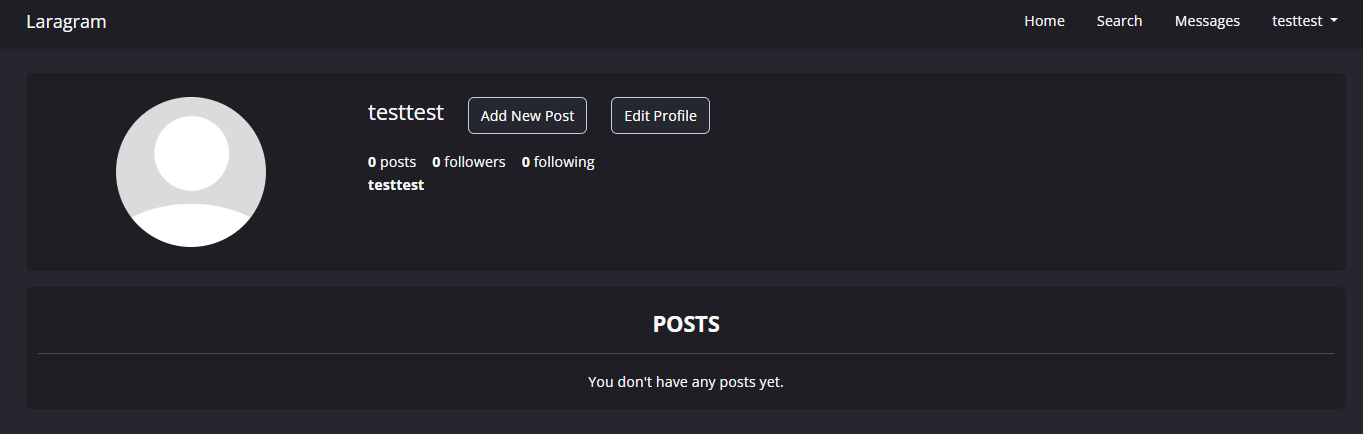
6.4. att. Galvenā lapa

**Lietotāja profila lapa**

Galvenajā lapā navigācijas panelī ir redzamas dažas pogas Home, Search, Messages un profila lietotājvārds ar dropdown menu. Uzklikšķinot uz lietotājvārda, parādās poga My Profile (skat. 6.5. att.).

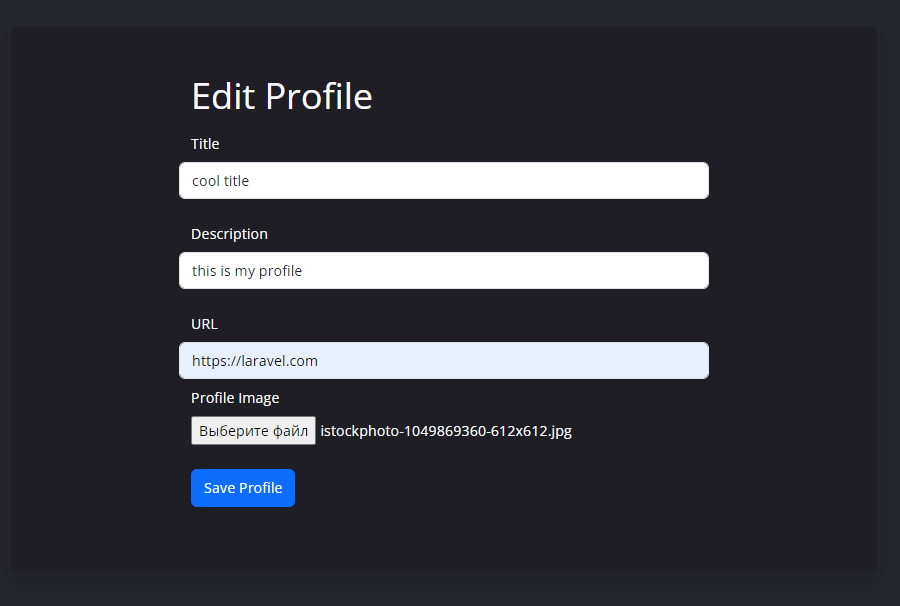


6.5. att. Poga My Profile

Uzklikšķinot uz pogas “My Profile”, parādās lietotāja profila lapa, kur mēs redzam profila attēlu pēc noklusējuma, ir 2 divas pogas – “Add New Post”, “Edit Profile”, kā arī rādās publikāciju, sekotāju un sekošanu skaits (skat. 6.6. att).

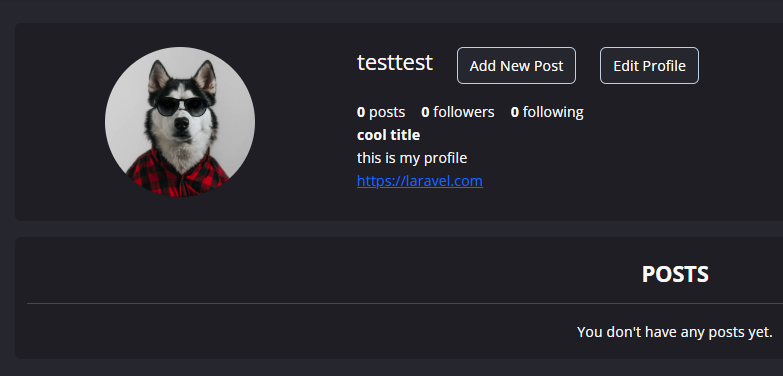
6.6. att. Lietotāja profils

Uzklikšķinot uz “Edit Profile”, parādās profila rediģēšanas lapa ar formu, kur mēs varam mainīt profila titulu, aprakstu, linku un profila attēlu. Pēc lauku aizpildīšanas, spiežam pogu Save Profile (skat. 6.7. att.).



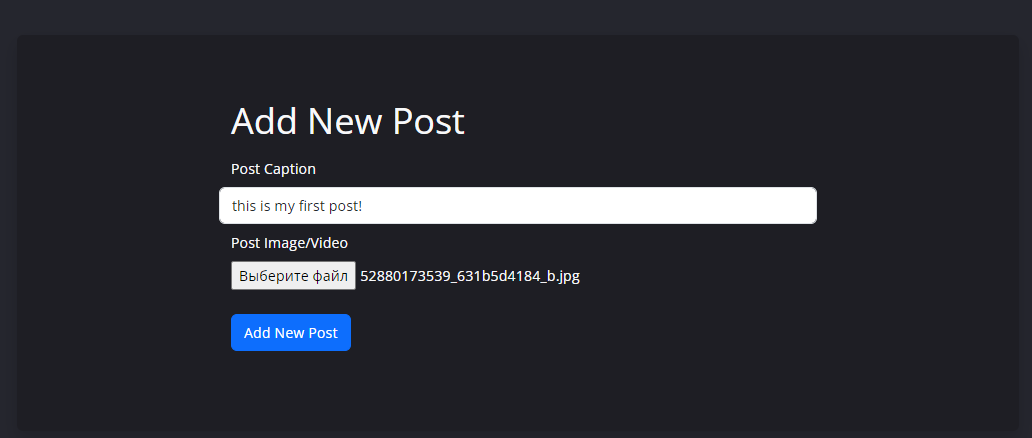
6.7. att. Profila rediģēšana

Pēc saglabāšans, mūs pārmet atpakaļ uz mūsu profilu, kur varam redzēt, ka mums mainijās profila attēls, apraksts un links (skat. 6.8. att.).



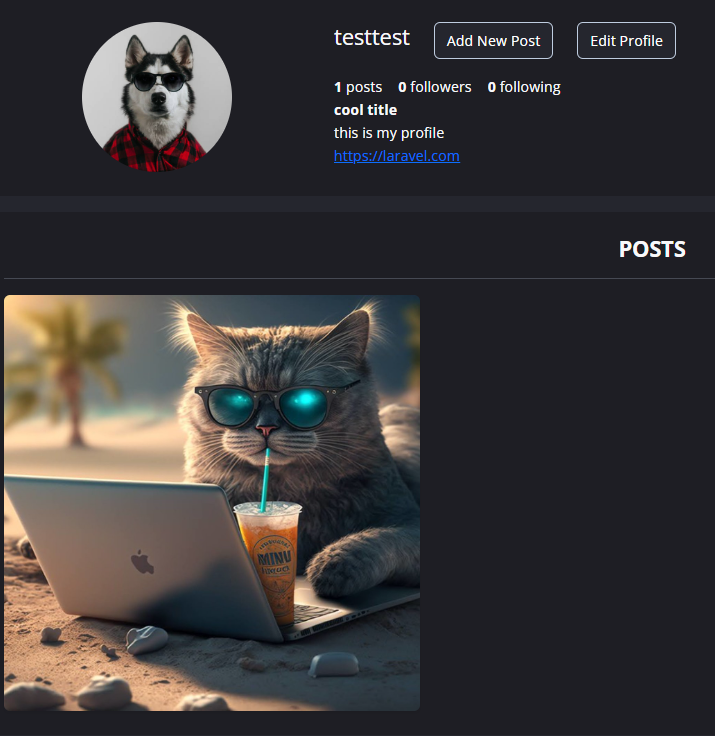
6.8. att. Lietotāja profils pēc rediģēšanas

Var redzēt, ka mums ir 0 publikāciju. Uzspiežot uz “Add New Post”, parādās publikāciju pievienošanas lapa. Ierakstam publikācijas aprakstu, izvēlamies attēlu un spiežam “Add New Post” (skat. 6.9. att.).

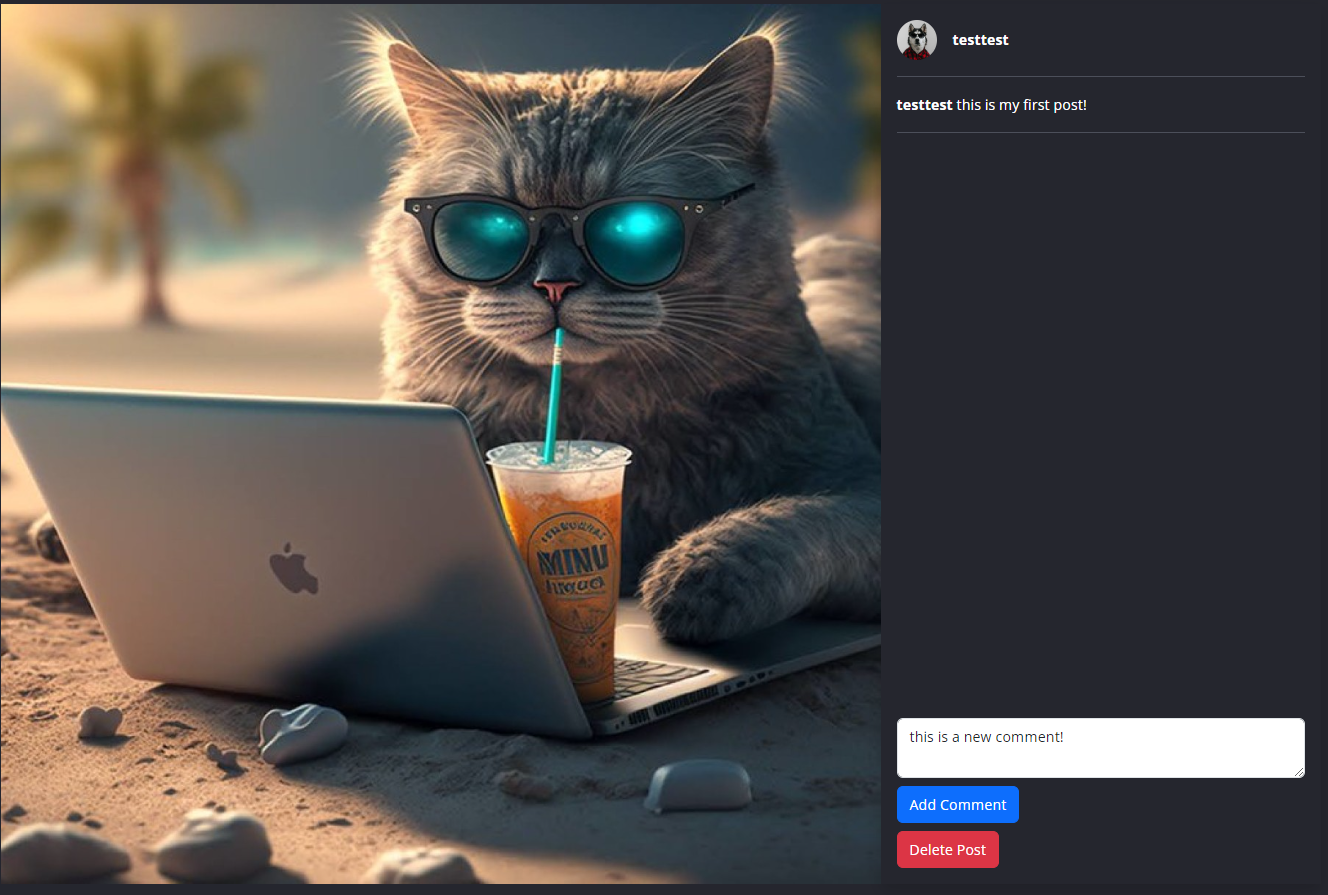


6.9. att. Publikāciju pievienošana

Mūs pārmet atpakaļ uz profilu, kur mēs redzam mūsu jaunu publikāciju. Publikāciju skaits profilā mainās uz 1 (skat. 6.10. att.).

