**Profesionālās izglītības kompetences centrs   
„Rīgas Valsts tehnikums”**

Izglītības programma: Programmēšana

**KVALIFIKĀCIJAS DARBS**

**Sludinājumu un izsoļu portāls retro automobiļu cienītājiem**

Paskaidrojošais raksts 187 lpp.

Audzēknis: P. Žuravļovs

Vadītājs: I. Litvjakovs

Normu kontrole: O. Sabanska

**Rīga**

**2018**

**anotācija**

Kvalifikācijas darbā ir aprakstīts sludinājumu un izsoļu portāls retro automobiļu cienītājiem izstrādāšanas process. Sistēma dod iespēju apmainīties un nopirkt detaļas, izmantojot sludinājumu formu, piedalīties labdarībās un komerciālās izsolēs, kuros lietotāji krāj naudu uz eksemlpāra atjaunošanu vai izspēlē vienu no tiem klasiskajā izsolē, ka arī komunicēt platformas ietvaros, izmantojot forumu. Sistēma tika izstrādāta ar PHP programmēšanas valodu, izmantojot Laravel 6.18.3 ietvaru, JavaScript programmēšanas valodu, izmantojot VueJS 2.6.11 ietvaru, un MySQL relācijas datubāzes vadības sistēmu.

Kvalifikācijas darbs sevi ietver ievadu, uzdevuma nostādni, prasību specifikāciju, risināšanas līdzekļu izvēles pamatojumu, programatūras produkta modelēšanas un projektēšanas aprakstu, datu struktūru aprakstu, lietotajā ceļvedi, nobeigumu, informācijas avotus un pielikumus. Kvalifikācijas darba ievadā ir aprakstīts retro automobiļu atjaunošanas sarežģitības aktuālā problēmā un tās risinājums. Uzdevumu nostādnē ir noradīti uzdevumi, kurus sistēmai būs nepieciešami izpildīt. Prasību specifikācijā sastāv no iejās un izejas informācijas, kā arī no funkcionālām un nefunkcionālam prasībām. Uzdevuma risināšanas līdzekļu izvēles pamatojumā ir norādīti, kādi un kāpēc līdzekļi tiks izmantoti sistēmas izstrādāšanā. Programatūras produkta modelēšanas un projektēšanas apraksts sastāv no struktūras modeļa, kas sevī ietvēr sistēmas arhitektūru un ER modeli, funkcionālās sistēmas modeļa, kas iekļauj datu plūsmu modeli. Datu struktūru aprakstā tiek aprakstīta datubāzes tabulu struktūra, kas ietvēr datu tipus un to garumus, un datubāzes tabulu relācijas shēmu. Lietotajā ceļvedī ir nodarītas nepieciešamās sistēmas prasības aparatūrai un programmatūrai, sistēmas instalācija un palaišana, kā arī rogrammas apraksts, kas paskaidro, kā pareizi lietot sistēmu. Testa piemērā ir dots detalizēts sludinājuma pievienošanas aprakst, kurš ir sadalīts vairākos posmos.

Kvalifikācijas darbs sastāv no 187 lapaspusēm, kurā ietilpst 59 attēli, 17 tabulas un 25 pielikumi. Pielikumi satur ER diagrammu , funkcionālas dekompozīcijas diagrammu, datu plūsmu diagrammas, datubāzes tabulu relācijas un programmas pirmkodu

**АННОТАЦИЯ**

В квалификационной работе описывается процесс разработки рекламно-аукционного портала для любителей ретро-автомобилей. Система позволяет обмениваться и покупать запчасти, используя форму объявлений, участвовать в благотворительных и коммерческих аукционах, где пользователи собирают деньги на реставрацию экземпляра аукциона и разыгрывают один из них на классическом аукционе, а также общаться в рамках платформы через форум. Система была разработана с использованием языка программирования PHP, используя фреймворк Laravel 6.18.3, языка программирования JavaScript, используя фреймворк VueJS 2.6.11 и системы управления реляционными базами данных MySQL.