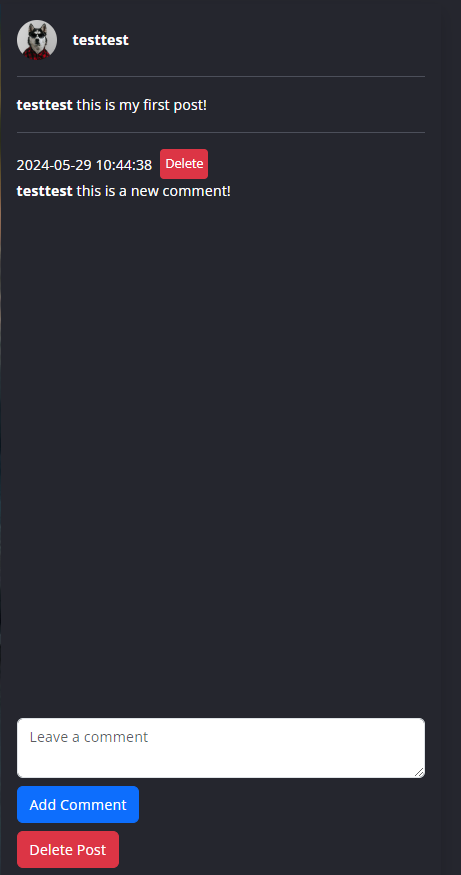


6.10. att. Lietotāja profils pēc publikācijas pievienošanas

Uzklikšķinot uz publikācijas, mūs pārmet uz tās publikācijas lapu, kur ir redzams lietotāja lietotājvārds, attēls un lietotāja publikācijas apraksts. Lai pievienotu komentāru, ierakstam to laukā un spiežam pogu “Add Comment” (skat. 6.11. att.).

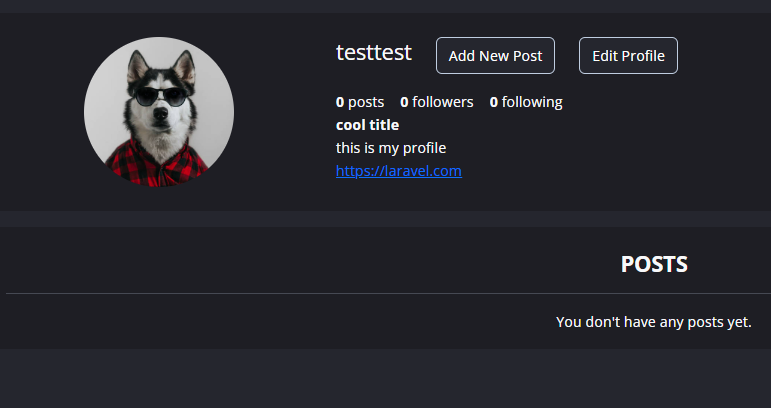
6.11. att. Lietotāja publikācijas atvēršana

Mums parādās jauns komentārs, datums un laiks, kad tas bija pievienots un poga “Delete”. Uzklikšķinot uz “Delete”, komentārs tiek izdzēsts (skat. 6.12. att.).



6.12. att. Pievienots komentārs

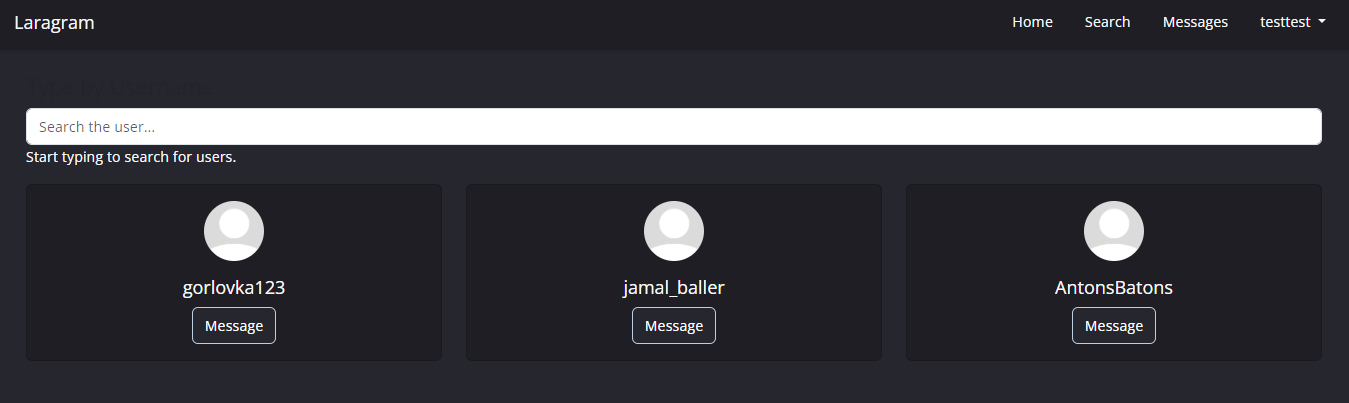
Lai izdzēstu savu publikāciju, lietotājam jāuzspiež sarkanā poga “Delete Post” (skat. 6.13. att.).



6.13. att. Lietotāja profils pēc publikācijas dzēšanas

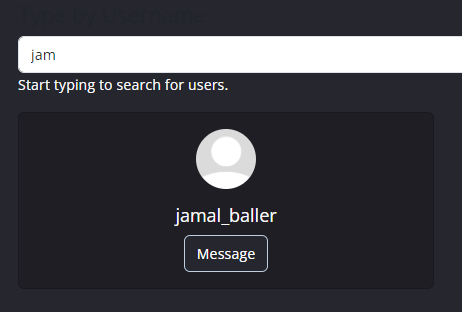
**Meklēšana**

Uzklikšķinot uz pogas Search, mūs pārmet uz lietotāju meklēšanas lapu, kur var redzēt visus lietotājus, kuri ir piereģistrēti sociālajā tīklā. (skat. 6.14. att.).



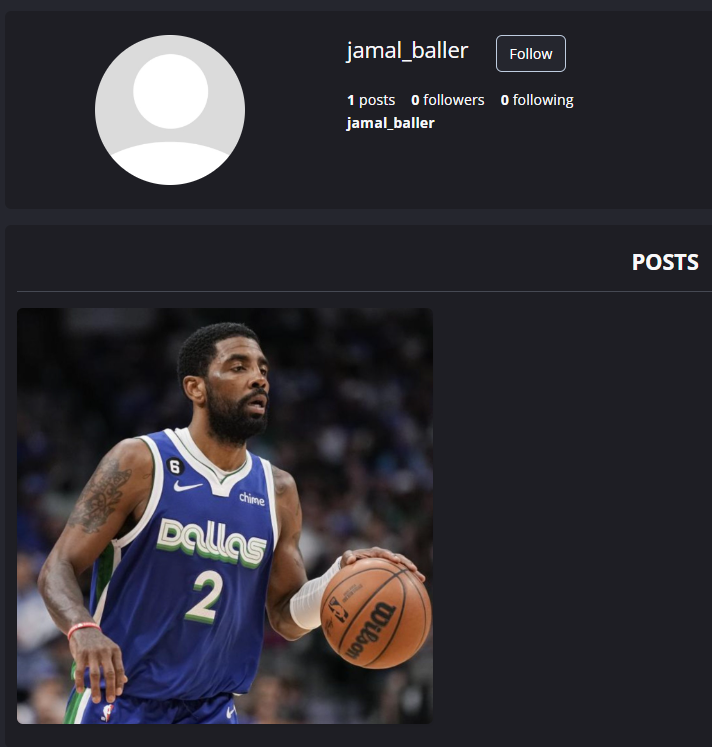
6.14. att. Lietotāju meklēšanas lapa

Lai atrastu kādu lietotāju, ir jāieraksta tā lietotāja lietotājvārds vai līdzīgus simbolus meklēšanas joslā. Rezultātā parādīsies visi lietotāji, kas ir saistīti ar to tekstu (skat. 6.15. att.).

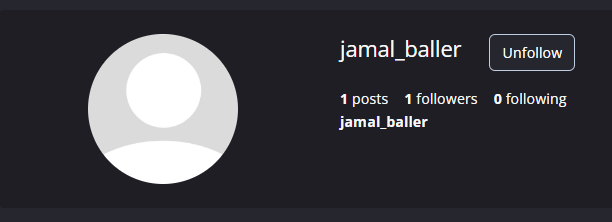


6.15. att. Meklēšanas rezultāts

Uzspiežot uz lietotājvārdu vai profila attēli, var pāriet uz viņa profilu, kur pārādīsies poga “Follow” (skat. 6.16. att.).

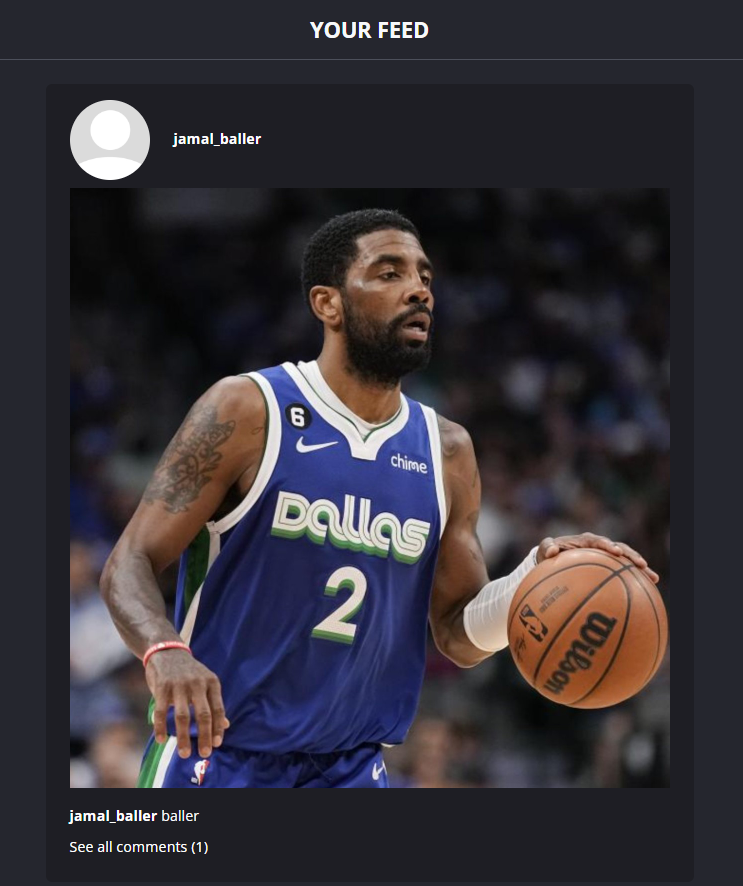


6.16. att. Profila atrādīšana no meklēšanas lapas

Uzklikšķinot Follow pogu, mēs varam sekot līdzi šim lietotājam, Follow samainīsies uz Unfollow. Tagad rāda, ka šim lietotājam ir 1 sekotājs. (skat. 6.17. att.).

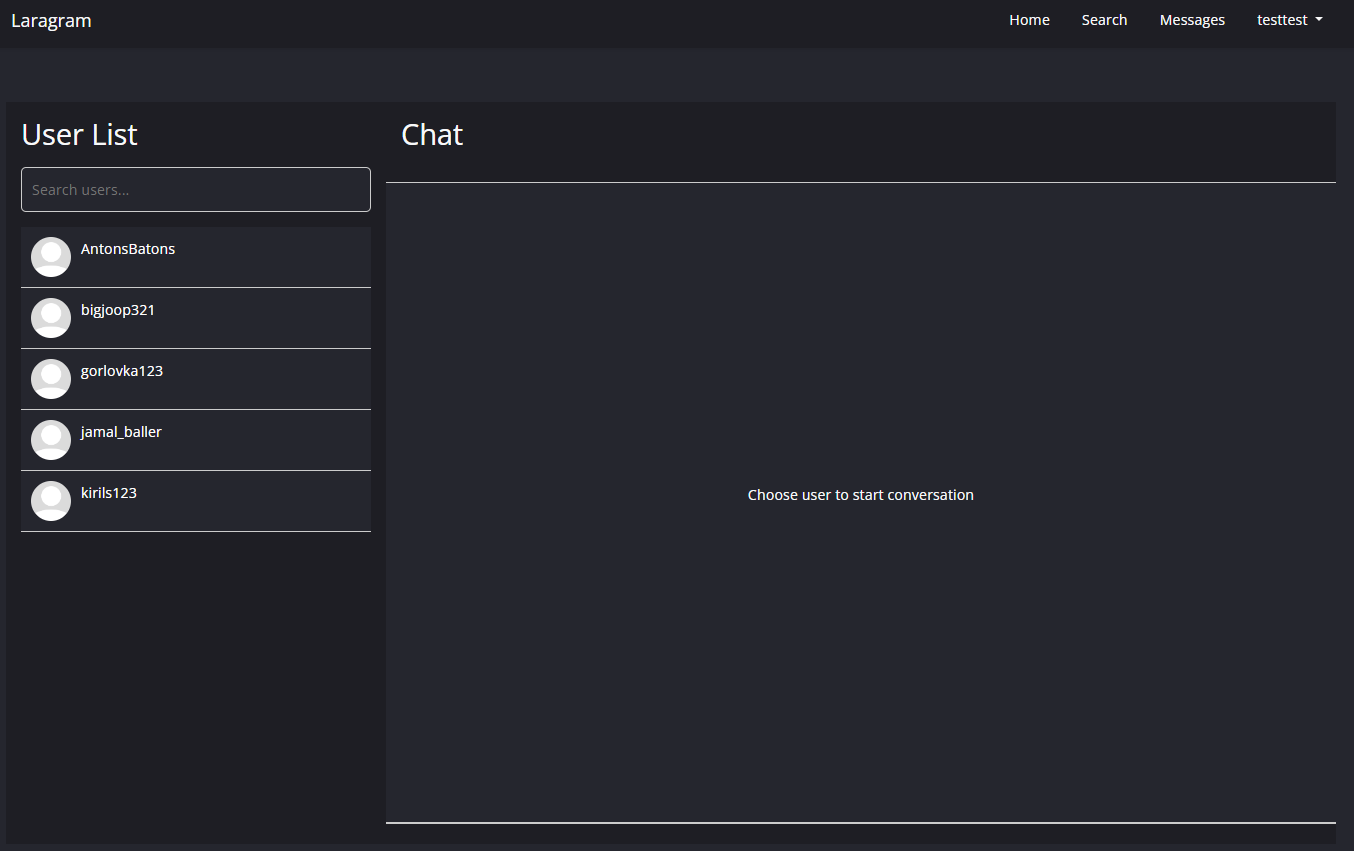
6.17. att. Sekošana citam lietotājam

Tagad uzklikšķinot uz Home un aizejot uz galveno lapu, mēs varēsim redzēt ši lietotāja publikāciju. Publikācijas lejā varēs redzēt, cik komentāru ir šai publikācijai, uzklikšķinot uz “See all comments”, pārmetīs uz publikācijas lapu, kur varēs redzēt tos komentārus. (skat. 6.18. att.).



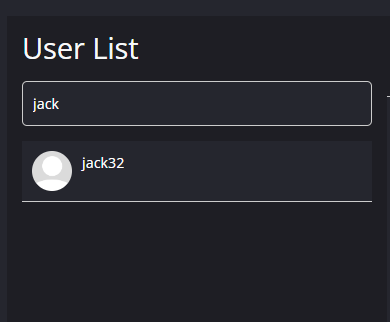
6.18. att. Sekoto lietotāju publikāciju atrādīšana galvenajā lapā

**Čats**

 Uzspiežot uz pogas Messages navigācijas paneļa labajā stūrī vai uzspiežot uz Message meklēšanas lapā pie noteikta lietotāja, parādīsies čats, kur mēs redzam visus lietotājus, ar kuriem mums bija kādi ziņojumi (skat 6.19. att.).

6.19. att. Čats ar lietotājiem

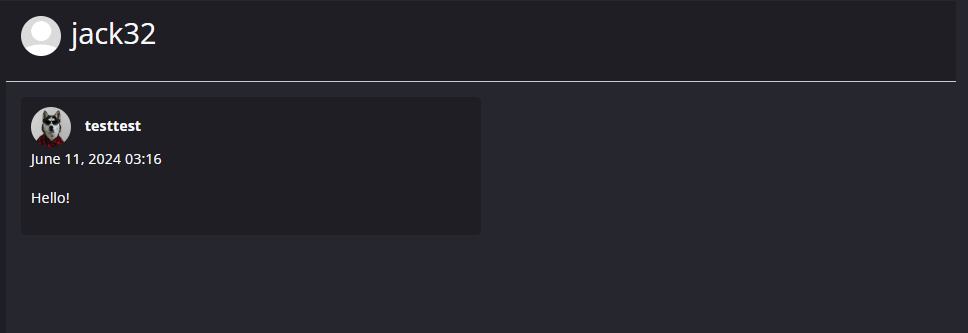
Ja nebija nekādu ziņojumu ar citiem lietotājiem, nerādīsises neviens lietotājs lietotāju sarakstā. Lai atrastu lietotāju, jāievada meklēšanas joslā lietotājvārds vai līdzīgi simboli. (skat. 6.20. att.).



6.20. att. Lietotāju meklēšana čata lapā

****Lai atsūtītu ziņojumu lietotājam, jāuzklikšķina uz lietotāja, pēc tam parādīsies čata logs ar to lietotāju. Lai atsūtītu ziņojumu jāuzraksta kāds teksts ziņojuma laukā un jāspiež poga Send (skat. 6.21. att).

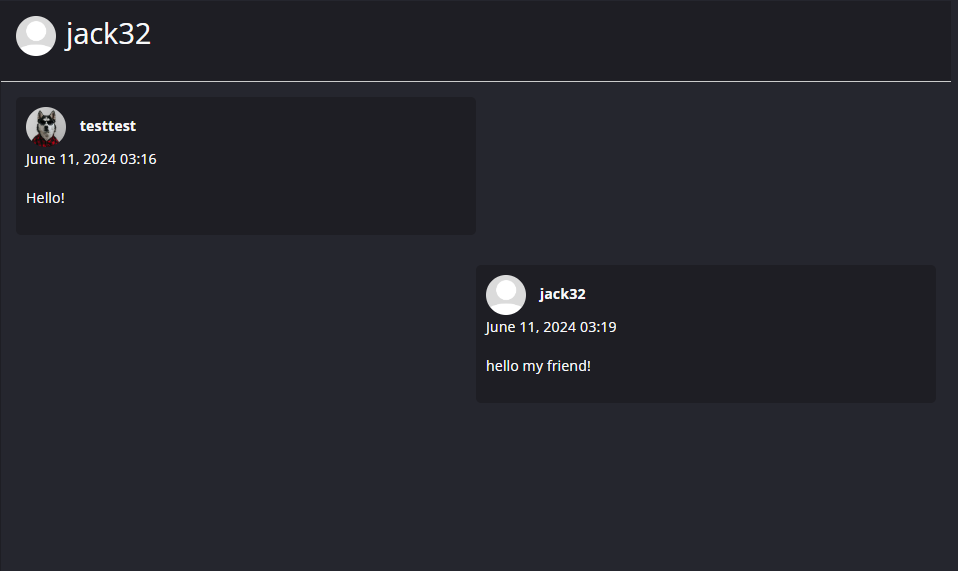
6.21. att. Čats ar lietotāju

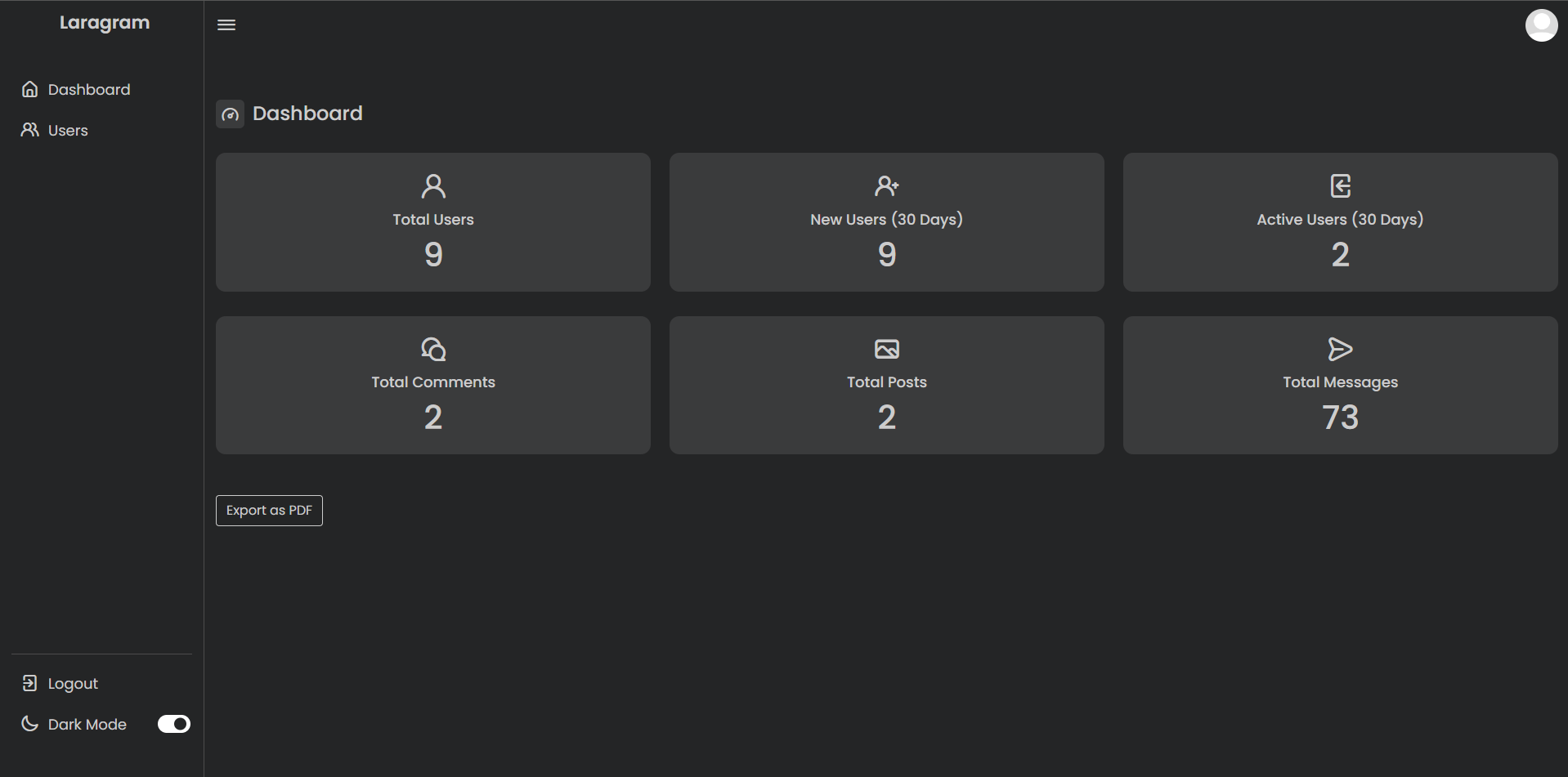
Sūtītāja ziņojums parādīses čata loga kreisajā pusē (skat. 6.22. att).

6.22. att. Sūtītāja ziņojums

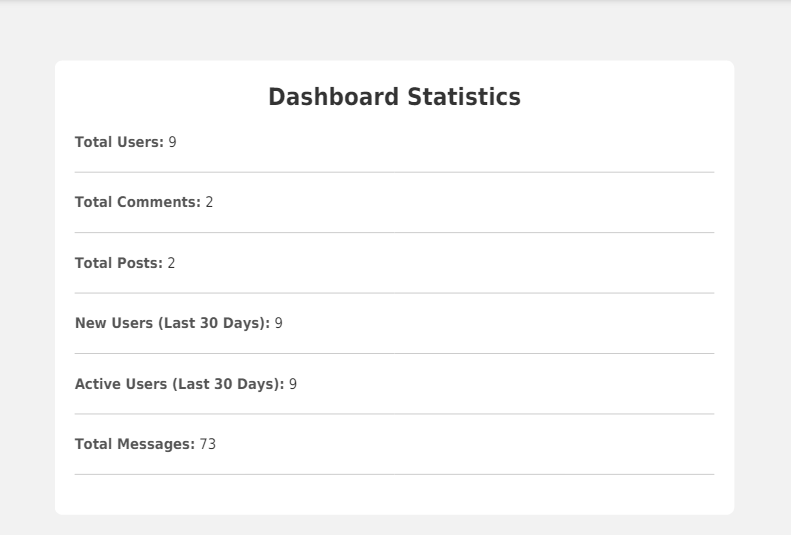
Saņēmēja ziņojums parādīsies čata loga labajā pusē (skat 6.23. att.).