Квалификационная работа включает введение, постановку задачи, спецификацию требований, обоснование выбора средств для разработки, описание моделирования и проектирования программного продукта, описание структур данных, руководство по использованию, заключение, источники информации и приложения. Во введении к квалификационной работе описывается осложнения модернизации и реставрации ретро-автомобилей и ее решение. Постановка задачи определяет инструкции, которые система должна будет выполнять. Спецификация требований состоит из входной и выходной информации, а также функциональных и нефункциональных требований. Обоснование выбора средств для разработки оисывает, какие и почему средства будут использоваться при разработке системы. Описание моделирования и проектирования программного продукта состоит из модели структуры, которая включает в себя архитектуру системы и модель ER, модель функциональной системы, которая включает модели потока данных. Описание структур данных описывает структуру таблиц базы данных, которая включает в себя типы данных и их длину, а также реляционную схему таблиц базы данных. Руководство пользователя содержит необходимые системные требования для аппаратного и программного обеспечения, установки и запуска системы, а также описание программы, которая объясняет, как правильно использовать систему. Тестовый пример дает подробное описание добавления обхявлений, которая разбита на несколько этапов.

Квалификационная работа состоит из 187 страниц, которая включает в себя 59 картинок, 17 таблиц и 25 приложений. Приложения содержат ER диаграмму, диаграмму функциональной декомпозиции, диаграммы потоков данных, связи таблиц базы данных и исходный код программы.

**Saturs**

[**Ievads** 6](#_Toc41841875)

[**1.** **Uzdevuma nostādne** 7](#_Toc41841876)

[**2.** **Prasību specifikācija** 8](#_Toc41841877)

[2.1. Ieejas informācijas apraksts 8](#_Toc41841878)

[2.2. Izejas informācijas apraksts 9](#_Toc41841879)

[2.3. Arējas informācijas apraksts 10](#_Toc41841880)

[2.4. Funkcionālās prasības 10](#_Toc41841881)

[2.5. Nefunkcionālās prasības 13](#_Toc41841882)

[**3.** **Uzdevuma risināšanas līdzekļu izvēles pamatojums** 15](#_Toc41841883)

[3.1. Uzdevuma analīze 15](#_Toc41841884)

[3.2. Rīku analīze 15](#_Toc41841885)

[3.3. Izmantojamie spraudņi 16](#_Toc41841886)

[**4.** **Programmatūras produkta modelēšana un projektēšana** 18](#_Toc41841887)

[4.1. Sistēmas struktūras modelis 18](#_Toc41841888)

[**4.1.1.** ***Sistēmu arhitektūra*** 18](#_Toc41841889)

[**4.1.2.** ***Sistēmas ER modelis*** 18](#_Toc41841890)

[4.2. Funkcionālais sistēmas modelis 20](#_Toc41841891)

[**4.2.1.** ***Datu plūsmu modelis*** 20](#_Toc41841892)

[**5.** **Datu struktūru apraksts** 28](#_Toc41841893)

[5.1. Datubāzes aprakstītas tabulas 28](#_Toc41841894)

[**6.** **Lietotāja ceļvedis** 33](#_Toc41841895)

[6.1. Sistēmas prasības aparatūrai un programmatūrai 33](#_Toc41841896)

[6.2. Sistēmas instalācija un palaišana 33](#_Toc41841897)

[**6.2.1.** ***Datubāze*** 34](#_Toc41841898)

[**6.2.2.** ***PayPal*** 35](#_Toc41841899)

[6.3. Programmas apraksts 36](#_Toc41841900)

[6.4. Testa piemērs 54](#_Toc41841901)

[**7.** **Nobeigums** 56](#_Toc41841902)

[**8.** **Informācijas avoti** 57](#_Toc41841903)

[**Pielikumi** 58](#_Toc41841904)

[**1. pielikums** 59](#_Toc41841905)

[**2. pielikums** 60](#_Toc41841906)

[**3. pielikums** 61](#_Toc41841907)

[**4. pielikums** 62](#_Toc41841908)

[**5. pielikums** 63](#_Toc41841909)

[**6. pielikums** 64](#_Toc41841910)

[**7. pielikums** 65](#_Toc41841911)

[**8. pielikums** 66](#_Toc41841912)

[**9. pielikums** 67](#_Toc41841913)

[**10. pielikums** 68](#_Toc41841914)

[**11. pielikums** 69](#_Toc41841915)

[**12. pielikums** 70](#_Toc41841916)

[**13. pielikums** 71](#_Toc41841917)

[**14. pielikums** 72](#_Toc41841918)

[**15. pielikums** 73](#_Toc41841919)

[**16. pielikums** 74](#_Toc41841920)

[**17. pielikums** 75](#_Toc41841921)