6.23. att. Saņēmēja ziņojums

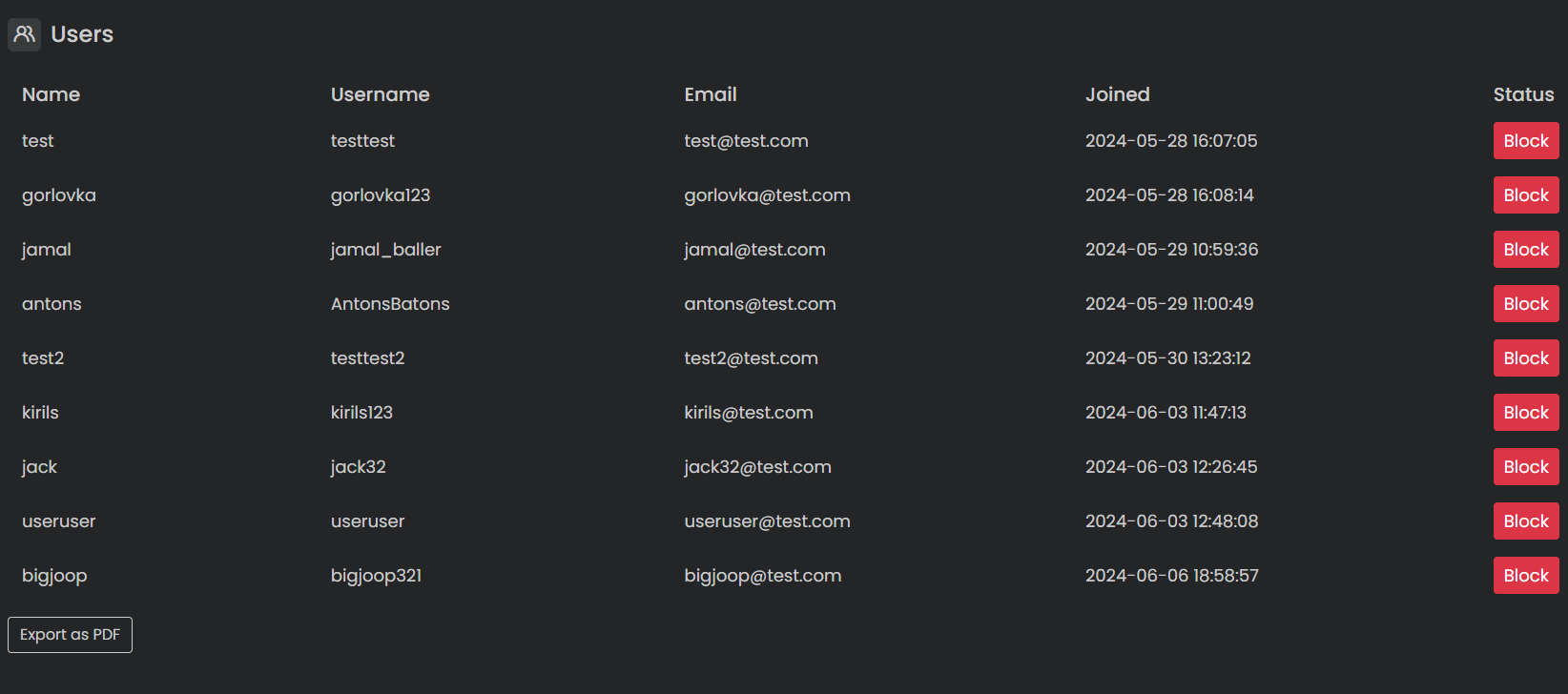
**Administratora** **panelis**

****Lai ienāktu administrātora profilā, jāievada administratora dati Login lapā. Pec Login pogas nospiešanas, mūs pārmet administratora lapā, kur varam redzēt statistiku par lietotājiem, komentāriem, publikācijām un ziņojumiem (skat. 6.24. att.).

6.24. att. Administratora panelis

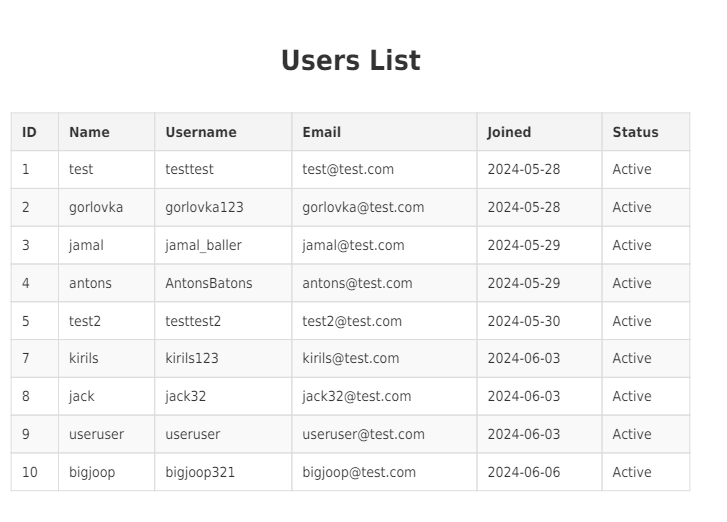
Uzspiežot uz pogas Export as PDF, var eksportēt šo statistiku .pdf failā. Mums lejupielādēsies .pdf fails dashboard\_statistics.pdf, kur var redzēt visu šo statistiku (skat. 6.25. att.).

6.25. att. Statistikas eksportēšana PDF failā

****Uzklikšķinot uz pogas Users navigācijas paneļa kreisajā stūrī, parādās visi lietotāji, kuri reģistrējas tīklā (skat. 6.26. att.).

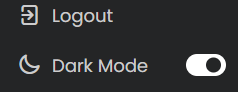
6.26. att. Lietotāju atrādīšana administratora panelī

Uzspiežot uz pogas Export as PDF, var eksportēt visu lietotāju sarakstu .pdf failā. Mums lejupielādēsies .pdf fails users.pdf, kur var redzēt visu lietotājus (skat. 6.27. att.).

****

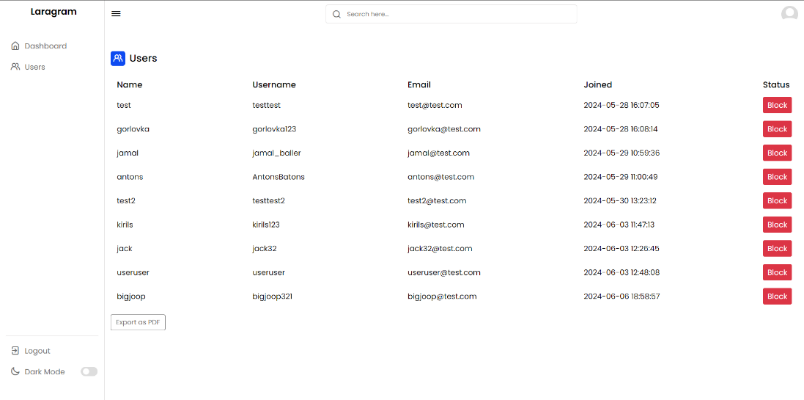
6.27. att. Lietotāju saraksta eksportēšana PDF failā

Ja administratoram nepatīk tumša tēma, to var samainīt uz gaišo tēmu uzklikšķinot uz Dark Mode navigācijas paneļa lejā, lai izslēgtu tumšo tēmu (skat. 6.28. att.).

****

6.28. att. Tēmas pārslēgšana

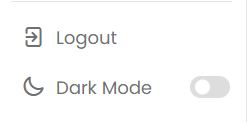
Var redzēt, ka tēma pārslēdzas veiksmīgi (skat. 6.29. att.).

****

6.29. att. Gaišā tēma

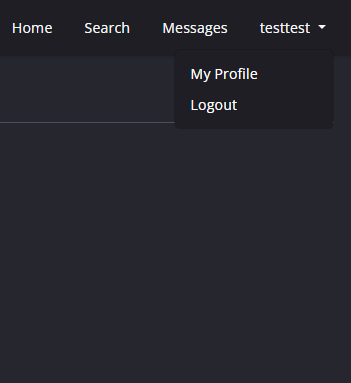
**Darbību beigšana**

Lai administratoram pabeigtu sesiju, jāspiež Logout navigācijas paneļa lejā, kreisajā stūrī. Administratoru izrakstīs no konta un pārmetīs atpakaļ Login lapā (skat. 6.30. att.).



6.30. att. Login poga priekš administratora

Lai lietotājam pabeigtu sesiju, jāspiež uz sava lietotājvārda navigācija paneļa, labajā stūrī, atvērsies dropdown menu, jāspiež poga Logout. Lietotāju izrakstīs no konta un pārmetīs Login lapā. (skat. 6.31. att.).



6.31. att. Poga Logout priekš lietotāja

* 1. **Testa piemērs**

Funkcionālo prasību testēšanas process ir būtisks, lai nodrošinātu, ka programmatūra darbojas pareizi un atbilst prasībām. Šajā pārskatā ir apkopotas trīs galvenās prasības un to testēšanas rezultāti, lai demonstrētu programmas pareizu darbību dažādos apstākļos.

**Testēšanas pārskats**

6.1. tabula

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr.** | **Prasības numurs** | **Prasības nosaukums** | **Ievaddati/situācijas apraksts** | **Sagaidāmais rezultāts** | **Statuss** |
| 1. | 4 | Profila rediģēšana | Pareizi dati  Attēls formātā png/jpeg  Aizpildītie titula, apraksta un url lauki | Profila attēls samainās uz izvēlēto, tituls, apraksts un url mainās | Pareizi |
| 2. | Attēls formātā png/jpeg  Tituls, apraksts vai url lauks nav aizpildīts | Profila attēls samainīts, izvada, lauks, kurš netika aizpildīts, paliek nemainīgs | Pareizi |
| 3. | Attēls netiek izvēlēts, citi lauki aizpildīti | Profila attēls nemainās, citi lauki mainās | Pareizi |
| 4. | 3 | Publikācijas pievienošana | Aizpildīts publikācijas apraksta lauks un izvēlēts attēls | Publikācija pievienota un tiek parādīta lietotāja profilā, apraksts rādās publikācijā | Pareizi |

6.1. tabulas turpinājums

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5. |  |  | Aizpildīts publikācijas apraksta lauks un izvēlēts video | Publikācija pievienota un tiek parādīta lietotāja profilā, apraksts rādās publikācijā | Pareizi |
| 6. | Tiek izvēlēts attēls, bet nav aizpildīts publikācijas apraksta lauks | Publikācija pievienota un tiek parādīta lietotāja profilā, apraksts nerādās publikācijā | Pareizi |
| 7. | 6.1 | Komentāru pievienošana | Komentāra teksts un peles klikšķis uz pogas “Add Comment” | Komentārs parādās publikācijā | Pareizi |
| 8. | 13 | Iespēja meklēt citus lietotājus | Pareizi dati  Lietotāja username | Parādās lietotāja profils un/vai līdzīgi lietotāji | Pareizi |
| 9. | Random teksts, kurš nav saistīts ne ar vienu lietotājvārdu | Parādās “No Results Found” | Pareizi |
| 10. | Tukšs meklēšanas lauks | Parādās visi lietotāji | Pareizi |
| 11. | Savs lietotājvārds | Savs profils neparādās, var parādīties līdzīgi lietotāji | Pareizi |

6.1. tabulas turpinājums

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 12. | 11.1 | Lietotāju bloķēšana | Peles klikšķis uz pogas “Block” | Lietotāju izmet no profila, ja viņš ir ielogots, ja viņš mēģinās ieiet profilā, izvada paziņojumu, ka viņš ir bloķēts | Pareizi |

Visi testēšanas laikā atrastie defekti tika nekavējoties izlaboti, un atkārtotā testēšana apstiprināja programmatūras pareizu un stabilu darbību. Sistēma tagad atbilst visām sākotnējām funkcionālajām prasībām.

# NOBEIGUMS

Kvalifikācijas darba izstrādes laikā tika izveidota sociālā tīkla sistēma, kas atbalsta publikāciju un ziņojumu kopīgošanu. Projekta ietvaros tika realizētas sekojošas funkcijas: attēlu un video publicēšana, saziņa un interakcija ar citiem lietotājiem (iespēja komentēt publikācijas), sekošana citiem lietotājiem, meklēšanas iespējas, lietotāju bloķēšana, kā arī reģistrācija un pieteikšanās. Papildus tam tika pievienota statistikas un visu lietotāju saraksta eksportēšana PDF formātā. Sistēmas izstrādei tika izmantotas šādas tehnoloģijas: Visual Studio Code, MySQL, PHP jeb Hypertext Preprocessor, Laravel, HTML jeb Hypertext Markup Language, CSS jeb Cascading Style Sheets, Bootstrap, Vue.js un Laragon. Pašlaik projekts ir pilnībā pabeigts, nodrošinot stabilu un funkcionālu sistēmu.

Projekts šobrīd tiek izmantots, lai nodrošinātu lietotājiem platformu, kurā tie var kopīgot attēlus un video saturu, sekot citiem lietotājiem un sazināties savā starpā. Turpmāk šo sistēmu varēs izmantot dažādos sociālo tīklu risinājumos, kā arī to varēs attīstīt tālāk, lai pievienotu jaunas funkcionalitātes un uzlabotu lietotāju pieredzi. Nākotnē ir plānots nodrošināt paziņojumu sistēmu, lai lietotāji tiktu informēti par jauniem komentāriem, sekotājiem vai ziņojumiem, kā arī integrēt grupu izveides iespēju, kas ļautu lietotājiem veidot un pārvaldīt interešu grupas. Tā kā sistēma jau ir pabeigta un atbilst sākotnēji izvirzītajiem tehniskajiem un funkcionālajiem mērķiem, tai ir potenciāls tikt izmantotai plašākā mērogā un pielāgotai lietotāju vajadzībām un prasībām nākotnē.

# INFORMĀCIJAS AVOTI

1. Laravel framework dokumentācija – <https://laravel.com/docs/10.x>. - (Resurss apskatīts 03.11.2023)
2. Sociālais tīkls Instagram – <https://www.instagram.com/>. – (Resurss apskatīts 03.11.2023)
3. Dizaina rīks priekš skicēm – Figma – <https://www.figma.com/>. – (Resurss apskatīts 03.11.2023)
4. Informācija par funkcionālām prasībām – <https://lv.itpedia.nl/2017/11/30/kenmerken-voor-software-requirements/>. – (Resurss apskatīts 03.11.2023)
5. Vue.js dokumentācija – <https://vuejs.org/guide/introduction.html>. - (Resurss apskatīts 03.11.2023)
6. Informācija par ER diagrammu – <https://www.lucidchart.com/pages/er-diagrams#:~:text=Make%20an%20ERD- ,What%20is%20an%20ER%20diagram%3F,each%20other%20within%20a%20system>. – (Resurss apskatīts 03.11.2023)
7. SQLite dokumentācija – <https://www.sqlite.org/docs.html>. - (Resurss apskatīts 03.11.2023)
8. Prasību specifikācija – <https://argondigital.com/blog/product-management/requirements-vs-specifications-create-a-shared-vocabulary/#:~:text=A%20requirement%20specification%20is%20a,and%20maintenance%20of%20the%20product>. – (Resurss apskatīts 03.11.2023)
9. Tīmekļa izstrāde – <https://developer.mozilla.org/en-US/docs/Learn>. - (Resurss apskatīts 03.11.2023)

# PIELIKUMI

1. pielikums. Programmas pirmteksts

**//Visas profila darbības**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\User;

use Illuminate\Support\Facades\Cache;

use Intervention\Image\Facades\Image;

class ProfilesController extends Controller

{

public function \_\_construct()

{

$this->middleware(‘auth’);

}

public function index(User $user)

{

if ($user->isAdmin) {

abort(404); // Return a 404 Not Found response

}

$follows = (auth()->user()) ? auth()->user()->following->contains($user->id) : false;

$postCount = Cache::remember(‘count.posts.’ . $user->id,

now()->addSeconds(5),

function () use ($user) {

return $user->posts->count();

});

$followersCount = Cache::remember(‘count.followers.’ . $user->id,

now()->addSeconds(5),

function () use ($user) {

return $user->profile->followers->count();

});

$followingCount = Cache::remember(‘count.following.’ . $user->id,

now()->addSeconds(5),

function () use ($user) {

return $user->following->count();

});

return view(‘profiles.index’, compact(‘user’, ‘follows’, ‘postCount’, ‘followersCount’, ‘followingCount’));

}

public function edit(User $user)

{

$this->authorize(‘update’, $user->profile);

return view(‘profiles.edit’, compact(‘user’));

}

public function update(User $user)

{

$this->authorize(‘update’, $user->profile);

$data = request()->validate([

‘title’ => ‘nullable’,

‘description’ => ‘nullable’,

‘url’ => ‘nullable|url’,

‘media’ => ‘mimetypes:image/jpeg,image/png’,

]);

if (request(‘media’))

{

$mediaPath = request(‘media’)->store(‘profile’, ‘public’);

$media = Image::make(public\_path(“storage/{$mediaPath}”))->fit(1000, 1000);

$media->save();

$mediaArray = [‘media’ => $mediaPath];

}

auth()->user()->profile->update(array\_merge(

$data,

$mediaArray ?? [],

));

return redirect(“/profile/{$user->id}”);

}

public function search()

{

return view(‘profiles.search’);

}

public function find($query)

{

$currentUserId = auth()->id();

$users = User::where(‘username’, ‘like’, “%$query%”)

->where(‘id’, ‘!=’, $currentUserId) // Exclude the logged-in user

->where(‘isAdmin’, false)

->with(‘profile’)->get();

$usersWithProfileImage = $users->map(function ($user) {

return [

‘id’ => $user->id,

‘username’ => $user->username,

‘media’ => $user->profile->profileImage(),

];

});

return response()->json($usersWithProfileImage);

}

public function allUsers() {

$currentUserId = auth()->id();

$users = User::where(‘id’, ‘!=’, $currentUserId) // Exclude the logged-in user

->where(‘isAdmin’, false)

->with(‘profile’)->get();

$usersWithProfileImage = $users->map(function ($user) {

$profileImage = $user->profile ? $user->profile->profileImage() : null;

return [

‘id’ => $user->id,

‘username’ => $user->username,

‘media’ => $profileImage,

];

});

return response()->json($usersWithProfileImage);

}

}

**//Lietotāja profila lapa**

@extends('layouts.app')

@section('content')

<div class="container">

<!-- User Profile Section -->

<div class="row rounded" style="background-color: #1E1E24">

@if(!$user->isAdmin)

<!-- Profile Image and Information -->

<div class="col-md-3 pt-4 pb-4">

<div class="d-flex justify-content-center">

<img src="{{ $user->profile->profileImage() }}" alt="Profile Image" class="profile-image rounded-circle img-fluid">

</div>

</div>

<!-- Profile Stats and Description -->

<div class="col-md-9 pt-4 text-light">

<div class="d-flex align-items-center pb-3">

<div class="h4">{{ $user->username }}</div>

@if(Auth::check() && Auth::user()->id !== $user->id)

<follow-button user-id="{{ $user->id }}" follows="{{ $follows }}"></follow-button>

@endif

@can('update', $user->profile)

<a href="/p/create"><button class="btn allBtn ms-4">Add New Post</button></a>

@endcan

@can('update', $user->profile)

<a href="/profile/{{ $user->id }}/edit"><button class="btn allBtn ms-4">Edit Profile</button></a>

@endcan

</div>

<div class="d-flex align-items-baseline">

<div class="pe-3"><strong class=>{{ $postCount }}</strong> posts</div>

<div class="pe-3"><strong>{{ $followersCount }}</strong> followers</div>

<div><strong>{{ $followingCount }}</strong> following</div>

</div>

<div class="">

<div class="fw-bold">{{ $user->profile->title }}</div>

<div>{{ $user->profile->description }}</div>

<div><a href="#">{{ $user->profile->url }}</a></div>

</div>

</div>

</div>

<!-- User Posts Section -->

<div class="row mt-3 rounded" style="background-color: #1E1E24">

<div class="col-md-12 pt-4">

<h4 class="fw-bold text-light text-center">POSTS</h4>

<hr class="text-light">

</div>

@if($user->posts->isEmpty())

<div class="col-md-12 text-light text-center">

@if(Auth::check() && Auth::user()->id === $user->id)

<p>You don't have any posts yet.</p>

@else

<p>This user doesn't have any posts yet.</p>

@endif

</div>

@else

@foreach($user->posts as $post)

<div class="col-md-4 pb-4">

<a href="/p/{{ $post->id }}">

<img src="/storage/{{ $post->media }}" alt="Post Image" class="w-100 shadow-sm rounded">

</a>

</div>

@endforeach

@endif

</div>

@endif

</div>

@endsection

**//Visas administratora darbības**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\User;

use App\Models\Comment;

use App\Models\Post;

use Mpdf\Mpdf;

use Carbon\Carbon;

use App\Models\Message;

class AdminController extends Controller

{

public function \_\_construct()

{

$this->middleware('auth');

}

public function dashboard()

{

$totalUsers = User::where('isAdmin', 0)->count();

$totalComments = Comment::count();

$totalPosts = Post::count();

$totalMessages = Message::count();

// Fetch new users in the last 30 days excluding admins

$newUsers = User::where('isAdmin', 0)

->where('created\_at', '>=', Carbon::now()->subDays(30))

->count();

// Fetch active users in the last 30 days excluding admins

$activeUsers = User::where('isAdmin', 0)

->where('last\_login\_at', '>=', Carbon::now()->subDays(30))

->count();

// Logic to fetch data for the admin dashboard

return view('admin.dashboard', compact('totalUsers', 'totalComments', 'totalPosts', 'newUsers', 'activeUsers', 'totalMessages'));

}

public function users()

{

$users = User::select('id', 'name', 'email', 'username', 'created\_at', 'status')

->where('isAdmin', '!=', 1)

->get();

return view('admin.users')->with([

'users' => $users

]);

}

/\*\*

\*To update status of user (ban/unban)

\* @param Integer $user\_id

\* @param Integer $status\_code

\* @return Success Response.

\*/

public function updateStatus($user\_id, $status\_code)

{

try {

$update\_user = User::whereId($user\_id)->update([

'status' => $status\_code

]);

if($update\_user) {

return redirect()->route('admin.users')->with('success', 'User Status Updated

Successfully.');

}

return redirect()->route('admin.users')->with('error', 'Failed to update user status.');

}

catch(\Throwable $th) {

throw $th;

}

}

public function exportUsersPdf()

{

// Fetch the users data

$users = User::select('id', 'name', 'email', 'username', 'created\_at', 'status')->where('isAdmin', '!=', 1)->get();

// Load the view and pass the data

$html = view('admin.users\_pdf', compact('users'))->render();

// Create an instance of mPDF

$mpdf = new Mpdf();

// Write the HTML content to the PDF

$mpdf->WriteHTML($html);

// Output the PDF as a download

return $mpdf->Output('users.pdf', 'D');

}

public function exportDashboardPdf()

{

// Fetch the dashboard statistics

$totalUsers = User::where('isAdmin', 0)->count();

$totalComments = Comment::count();

$totalPosts = Post::count();

$totalMessages = Message::count();

$newUsers = User::where('isAdmin', 0)

->where('created\_at', '>=', Carbon::now()->subDays(30))

->count();

$activeUsers = User::where('isAdmin', 0)

->where('updated\_at', '>=', Carbon::now()->subDays(30))

->count();

// Generate HTML for the PDF content

$html = view('admin.dashboard\_pdf', compact('totalUsers', 'totalComments', 'totalPosts', 'newUsers', 'activeUsers', 'totalMessages'))->render();

// Initialize mPDF instance

$mpdf = new Mpdf();

// Write HTML content to PDF

$mpdf->WriteHTML($html);

// Output PDF as download

$mpdf->Output('dashboard\_statistics.pdf', 'D');

}

}

**//Administratora galvenā lapa**

@extends('layouts.admin')

@section('content')