[**18. pielikums** 76](#_Toc41841922)

[**19. pielikums** 77](#_Toc41841923)

[**20. pielikums** 78](#_Toc41841924)

[**21. pielikums** 79](#_Toc41841925)

[**22. pielikums** 80](#_Toc41841926)

[**23. pielikums** 81](#_Toc41841927)

[**24. pielikums** 82](#_Toc41841928)

[**25. pielikums** 83](#_Toc41841929)

**Ievads**

Mūsdienu jau ir grutāk attīstīt un atbalstīt cilvēka pagātnes mantojumu, īpaši kad tas attiecas uz tehniskiem ierīcem, jo ražošanas uzņēmumi un rūpnīcas vairs neeksistē vai neražo to konkrētu produktu novecojušo tehnoloģiju dēļ, vai nomainīja temātu un ražošanas virzienu.

Veci automobiļi pieder pie tām tehnoloģijam, tāpēc ka ar laiku automobiļu industrija ļoti ātri attīstas un tehnoloģijas, kas bija domāti izmainīt šī produkicjas nākotni, tiek aizvietoti ar jauniem, moderniem risinājumiem, bet vairāki automašīnas cienītāji līdz šīm brīdim joprojam mēģinā popularizēt šo virzienu, veidojot savus atjaunošanas un modernizācijas projektus, izmantojot automobiļus no 80-tiem, 50-tiem, kā arī no 30-tiem gadiem XX gadsimtā, taču atrast nepieciešamas detaļas ir grutāk un grutāk.

Viens no galvēniem principiem, pēc kuras būtu iespējams savākt tādus cilvēkus un uzturēt viņus saderinātus šajā sfērā, ir ne tikai nepieciešamu rīku izveidošana, kas darbojas kā galvenais veids detaļu apmainīšanā un izsoles piedalīšanās, bet arī nodrošināt iespēju komunicēt ar citiem līdzīgi domājošiem cilvēkiem.

Šī projekta mērķis ir izveidot sistēmu, kurā jebkurš cilvēks varēs apmainīties un nopirkt mašīnas detaļas ar sludinājumu formu palīdzību, kā arī palīdzēt uzkrāt nepieciešamu materiālu atbalstu priekš atjaunošanas eksemplāriem un brīvi komunicēt foruma ietvaros. Lietotājam ir iespēja pieteikties sistēmā un veikt tranzākcijas, izmantojot PayPal kontu.

# **Uzdevuma nostādne**

Projekta galvēnais uzdevums ir uztaisīt platformu, kura orientējas uz automašīnas cienītājiem, kura sevī ietvēr: tirgus, kas dod iespēju lietotājiem apmainīties / nopirkt mašīnas detaļas, izsoli, kas palīdz ne tikai atbalstīt projektu, bet arī atjaunot ļoti retus automašīnas eksemplārus, ka arī sociālo pusi, kura ļauj cilvēkiem komunicēt platformas ietvaros. Platforma orientējas vairāk uz retro automašīnu īpašniekiem, kuriem ir iespēja piedalīties kopīgā eksemplārā atjaunošanā un detaļu / donoru automašīnu apmaiņā ar citiem lietotājiem.

Platformai ir realizēta sekojoša funkcionalitāte no lietotāja puses:

* reģistrācija / pieteikšanas;
* sava profila pārvalde;
* sludinājumu izveidošana;
* komunicēšana platformas ietvaros, izmantojot forumu;
* piedalīšanas izsolē.

Platformai ir realizēta sekojoša funkcionalitāte no administrācijas puses:

* lietotāju pārvalde;
* foruma pārvalde;
* sludinājumu pārvalde;
* izsoļu pārvalde.

# **Prasību specifikācija**

## Ieejas informācijas apraksts

Sistēmā tiks nodrošinātā šāda ieejas informācijas apstrāde:

1. Informācija par lietotāju reģistrācijas / profila rediģēšanas laikā sastāv no sekojošiem datiem:

* Lietotājvārds – ciparu un simbolu kombinācijas teksts ar izmēru līdz 16 rakstzīmēm.
* Vārds – burtu teksts ar izmēru līdz 24 rakstzīmēm.
* Uzvārds – burtu teksts ar izmēru līdz 24 rakstzīmēm.
* E-pasts – ciparu un simbolu kombinācijas teksts ar izmēru līdz 255 rakstzīmēm.
* Parole – ciparu un simbolu kombinācijas teksts ar izmēru līdz 16.
* Dzimums – 2 variantu radiopogas.
* Bilde – lietotāja attēlojums (tikai rediģēšanas laikā).

1. Informācija par lietotāju pieteikšanas laikā sastāv no sekojošiem datiem:

* Lietotājvārds – ciparu un simbolu kombinācijas teksts ar izmēru līdz 16 rakstzīmēm.
* E-pasts – ciparu un simbolu kombinācijas teksts ar izmēru līdz 16 rakstzīmēm.
* Parole – ciparu un simbolu kombinācijas teksts ar izmēru līdz 16 rakstzīmēm.

1. Informācija par sludinājuma filtrāciju sastāv no sekojošiem datiem:

* Sludinājuma virsraksts – ciparu un simbolu kombinācijas teksts.
* Sludinājuma saturs – ciparu un simbolu kombinācijas teksts.
* Sludinājuma detaļas nosaukums – ciparu un simbolu kombinācijas teksts.
* Sludinājuma / detaļas tips – nolaižamā izvēlne.
* Sludinājuma izveidošanas datums – nolaižamā izvēlne.

1. Informācija par sludinājuma pievienošanu / rediģēšanu sastāv no sekojošiem datiem:

* Virsraksts – ciparu un simbolu kombinācijas teksts ar izmēru līdz 32 rakstzīmēm.
* Saturs – ciparu un simbolu kombinācijas teksts ar izmēru līdz 600 rakstzīmēm.
* Detaļas nosaukums – ciparu un simbolu kombinācijas teksts ar izmēru līdz 32 rakstzīmēm.
* Detaļas kategorija – nolaižamā izvēlne.
* Bilde(s) – pielikums (fails(-i)) sludinājumam.
* Tālrunis – veselu skaitļu teksts no 6 līdz 12 rakstzīmēm.

1. Informācija par lietotāju izsoles pieteikšanas laikā sastāv no sekojošiem datiem:

* Naudu apjoms – decimālu skaitļu teksts (+cipari pēc komata).

1. Informācija par ziņu pievienošanu forumā sastāv no sekojošiem datiem:

* Virsraksts – ciparu un simbolu kombinācijas teksts ar izmēru līdz 32 rakstzīmēm.
* Saturs – ciparu un simbolu kombinācijas teksts ar izmēru līdz 600 rakstzīmēm

1. Informācija par komentāru pievienošanu ziņai sastāv no sekojošiem datiem:

* Saturs – ciparu un simbolu kombinācijas teksts ar izmēru līdz 600 rakstzīmēm.

1. Informācija par labdarības izsoles pievienošanu sastāv no sekojošiem datiem:

* Izsoles eksemplāra nosaukums – ciparu un simbolu kombinācijas teksts ar izmēru līdz 24 rakstzīmēm.
* Izsoles eksemplāra bilde – pielikums (fails).
* Uzkrāšanas naudas apjoms – decimālu skaitļu teksts (+2 cipari pēc komata).

1. Informācija par komerciālas izsoles pievienošanu sastāv no sekojošiem datiem:

* Izsoles eksemplāra nosaukums – ciparu un simbolu kombinācijas teksts ar izmēru līdz 24 rakstzīmēm.
* Izsoles eksemplāra bilde – pielikums (fails).
* Sākuma cena – decimālu skaitļu teksts (+2 cipari pēc komata).
* Izsoles beigšanas datums – ciparu un simbolu kombinācijas teksts atbilstoši datuma formātam.

## Izejas informācijas apraksts

1. Izsoles atspoguļojums. Šeit būs redzams izsoles informācija, ka arī visi izsoles dalībnieki un viņu ieguldījums dotājā izsolē. Izsoles dalībnieki tika sakartoti pēc ieguldījuma apjoma.
2. Izsoles pieteikšanas informācijas izvade. Šeit lietotājam būs iespēja pārbaudīt informāciju par izsoli, kurā viņs grib pieteikties, iekļaujot izsoles eksemplāru un velāmo nauda apjomu.
3. Sludinājumu meklēšanas aspoguļojums. Šeit lietotājam būs iespēja atspoguļot sludinājumus pēc izvēlētājiem filtrācijas laukiem un izvēlnem.
4. Izsoles rezultātu izvade. Šeit lietotājiem būs pieejama izsoles vēsture, kura atspoguļo informāciju par izsolēm un uzvarētājiem.
5. Statistikas izvade. Lietotājiem ir iespēja aplūkot sludinājumu un izsoles piedalīšanas vēsturi sava profilā.