<div class="top">

<i class="uil uil-bars sidebar-toggle"></i>

<img src="{{auth()->user()->profile->profileImage()}}" alt="">

</div>

<div class="dash-content">

<div class="overview">

<div class="title">

<i class="uil uil-tachometer-fast-alt"></i>

<span class="text">Dashboard</span>

</div>

<div class="boxes">

<div class="box box1">

<i class="uil uil-user"></i>

<span class="text">Total Users</span>

<span class="number">{{ $totalUsers }}</span>

</div>

<div class="box box2">

<i class="uil uil-user-plus"></i>

<span class="text">New Users (30 Days)</span>

<span class="number">{{ $newUsers }}</span>

</div>

<div class="box box3">

<i class="uil uil-signin"></i>

<span class="text">Active Users (30 Days)</span>

<span class="number">{{ $activeUsers }}</span>

</div>

<div class="box box1">

<i class="uil uil-comments"></i>

<span class="text">Total Comments</span>

<span class="number">{{ $totalComments }}</span>

</div>

<div class="box box2">

<i class="uil uil-image"></i>

<span class="text">Total Posts</span>

<span class="number">{{ $totalPosts }}</span>

</div>

<div class="box box3">

<i class="uil uil-message"></i>

<span class="text">Total Messages</span>

<span class="number">{{ $totalMessages }}</span>

</div>

</div>

<div class="export-button ">

<a href="{{ route('admin.export\_dashboard\_pdf') }}" class="btn allBtn">Export as PDF</a>

</div>

</div>

</div>

@endsection

**//Statistikas PDF eksportēšanas lapa**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Dashboard Statistics</title>

<style>

/\* Define your CSS styles for the PDF here \*/

body {

font-family: Arial, sans-serif;

background-color: #f2f2f2;

margin: 0;

padding: 20px;

}

.container {

max-width: 600px;

margin: 0 auto;

background-color: #fff;

padding: 20px;

border-radius: 8px;

}

.title {

font-size: 24px;

font-weight: bold;

color: #333;

margin-bottom: 20px;

text-align: center;

}

.statistic {

margin-bottom: 20px;

border-bottom: 1px solid #ccc;

padding-bottom: 20px;

}

.statistic:last-child {

border-bottom: none;

}

.statistic .name {

font-weight: bold;

color: #555;

}

.statistic .value {

color: #333;

}

</style>

</head>

<body>

<div class="container">

<div class="title">Dashboard Statistics</div>

<div class="statistic">

<span class="name">Total Users:</span>

<span class="value">{{ $totalUsers }}</span>

</div>

<div class="statistic">

<span class="name">Total Comments:</span>

<span class="value">{{ $totalComments }}</span>

</div>

<div class="statistic">

<span class="name">Total Posts:</span>

<span class="value">{{ $totalPosts }}</span>

</div>

<div class="statistic">

<span class="name">New Users (Last 30 Days):</span>

<span class="value">{{ $newUsers }}</span>

</div>

<div class="statistic">

<span class="name">Active Users (Last 30 Days):</span>

<span class="value">{{ $activeUsers }}</span>

</div>

<div class="statistic">

<span class="name">Total Messages:</span>

<span class="value">{{ $totalMessages }}</span>

</div>

</div>

</body>

</html>

**//Visu reģistrēto lietotāju saraksts administratora lapā**

@extends('layouts.admin')

@section('content')

<div class="top">

<i class="uil uil-bars sidebar-toggle"></i>

<div class="search-box">

<i class="uil uil-search"></i>

<input type="text" placeholder="Search here...">

</div>

<img src="{{auth()->user()->profile->profileImage()}}" alt="">

</div>

<div class="dash-content">

<div class="activity">

<div class="title">

<i class="uil uil-users-alt"></i>

<span class="text">Users</span>

</div>

<div class="activity-data">

<div class="data names">

<span class="data-title">Name</span>

@foreach($users as $user)

<span class="data-list">{{$user->name}}</span>

@endforeach

</div>

<div class="data email">

<span class="data-title">Username</span>

@foreach($users as $user)

<span class="data-list">{{$user->username}}</span>

@endforeach

</div>

<div class="data joined">

<span class="data-title">Email</span>

@foreach($users as $user)

<span class="data-list">{{$user->email}}</span>

@endforeach

</div>

<div class="data type">

<span class="data-title">Joined</span>

@foreach($users as $user)

<span class="data-list">{{$user->created\_at}}</span>

@endforeach

</div>

<div class="data status">

<span class="data-title">Status</span>

@foreach($users as $user)

@if($user->status == 1)

<span class="data-list">

<div class="buttons">

<a href="{{ route('users.status.update', ['user\_id' => $user->id, 'status\_code' => 0])}}" class="btn btn-danger">Block</a>

</div>

</span>

@else

<span class="data-list">

<div>

<a href="{{ route('users.status.update', ['user\_id' => $user->id, 'status\_code' => 1])}}" class="btn btn-success">Unblock</a>

</div>

</span>

@endif

@endforeach

</div>

</div>

</div>

</div>

<div class="export-button">

<a href="{{ route('admin.export\_users\_pdf') }}" class="btn allBtn">Export as PDF</a>

</div>

@endsection

**//Visu reģistrēto lietotāju PDF eksportēšanas lapa**

<!DOCTYPE html>

<html>

<head>

<title>Users List</title>

<style>

body {

font-family: Arial, sans-serif;

font-size: 14px;

color: #333;

}

h1 {

text-align: center;

color: #333;

}

table {

width: 100%;

border-collapse: collapse;

margin-top: 20px;

}

th, td {

padding: 10px;

border: 1px solid #ddd;

text-align: left;

}

th {

background-color: #f4f4f4;

color: #333;

}

tr:nth-child(even) {

background-color: #f9f9f9;

}

tr:hover {

background-color: #f1f1f1;

}

</style>

</head>

<body>

<h1>Users List</h1>

<table>

<thead>

<tr>

<th>ID</th>

<th>Name</th>

<th>Username</th>

<th>Email</th>

<th>Joined</th>

<th>Status</th>

</tr>

</thead>

<tbody>

@foreach($users as $user)

<tr>

<td>{{ $user->id }}</td>

<td>{{ $user->name }}</td>

<td>{{ $user->username }}</td>

<td>{{ $user->email }}</td>

<td>{{ $user->created\_at->format('Y-m-d') }}</td>

<td>{{ $user->status ? 'Active' : 'Blocked' }}</td>

</tr>

@endforeach

</tbody>

</table>

</body>

</html>

**//Visas publikāciju darbības**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\User;

use App\Models\Post;

use Intervention\Image\Facades\Image;

class PostsController extends Controller

{

public function \_\_construct()

{

$this->middleware('auth');

}

public function index()

{

$users = auth()->user()->following()->pluck('profiles.user\_id');

$posts = Post::whereIn('user\_id', $users)->with('user', 'comments')->latest()->simplePaginate(5);

return view('posts.index', compact('posts'));

}

public function create()

{

return view('posts.create');

}

public function store()

{

$data = request()->validate([

'caption' => 'required',

'media' => ['required', 'mimetypes:image/jpeg,image/png,video/mp4,video/mpeg,video/quicktime'],

]);

$mediaPath = request('media')->store('uploads', 'public');

$media = Image::make(public\_path("storage/{$mediaPath}"))->fit(1200, 1200);

$media->save();

auth()->user()->posts()->create([

'caption' => $data['caption'],

'media' => $mediaPath,

]);

return redirect('/profile/'. auth()->user()->id);

}

public function destroy($id)

{

$post = Post::where('id', $id)->firstOrFail();

$userId = $post->user->id;

$post->delete();

return redirect()->route("profile.show", ['user' => $userId])->with("success", "Post deleted successfully!");

}

public function show(\App\Models\Post $post, User $user)

{

return view('posts.show', compact('post', 'user'));

}

}

**//Publikāciju izveidošanas lapa**

@extends('layouts.app')

@section('content')

<div class="container mt-5">

<div class="d-flex align-items-center justify-content-center h-100">

<form action="/p" enctype="multipart/form-data" method="POST" class="shadow p-3 mb-5 rounded" style="background-color: #1E1E24">

@csrf

<div class="row p-5 text-light">

<div class="col-8 offset-2">

<h1>Add New Post</h1>

<div class="row mb-3">

<label for="caption" class="col-md-4 col-form-label">Post Caption</label>

<input id="caption"

type="text"

class="form-control @error('caption') is-invalid @enderror"

name="caption"

value="{{ old('caption') }}"

autocomplete="caption" autofocus>

@error('caption')

<span class="invalid-feedback" role="alert">

<strong>{{ $message }}</strong>

</span>

@enderror

<div class="row">

<label for="image" class="col-md-4 col-form-label">Post Image/Video</label>

<input type="file" class="form-control-file" id="media" name="media">

@error('media')

<strong>{{ $message }}</strong>

@enderror

</div>

<div class="pt-4">

<button class="btn btn-primary">Add New Post</button>

</div>

</div>

</div>

</div>

</form>

</div>

</div>

@endsection

**//Galvenā lapa ar publikācijām**

@extends('layouts.app')

@section('content')

<!--<div class="container bg-white rounded border">

@foreach($posts as $post)

<div class="row pt-3">

<div class="col-6 offset-3">

<a href="/profile/{{ $post->user->profile->id }}">

<img src="{{ $post->user->profile->profileImage() }}" alt="" class="rounded-circle w-100" style="max-width: 80px;">

</a>

</div>

</div>

<div class="row">

<div class="col-6 offset-3">

<a href="/p/{{ $post->id }}">

<img src="/storage/{{ $post->media }}" alt="" class="w-100">

</a>

</div>

</div>

<div class="row pt-2 pb-4">

<div class="col-6 offset-3 py-2">

<hr>

<p>

<span class="fw-bold">

<a href="/profile/{{ $post->user->id }}">

<span class="text-dark">{{ $post->user->username}}</span>

</a>

</span> {{ $post->caption }}

</p>

<hr>

@foreach($post->comments as $comment)

<p>

<span class="fw-bold">

<a href="/profile/{{ $post->user->id }}">

<span class="text-dark">{{ $comment->user->username}}</span>

</a>

</span>

{{ $comment->comment }}

</p>

@endforeach

</div>

</div>

@endforeach

<div class="row">

<div class="col-12 d-flex justify-content-center">

{{ $posts->links() }}

</div>

</div>

</div>-->

<div>

<div class="container d-flex justify-content-center p-0">

<h4 class="text-light"><strong>YOUR FEED</strong></h4>

</div>

<div class="container p-0">

<hr class="mt-2 mb-4">

</div>

<div class="container d-flex justify-content-center posts">

<div class="cont rounded m-0 p-0">

@foreach($posts as $post)

<div class="row pt-3 ps-4">

<div class="row">

<a href="/profile/{{ $post->user->profile->id }}">

<img src="{{ $post->user->profile->profileImage() }}" alt="" class="rounded-circle w-100" style="max-width: 80px;">

<span class="fw-bold ps-4">{{ $post->user->username}}</span>

</a>

</div>

</div>

<div class="row mx-0 p-0">

<div class="row pt-2 m-0">

<a href="/p/{{ $post->id }}">

<img src="/storage/{{ $post->media }}" alt="" class="img-fluid">

</a>

</div>

</div>

<div class="row pt-2 ps-4">

<div class="col-6 py-2 text-light">

<p class="mb-2">

<span class="fw-bold">

<a href="/profile/{{ $post->user->id }}">

<span>{{ $post->user->username}}</span>

</a>

</span> {{ $post->caption }}

</p>

@if($post->comments->count() > 0)

<p>

<a href="{{ route("posts.show", [$post, $comment->id]) }}">See all comments ({{ $post->comments->count() }})</a>

</p>

@endif

</div>

@endforeach

<div class="row">

<div class="col-12 d-flex justify-content-center">

{{ $posts->links() }}

</div>

</div>

</div>

</div>

</div>

</div>

@endsection

**//Publikāciju atrādīšanas lapa**

@extends('layouts.app')

@section('content')

<div class="container posts">

<div class="row">

<div class="col-8 p-0">

<img src="/storage/{{ $post->media }}" class="object-fit-cover w-100">

</div>

<div class="col-4 d-flex flex-column shadow p-3 rounded-right border-light">

<div class="d-flex align-items-center">

<div class="pe-3">

<a href="/profile/{{ $post->user->id }}"><img src="{{ $post->user->profile->profileImage() }}" alt="" class="rounded-circle w-100" style="max-width: 40px;"></a>

</div>

<div>

<div class="fw-bold">

<a href="/profile/{{ $post->user->id }}">

<span class="text-white">{{ $post->user->username}}</span>

</a>

<!--<a href="" class="follow-link ps-3">Follow</a>-->

</div>

</div>

</div>

<hr>

<div class="">

<p class="m-0 text-white">

<span class="fw-bold ">

<a href="/profile/{{ $post->user->id }}">

<span class="text-white">{{ $post->user->username}}</span>

</a>

</span> {{ $post->caption }}

</p>

</div>

<hr>

@foreach($post->comments as $comment)