## Arējas informācijas apraksts

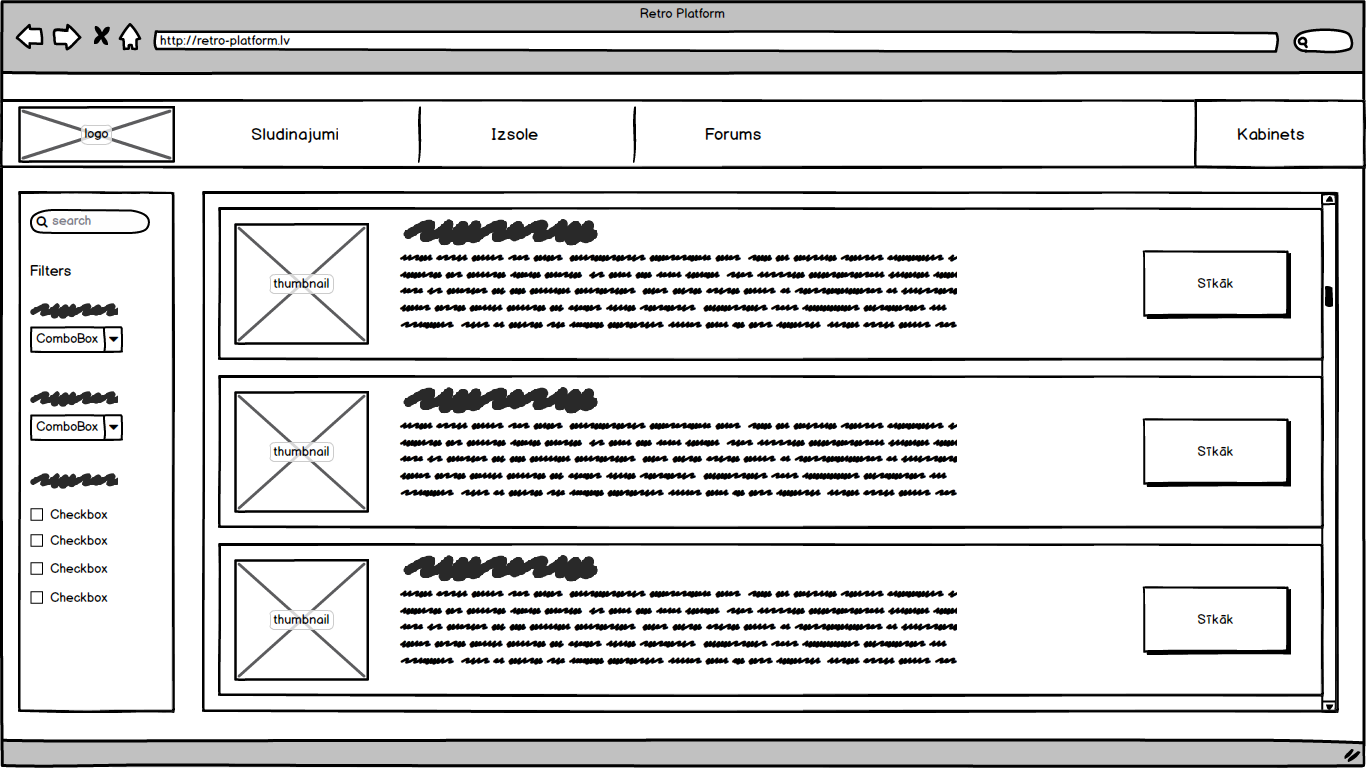
1. PayPal API. Sistēmai ir piekļūve pie PayPal servisiem, kas palīdz autorizēt tranzakciju. Lai atveidotu PayPal pogu, tiek izmantots PayPal JavaScript SDK. Uznirstošajā logā tiek palaista PayPal Checkout plūsma. Pēc tam, kad pircējs ir autorizējis darījumu, PayPal poga izsauc JavaScript atzvanīšanas funkciju programmas pusē, kura pievieno lietotāju pie konkrētas izsoles dalībniekiem, ja PayPal Checkout plūsmai izpildīšanas rezultāts bija pozitīvs.