<div class="d-flex align-items-center">

<small class="fs-6 fw-light text-white">{{ $comment->created\_at->format('Y-m-d H:i:s') }}</small>

@if(Auth::check() && Auth::user()->id == $comment->user->id)

<form action="{{ route("posts.comments.destroy", [$post, $comment->id]) }}" method="post">

@csrf

@method("delete")

<button class="btn btn-danger btn-sm p-1 ms-2">Delete</button>

</form>

@endif

</div>

<p class="text-white">

<span class="fw-bold">

<a href="/profile/{{ $post->user->id }}">

<span class="text-white">{{ $comment->user->username}}</span>

</a>

</span>

{{ $comment->comment }}

</p>

@endforeach

<form action="{{ route("posts.comments.store", $post->id) }}" method="post" class="mt-auto">

@csrf

<div class="form-group mt-auto">

<textarea name="comment" class="form-control mb-2" placeholder="Leave a comment"></textarea>

<div class="d-flex justify-content-between">

<button type="submit" class="btn btn-primary">Add Comment</button>

</div>

</div>

</form>

@if(Auth::check() && Auth::user()->id == $post->user->id)

<form action="{{ route("posts.destroy", $post->id) }}" method="post" class="mt-2">

@csrf

@method('delete')

<button class="btn btn-danger">Delete Post</button>

</form>

@endif

</div>

</div>

</div>

@endsection

**//Visas čata darbības**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\User;

use App\Models\Message;

use App\Models\Profile;

class ChatController extends Controller

{

public function index()

{

$authUser = auth()->user();

$users = $this->getConversationUsers($authUser->username);

return view('chat.index', compact('users'));

}

public function chatWithUser($id = null)

{

$authUser = auth()->user();

$authUserUsername = $authUser->username;

$selectedUser = null;

$selectedUsername = null; // Initialize $selectedUsername

$messages = collect();

$authUserId = $authUser->id;

$selectedUserId = $id;

// Fetching images

$authUserImage = $this->getProfileImage($authUserId);

$selectedUserImage = $this->getProfileImage($selectedUserId);

if ($id) {

$selectedUser = User::findOrFail($id);

$selectedUsername = $selectedUser->username;

// Fetch messages with the selected user

$messages = Message::where(function ($query) use ($authUserUsername, $selectedUsername) {

$query->where('message\_sender', $authUserUsername)

->where('message\_receiver', $selectedUsername);

})->orWhere(function ($query) use ($authUserUsername, $selectedUsername) {

$query->where('message\_sender', $selectedUsername)

->where('message\_receiver', $authUserUsername);

})->orderBy('created\_at', 'asc')->get()->toArray();

}

// Fetch users with whom the authenticated user has had conversations

$users = $this->getConversationUsers($authUserUsername);

// Add the selected user to the users list if not already in the list

if ($selectedUser && !$users->contains($selectedUser)) {

$users->push($selectedUser);

}

return view('chat.chatWithUser', compact('selectedUser', 'selectedUsername', 'messages', 'users', 'authUserImage', 'selectedUserImage'));

}

private function getConversationUsers($username)

{

$messages = Message::select('message\_sender', 'message\_receiver')

->where('message\_sender', $username)

->orWhere('message\_receiver', $username)

->get();

$users\_usernames = array();

foreach ($messages as $message) {

if ($message['message\_sender'] != $username && !in\_array($message['message\_sender'], $users\_usernames)) {

$users\_usernames[] = $message['message\_sender'];

}

if ($message['message\_receiver'] != $username && !in\_array($message['message\_receiver'], $users\_usernames)) {

$users\_usernames[] = $message['message\_receiver'];

}

}

return User::whereIn('username', $users\_usernames)->get();

}

private function getProfileImage($userId)

{

$profileImage = Profile::select('media')->where('user\_id', '=', $userId)->first();

if ($profileImage && $profileImage->media) {

return '/storage/' . $profileImage->media;

}

return '/storage/profile/Wuslb5XmCz5F1cHQMHhHAlqEKXEhlJta1Uv0xDze.jpg';

}

public function send\_message(Request $request) {

$authUserId = auth()->id();

$selectedUser = User::findOrFail($authUserId);

/\*$profile = Profile::where('user\_id', '=', $authUserId)->get();\*/

/\*$user = User::where('username' , '=', $request['message\_sender'])->get();\*/

$form\_fields = $request->validate([

'content' => 'required'

], [

'content.required' => 'Message text is required'

]);

$form\_fields['message\_sender'] = $request['message\_sender'];

$form\_fields['message\_receiver'] = $request['message\_receiver'];

$form\_fields['status'] = 'UNREAD';

Message::create($form\_fields);

$time = date("h:i A, d/m/Y", time());

return response()->json([

'name' => auth()->user()['username'],

'image' => $selectedUser->profile->profileImage(),

'content' => $request['content'],

'time' => $time,

'message\_sender' => $request['message\_sender']

]);

}

public function refresh\_messages(Request $request) {

$user = $request['user'];

// mark unread messages as read when user opens the chat

Message::where('message\_sender', '=', $user)->where('message\_receiver', '=', auth()->user()->username)->update(['status' => 'READ']);

return response()->json();

}

public function chat\_search(Request $request)

{

$query = $request->input('query');

$authUser = auth()->user();

if (empty($query)) {

// Return users with whom the authenticated user has had conversations

$users = $this->getConversationUsers($authUser->username);

} else {

// Search users by username, excluding the authenticated user and the admin

$users = User::where('username', 'like', '%' . $query . '%')

->where('username', '!=', $authUser->username)

->where('isAdmin', '!=', '1')

->get();

}

$users = $users->map(function ($user) {

$user->profile\_image = $user->profile->profileImage() ?? '/storage/profile/Wuslb5XmCz5F1cHQMHhHAlqEKXEhlJta1Uv0xDze.jpg';

return $user;

});

return response()->json($users);

}

}

**//Čata galvenā lapa ar lietotāju meklēšanu, izmantojot AJAX**

@extends('layouts.app')

@section('content')

<div class="custom-container text-light">

<div class="custom-sidebar">

<div class="custom-card">

<div class="custom-card-body">

<h2 class="custom-card-title">User List</h2>

<input type="text" class="custom-search-input" id="user-search" placeholder="Search users...">

<ul class="custom-user-list">

<!-- Example of dynamic content -->

@foreach($users as $user)

@php

$isSelected = isset($selectedUser) && $user->id == $selectedUser->id;

@endphp

<li class="custom-user-item custom-selected-user">

<a href="{{ route('chatWithUser', $user->id) }}" class="custom-user-link">

<img src="{{ $user->profile->profileImage() }}" alt="{{ $user->username }}" class="custom-user-avatar">

<div>

<p>{{ $user->username }}</p>

</div>

</a>

</li>

@endforeach

<!-- More user items can be added here -->

</ul>

</div>

</div>

</div>

<div class="custom-main">

<div class="custom-card">

<div class="custom-card-header">

<h2 class="custom-card-title">Chat</h2>

</div>

<div class="custom-card-body2 custom-chat-messages2" id="chat\_window">

<p>Choose user to start conversation</p>

</div>

<div class="custom-card-footer custom-chat-input">

<!-- Input elements can be added here -->

</div>

</div>

</div>

</div>

<script>

document.addEventListener('DOMContentLoaded', function() {

const searchInput = document.getElementById('user-search');

const userList = document.querySelector('.custom-user-list');

searchInput.addEventListener('input', function() {

const query = searchInput.value;

fetch('/chat\_search', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').getAttribute('content')

},

body: JSON.stringify({ query: query })

})

.then(response => response.json())

.then(data => {

userList.innerHTML = '';

if (data.length > 0) {

data.forEach(user => {

const userItem = document.createElement('li');

userItem.classList.add('custom-user-item');

const userLink = document.createElement('a');

userLink.href = `/chat/${user.id}`;

userLink.classList.add('custom-user-link');

const userAvatar = document.createElement('img');

userAvatar.src = user.profile\_image; // Assuming a default image if none is provided

userAvatar.alt = user.username;

userAvatar.classList.add('custom-user-avatar');

const userInfo = document.createElement('div');

const userName = document.createElement('p');

userName.textContent = user.username;

userInfo.appendChild(userName);

userLink.appendChild(userAvatar);

userLink.appendChild(userInfo);

userItem.appendChild(userLink);

userList.appendChild(userItem);

});

} else {

const noUserItem = document.createElement('li');

noUserItem.classList.add('custom-user-item');

noUserItem.textContent = 'No users found';

userList.appendChild(noUserItem);

}

})

.catch(error => console.error('Error:', error));

});

});

</script>

@endsection

**//Čats ar izvēlēto lietotāju**

@extends('layouts.app')

@section('content')

<!--<div class="container mt-5">

<div class="row">

<div class="col-md-4">

<div class="card">

<div class="card-body" style="background-color: #1E1E24;">

<h2 class="card-title text-light">User List</h2>

<input type="text" class="form-control mb-3" id="user-search" placeholder="Search users...">

<ul class="list-group">

@foreach($users as $user)

@php

$isSelected = isset($selectedUser) && $user->id == $selectedUser->id;

@endphp

<li class="list-group-item d-flex align-items-center {{ $isSelected ? 'selected-user' : '' }}" style="background-color: #25262E">

<a href="{{ route('chatWithUser', $user->id) }}" class="d-flex align-items-center text-decoration-none text-light">

<img src="{{ $user->profile->profileImage() }}" alt="{{ $user->username }}" class="rounded-circle me-3" style="width: 40px; height: 40px;">

<div>

<p class="mb-0">{{ $user->username }}</p>

</div>

</a>

</li>

@endforeach

</ul>

</div>

</div>

</div>

<div class="col-md-8">

<div class="card">

<div class="card-header d-flex align-items-center text-light" style="background-color: #1E1E24;">

@if(isset($selectedUser))

<img src="{{ $selectedUser->profile->profileImage() }}" alt="{{ $selectedUser->username }}" class="rounded-circle me-3" style="width: 40px; height: 40px;">

<h2 class="card-title mb-0">{{ $selectedUser->username }}</h2>

@else

<h2 class="card-title mb-0">Chat</h2>

@endif

</div>

<div class="card-body chat-messages text-light m-0" id="chat\_window" style="background-color: #25262E">

@if(isset($messages) && sizeof($messages) > 0)

@foreach($messages as $message)

@php

$fmt = new IntlDateFormatter( "en\_GB" ,IntlDateFormatter::FULL, IntlDateFormatter::FULL, null,IntlDateFormatter::GREGORIAN, "LLLL d, yyyy HH:mm");

$message\_sent = ucfirst(datefmt\_format($fmt, strtotime($message['created\_at'])));

@endphp

@if ($message['message\_sender'] == $selectedUsername)

<div class="{{ $message['message\_sender'] == auth()->id() ? 'text-right' : 'text-left' }}">

<img src="{{$selectedUserImage}}" style="max-width: 40px;" class="rounded-circle">

<strong>{{$message['message\_sender']}}</strong>

<p>{{ $message\_sent}}</p>

<p>{{ $message['content'] }}</p>

</div>

@else

<div class="{{ $message['message\_sender'] == auth()->id() ? 'text-right' : 'text-left' }}">

<img src="{{$authUserImage}}" style="max-width: 40px;" class="rounded-circle">

<strong>{{$message['message\_sender']}}</strong>

<p class="mb-2">{{ $message\_sent }}</p>

<p>{{ $message['content'] }}</p>

</div>

@endif

@endforeach

@else

<p>Be the first to send a message!</p>

@endif

</div>

<div class="card-footer chat-input" style="background-color: #1E1E24;">

@if(isset($selectedUser))

<form id="message\_form" action="{{ route('chat.send\_message') }}" method="POST" class="d-flex">

@csrf

<input type="hidden" name="message\_sender" value='{{ auth()->user()->username }}'>

<input type="hidden" name='message\_receiver' value='{{ $selectedUser->username }}'>

<textarea id="new\_message\_text" name="content" rows="3" form="message\_form"></textarea>

<button type="submit" class="btn btn-primary">Send</button>

</form>

@endif

</div>

</div>

</div>

</div>

</div>-->

<div class="custom-container text-light">

<div class="custom-sidebar">

<div class="custom-card">

<div class="custom-card-body">

<h2 class="custom-card-title">User List</h2>

<input type="text" class="custom-search-input" id="user-search" placeholder="Search users...">

<ul class="custom-user-list">

@foreach($users as $user)

@php

$isSelected = isset($selectedUser) && $user->id == $selectedUser->id;

@endphp

<li class="custom-user-item {{ $isSelected ? 'custom-selected-user' : '' }}">

<a href="{{ route('chatWithUser', $user->id) }}" class="custom-user-link">

<img src="{{ $user->profile->profileImage() }}" alt="{{ $user->username }}" class="custom-user-avatar">