## Funkcionālās prasības

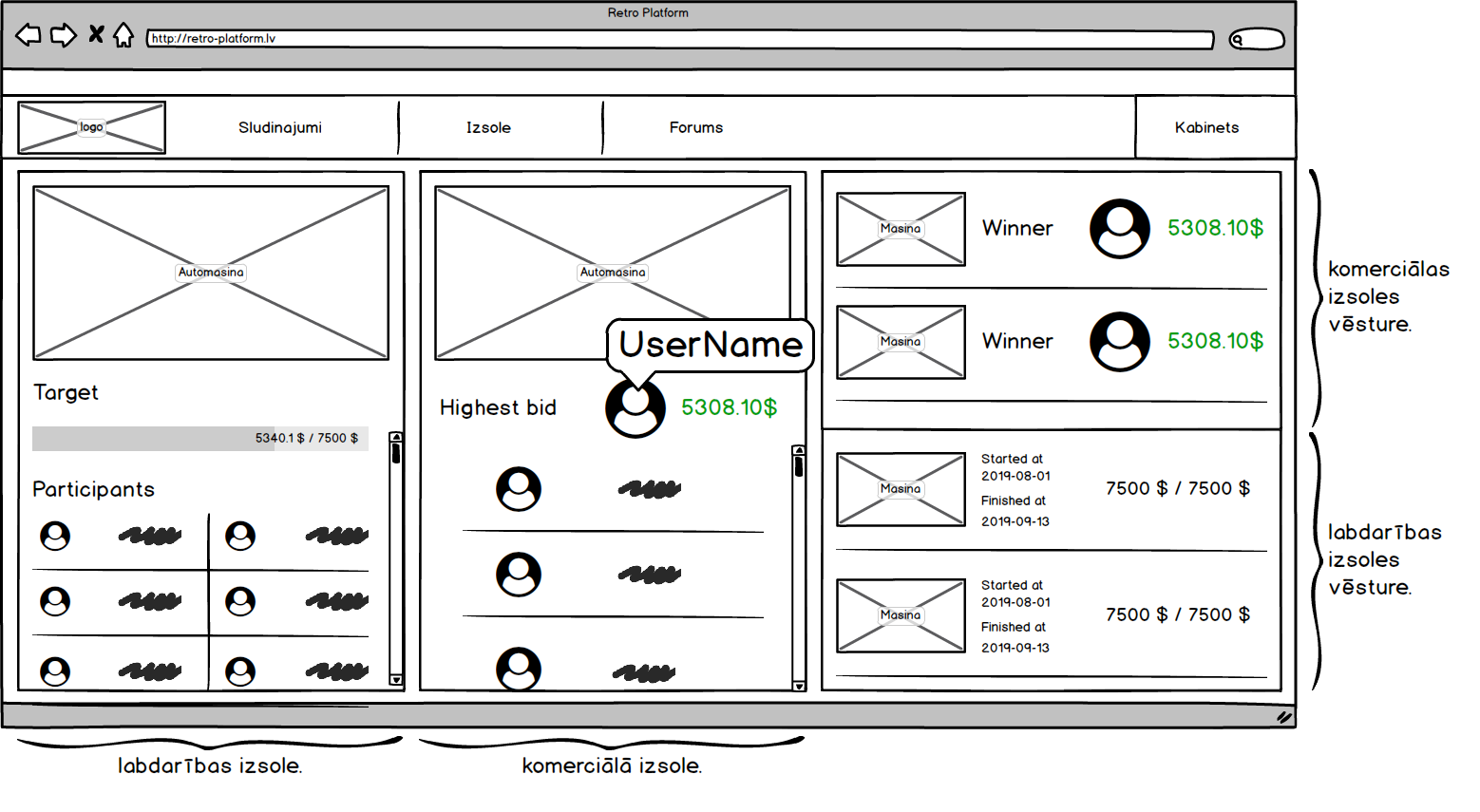
1. Jānodrošina lomu dalīšanu.
   1. Sistēmā eksistē 4 tipu lietotāji:
      1. Virs Administrātors – veic sistēmas pārvaldi, kas sevī ietvēr visu administratoru lomas funcionālu un arī sistēmas administrātoru pārvaldi.
      2. Administrātors – veic sistēmas pārvaldi, kas sevī ietvēr lietotāju, izsoles un forumu datu pārvaldi.
      3. Lietotājs – veic profila pārvaldi, pietiekšanas izsolē un sludinājumu izveidošana.
      4. Viesis – aplūkot visas 3 galvēnas platformas sastāvdaļas – sludinājumi, izsole un forums – bez iespējas piedalīties.
2. Jānodrošina sludinājumu atspoguļošanu.
   1. Lietotājam būs iespēja atspoguļot vairāk sludinājumu, izmantojot pagināciju.
3. Jānodrošina sludinājumu pievienošanu pie grāmatzīmēm.
   1. Lietotājam būs iespēja atzīmēt cita lietotāja sludinājumu, lai atrastu to vēlāk.
   2. Ja lietotājam vairs nav vajadzīga norāde uz sludinājumu vai sludinājums vairs neeksistē, jānodrošina grāmatzīmju noņemšanu no sludinājuma.
4. Jānodrošina sludinājumu filtrāciju.
   1. Meklēšana notiek pēc ievadītas kritērijas, kad lietotājs spied pogu “Search (Mēklēt)”.
   2. Lietotājam ir iespēja iztīrīt visus filtrus, nospiežot pogu “Clear (Notīrīt)”, un veikt jauno pieprasījumu, lai atspoguļotu visus sludinājumus.
5. Jānodrošina sludinājumu pievienošanu.
   1. Lietotājam jāizpilda nepieciešami lauki, lai pievienotu slūdinājumu.
      1. Gadījumā, kad lauki netiek aizpildīti vai aizpildīti nepareizi, sistēma izvada kļūdu un neļauj pievienot slūdinājumu.
6. Jānodrošina sludinājumu dzēšanu.
   1. Lietotājam jānospiž poga, lai nodzēstu viņa pievienotu sludinājumu.
      1. Gadījumā, kad lietotājs apstiprinā savu izvēli, sludinājums pazūd no publiskas pieejas un tiek arhivēts. Notiek mīksta dzēšana (*soft delete*).
7. Jānodrošina izsoles atspoguļošanu.
   1. Lietotājam būs iespēja aplūkot aktīvas izsoles un izsoles vēsturi.
8. Jānodrošina pieteikšanas izsolē.
   1. Lietotājs piedālas izsolē pēc summas ievades un nospiešanas ‘Piedalīties’
      1. Ja naudas vērtība netika ievadīta vai ievadīta nepareizi, sistēma izvada kļūdu un neļauj pieteikties izsolei.
      2. Jānodrošina izsoles informācijas izvadi iespēja, kas dod lietotājam pārlicināties vai piedalīšanas informācija ir pareiza.
      3. Gadījumā, kad transakcijas plūsma neizpildas, sistēma izvada kļūdu un novērš lietotāju pidalīšanu izsolē
   2. Pēc piedalīšanas, lietotājam jābūt redzamam dalībnieku sarakstā.
   3. Izsoles lietotājs ar lielāku naudas apjoms tiks atspoguļots kā vadošs izsoles dalībnieks, aizvietojot eksistējošo līderi, ja tas eksistēja (attiecas uz komerciāliem izsolēm).
9. Jānodrošina pieteikšamas diskusijā foruma ietvaros.