<div class="custom-user-info">

<p class="custom-user-name">{{ $user->username }}</p>

</div>

</a>

</li>

@endforeach

</ul>

</div>

</div>

</div>

<div class="custom-main">

<div class="custom-card">

<div class="custom-card-header">

@if(isset($selectedUser))

<img src="{{ $selectedUser->profile->profileImage() }}" alt="{{ $selectedUser->username }}" class="custom-user-avatar">

<h2 class="custom-card-title">{{ $selectedUser->username }}</h2>

@else

<h2 class="custom-card-title">Chat</h2>

@endif

</div>

<div class="custom-card-body custom-chat-messages" id="chat\_window">

@if(isset($messages) && sizeof($messages) > 0)

@foreach($messages as $message)

@php

$fmt = new IntlDateFormatter( "en\_GB" ,IntlDateFormatter::FULL, IntlDateFormatter::FULL, null,IntlDateFormatter::GREGORIAN, "LLLL d, yyyy HH:mm");

$message\_sent = ucfirst(datefmt\_format($fmt, strtotime($message['created\_at'])));

@endphp

@if ($message['message\_sender'] == $selectedUsername)

<div class="custom-message custom-message-sender">

<div class="custom-message-content">

<img src="{{$selectedUserImage}}" class="custom-message-avatar">

<strong>{{$message['message\_sender']}}</strong>

<p>{{ $message\_sent}}</p>

<p>{{ $message['content'] }}</p>

</div>

</div>

@else

<div class="custom-message custom-message-receiver">

<div class="custom-message-content">

<img src="{{$authUserImage}}" class="custom-message-avatar">

<strong>{{$message['message\_sender']}}</strong>

<p>{{ $message\_sent }}</p>

<p>{{ $message['content'] }}</p>

</div>

</div>

@endif

@endforeach

@else

<p>Be the first to send a message!</p>

@endif

</div>

<div class="custom-card-footer custom-chat-input">

@if(isset($selectedUser))

<form id="message\_form" action="{{ route('chat.send\_message') }}" method="POST" class="custom-message-form">

@csrf

<input type="hidden" name="message\_sender" value='{{ auth()->user()->username }}'>

<input type="hidden" name='message\_receiver' value='{{ $selectedUser->username }}'>

<textarea id="new\_message\_text" name="content" rows="3" form="message\_form"></textarea>

<button type="submit" class="custom-send-button">Send</button>

</form>

@endif

</div>

</div>

</div>

</div>

<script>

$(function() {

$("#chat\_window").scrollTop($("#chat\_window")[0].scrollHeight);

// Ajax request for sending a message

$('#message\_form').submit(function(event) {

var form\_id = $(this).attr('id');

event.preventDefault();

var formData = $(this).serializeArray();

$.ajax({

url: `{{ route('chat.send\_message') }}`,

type: 'post',

data: formData,

success: function(response) {

var content = response.content;

content = content.replace(/\n/g, "<br />");

var image = response.image;

var name = response.name;

var time = response.time;

var message\_sender = response.message\_sender;

var message = ` <div class="{{ `+message\_sender+` == auth()->id() ? 'text-right' : 'text-left' }}">

<img src="`+image+`" style="max-width: 40px;" class="rounded-circle">

<strong>`+message\_sender+`</strong>

<p>`+content+` </p>

</div>`;

$('#chat\_window').append(message);

$("#chat\_window").scrollTop($("#chat\_window")[0].scrollHeight);

$('#new\_message\_text').val('');

},

error: function(response) {

alert('error');

}

});

});

// Refreshing the page for possible messages every 3 seconds

var intervalId = window.setInterval(function() {

refresh\_chat();

}, 3000);

// Ajax request for refreshing the chat window

function refresh\_chat() {

$.ajax({

url: `{{ route('refresh\_messages') }}`,

type: 'post',

headers: {

'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')

},

data: {user: `{{ $selectedUsername }}`},

success: function(response) {

$('#chat\_window').load(' #chat\_window > \*');

var chat\_window = document.getElementById('chat\_window');

var scroll = false;

//console.log(response.user);

if (Math.abs(chat\_window.scrollTop - (chat\_window.scrollHeight - chat\_window.offsetHeight)) <= 5) {

scroll = true;

}

// a small delay so that element could properly render

setTimeout(() => {

if (scroll) {

var div = document.querySelector('#chat\_window');

div.scrollTop = div.scrollHeight;

}

}, 1000);

},

error: function(response) {

alert('error');

}

});

}

});

document.addEventListener('DOMContentLoaded', function() {

const searchInput = document.getElementById('user-search');

const userList = document.querySelector('.custom-user-list');

searchInput.addEventListener('input', function() {

const query = searchInput.value;

fetch('/chat\_search', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').getAttribute('content')

},

body: JSON.stringify({ query: query })

})

.then(response => response.json())

.then(data => {

userList.innerHTML = '';

if (data.length > 0) {

data.forEach(user => {

const userItem = document.createElement('li');

userItem.classList.add('custom-user-item');

const userLink = document.createElement('a');

userLink.href = `/chat/${user.id}`;

userLink.classList.add('custom-user-link');

const userAvatar = document.createElement('img');

userAvatar.src = user.profile\_image; // Assuming a default image if none is provided

userAvatar.alt = user.username;

userAvatar.classList.add('custom-user-avatar');

const userInfo = document.createElement('div');

const userName = document.createElement('p');

userName.textContent = user.username;

userInfo.appendChild(userName);

userLink.appendChild(userAvatar);

userLink.appendChild(userInfo);

userItem.appendChild(userLink);

userList.appendChild(userItem);

});

} else {

const noUserItem = document.createElement('li');

noUserItem.classList.add('custom-user-item');

noUserItem.textContent = 'No users found';

userList.appendChild(noUserItem);

}

})

.catch(error => console.error('Error:', error));

});

});

</script>

</script>

@endsection

**//Visas komentāru darbības**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\Post;

use App\Models\Comment;

use App\Models\User;

class CommentsController extends Controller

{

public function store(Post $post)

{

$user = auth()->user();

$comment = new Comment();

$comment->post\_id = $post->id;

$comment->user\_id = $user->id;

$comment->comment = request()->get('comment');

$comment->save();

return redirect()->route('posts.show', $post->id)->with('success', "Comment posted successfully!");

}

public function destroy(Post $post, $id)

{

$comment = Comment::where('id', $id)->firstOrFail();

$comment->delete();

return redirect()->route("posts.show", $post)->with("success", "Comment deleted successfully!");

}

}

**//Visas sekošanu darbības**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\User;

class FollowsController extends Controller

{

public function \_\_construct()

{

$this->middleware('auth');

}

public function store(User $user)

{

return auth()->user()->following()->toggle($user->profile);

}

}

**//Vue.js komponents priekš sekošanas**

<template>

<div class="container">

<button class="btn allBtn ms-3" @click="followUser" v-text="buttonText"></button>

</div>

</template>

<script>

export default {

props: ['userId', 'follows'],

data() {

return {

status: this.follows,

};

},

methods: {

followUser() {

axios.post('/follow/' + this.userId)

.then(response => {

this.status = !this.status;

this.$emit('update-follow-status', { userId: this.userId, follows: this.status });

})

.catch(errors => {

if (errors.response.status === 401) {

window.location = '/login';

}

});

}

},

computed: {

buttonText() {

return this.status ? 'Unfollow' : 'Follow';

}

},

watch: {

follows(newVal) {

this.status = newVal;

}

}

};

</script>

**//Vue.js lietotāju meklēšana**

<template>

<div class="container">

<div class="form-group">

<h4>Type by Username</h4>

<input type="text" v-model="searchQuery" placeholder="Search the user..." class="form-control">

<p id="search-message" class="search-message text-light">Start typing to search for users.</p>

</div>

<div v-if="results.length > 0 || !searchQuery">

<div class="row">

<div class="col-md-4" v-for="(result, index) in results" :key="index">

<div class="card mb-4" style="background-color: #1E1E24">

<div class="card-body text-center">

<a :href="'/profile/' + result.id">

<img :src="result.media" class="rounded-circle mb-3" style="max-width: 60px;">

</a>

<h5 class="card-title">

<a :href="'/profile/' + result.id" class="text-decoration-none text-light">

{{ result.username }}

</a>

</h5>

<!--<follow-button :user-id="result.id" :follows="result.follows"></follow-button>-->

<a :href=" 'chat/' + result.id" class="btn allBtn">Message</a>

</div>

</div>

</div>

</div>

</div>

<div class="col-12 text-light" v-else>

<p>No Results Found</p>

</div>

</div>

</template>

<script>

export default {

mounted() {

console.log("Component mounted.");

this.getResults();

},

data() {

return {

searchQuery: null,

results: [],

};

},

methods: {

getResults() {

let url = this.searchQuery ? `/search/${this.searchQuery}` : '/allUsers';

axios.get(url)

.then(response => {

this.results = response.data.map(user => ({

id: user.id,

username: user.username,

media: user.media

}));

})

.catch(error => {

console.error('Error fetching users:', error);

});

},

},

watch: {

searchQuery(after, before) {

this.getResults();

}

}

}

</script>

**//Routes jeb maršruti**

<?php

use Illuminate\Support\Facades\Route;

use Illuminate\Support\Facades\Mail;

use App\Http\Controllers\ChatController;

/\*

|--------------------------------------------------------------------------

| Web Routes

|--------------------------------------------------------------------------

|

| Here is where you can register web routes for your application. These

| routes are loaded by the RouteServiceProvider and all of them will

| be assigned to the "web" middleware group. Make something great!

|

\*/

Auth::routes();

Route::get('/email', function () {

return new NewUserWelcomeMail();

});

Route::post('p/{post}/comments', 'App\Http\Controllers\CommentsController@store')->name('posts.comments.store');

Route::delete('p/{post}/comments/{id}', 'App\Http\Controllers\CommentsController@destroy')->name('posts.comments.destroy');

Route::post('follow/{user}', 'App\Http\Controllers\FollowsController@store')->name('profile.follow');

Route::get('/', 'App\Http\Controllers\PostsController@index')->name('posts.index');

Route::get('/p/create', 'App\Http\Controllers\PostsController@create');

Route::post('/p', 'App\Http\Controllers\PostsController@store')->name('posts.store');

Route::get('/p/{post}', 'App\Http\Controllers\PostsController@show')->name('posts.show');

Route::delete('/p/{post}', 'App\Http\Controllers\PostsController@destroy')->name('posts.destroy');

Route::get('/search', 'App\Http\Controllers\ProfilesController@search')->name('profile.search');

Route::get('/allUsers', 'App\Http\Controllers\ProfilesController@allUsers');

Route::get('/search/{query}', 'App\Http\Controllers\ProfilesController@find')->name('profile.find');

Route::get('/profile/{user}', 'App\Http\Controllers\ProfilesController@index')->name('profile.show');

Route::get('/profile/{user}/edit', 'App\Http\Controllers\ProfilesController@edit')->name('profile.edit');

Route::patch('/profile/{user}', 'App\Http\Controllers\ProfilesController@update')->name('profile.update');

Route::middleware('auth')->group(function(){

Route::resource('users', App\Http\Controllers\AdminController::class);

Route::get('/dashboard', 'App\Http\Controllers\AdminController@dashboard')->name('admin.dashboard');

Route::get('/users', 'App\Http\Controllers\AdminController@users')->name('admin.users');

Route::get('/users/status/{user\_id}/{status\_code}', 'App\Http\Controllers\AdminController@updateStatus')->name('users.status.update');

Route::get('/export\_users\_pdf', [App\Http\Controllers\AdminController::class, 'exportUsersPdf'])->name('admin.export\_users\_pdf');

Route::get('/export\_dashboard\_pdf', [App\Http\Controllers\AdminController::class, 'exportDashboardPdf'])->name('admin.export\_dashboard\_pdf');

});

Route::middleware(['auth', 'verified'])->get('/chats', [ChatController::class, 'index'])->name('chats');

Route::middleware(['auth', 'verified'])->get('/chat/{id}', [ChatController::class, 'chatWithUser'])->name('chatWithUser');

Route::middleware(['auth', 'verified'])->post('/send\_message', [ChatController::class, 'send\_message'])->name('chat.send\_message');

Route::middleware(['auth', 'verified'])->post('/refresh\_messages', [ChatController::class, 'refresh\_messages'])->name('refresh\_messages');

Route::middleware(['auth', 'verified'])->post('/chat\_search', [ChatController::class, 'chat\_search'])->name('chat.chat\_search');