   1. Lietotājam jāizpilda jāizvēlas vēlāmu kategoriju priekš ziņu pievienošanai.
   2. Lietotājam jāizpilda nepieciešami lauki, lai pievienotu ziņu.
      1. Gadījumā, kad lauki netiek aizpildīti vai aizpildīti nepareizi, sistēma izvada kļūdu un neļauj pievienot ziņu.
   3. Lietotājam jāizpilda nepieciešami lauki, lai pievienotu komentārus.
      1. Gadījumā, ja komentārs ir tukšs, sistēma izvada kļūdu un neļauj pievienot komentārus.
10. Jānodrošina diskusijas pārvaldi foruma ietvaros.
    * 1. Lietotājiem būs iespēja nodzēst savas ziņas un komentārus.
      2. Administratoram būs iespēja nodzēst savas un citu lietotāju ziņas un komentārus.
11. Jānodrošina pieteikšamas sistēmā.
    1. Lietotājam jāizpilda pieteikumvārds un paroles lauki.
       1. Lietotājam ir iespēja pieteikties sistēmā, izmantojot vai lietotājvārdu, vai e-pastu kā pieteikumvārdu.
       2. Gadījumā, kad lauki netiek aizpildīti vai lietotājs netika atrasts datu bāze, vai parole nav pareiza, sistēma izvada kļūdu un neļauj pieteikties sistēmai.
       3. Gadījumā, kad lietotāja profila status ir “Suspended(Apturēts)”, sistēma izvada kļūdu un neļauj pieteikties sistēmai.
12. Jānodrošina lietotāja profila pārvaldi.
    1. Lietotājam būs iespēja nomainīt savu lietotājvārdu, vārdu, uzvārdu, e-pastu, dzimumu, paroli un profila bildi.
       1. Gadījumā, kad lauki lietotājvārds, vārds, uzvārds, e-pasts, parole netiek naizpildīti vai e-pasts jau ir aizņemts, un/vai parole neatbilst nosacījumiem, sistēma izvada kļūdu un prasa ievadīt informāciju korekti.
       2. Gadījumā, kad lietotājs augšpielādē jauno bildi, viņam būs iespēja apstiprināt savu izvēli.
       3. Ja izvēle bija apstiprināta, veca lietotāja bilde tiks aizvietota ar jauno.
    2. Lietotājam būs iespēja nodzēst savu profilu.
       1. Nospiedot pogu “Delete (Nodzēst)”, lietotājam būs piedāvāta iespēja apstiprināt vai atcelt savu izvēli.
       2. Gadījumā, kad lietotājs apstiprinā savu izvēli, notiek lietotāju atteikšanās no sistēmas un lietotāja profils tiks izdzēst datubāzē.
13. Jānodrošina izsoles pārvaldi.
    1. Administratoram būs iespēja pievienot jaunas izsoles un manuāli pabeigt vēl aktīvas izsoles.
       1. Gadījumā, kad lauki netiek aizpildīti vai aizpildīti nepareizi, sistēma izvadu kļūdu un prasa ievadīt informāciju korekti.
14. Jānodrošina sludinājumu pārvaldi.
    1. Administratoram būs iespēja arhivēt citu lietotāju sludinājumus.
15. Jānodrošina lietotāju pārvaldi.
    1. Administratoram būs iespēja apturēt visas lietotāju darbības platformas ietvaros, ka arī atjaunot viņa statusu atpakaļ.
16. Jānodrošina administratoru pārvaldi.
    1. Virs Administratoram būs iespēja paaugstināt lietotāju statusu līdz administratoram, ka arī noņemt to no jebkura administratora.

## Nefunkcionālās prasības

1. Jānodrošina SPA (Single Page Application) pieeju platformu izveidošanai. *Single Page Application* – tīmekļu programmas tips, kas izmanto vienīgas parogrammas lapas principu, t.i. nepārlādē mājas lapu pēc pieprasījuma nosūtīšanas, rezultāta saņemšanas, datu izmaiņam, kā neredzamām lietotājam (iekšējie dati, mainīgie, skaitītāji utt.), tā arī vizuālām (dokumentu elementu paradīšana / paslēpšana, to atjaunošana un adaptācija
   1. Visa vietne (DOM) lādējas pēc pirmas lietotāja aplūkošanas.
   2. Pārejot uz citu sadaļu, vietne nepārladējas un atspoguļo interesejošo informāciju, izmantojot komponentus, kas paātrina izvades laiku un servera pieprasījumus.
2. Jānodrošina sludinājumu detalizētu aprakstu.
   1. Spiežot uz sludinājumu, parādas pilns sludinājuma apraksts.
   2. Sludinājuma detaļām jābūt sakārtotam pēc tas kategorijas.
3. Jānodrošina vietnes navigācija, kas palīdz lietotājam pārvietoties platformas ietvaros, kura ir pieejāma jebkurā liakā, kad lietotājs izmanto sistēmu.
4. Jānodrošina sludinājumu ērtāku atspoguļošanu.
   1. Atspoguļošana komponentā, kas ierobiežo sludinājumu saraksta augstumu (*sk. Att. 1.*).
5. Sludinājumu filtrācija ir jāparadās nolaižamā sarakstā kreisajā pusē (*sk. Att. 1.*).
6. Profilu rediģēšana ir jāparadās modālā dialogā ar iespēju aizvērt to, uzspiežot un krustiņu un arī jebkur ārpus loga rebožām.
7. Izsoles dalībnieku sarakstam jābūt redzamām un sakartotām pēc ieguldītās naudas apjoma dilstošā secībā, lai redzētu vislielāku ieguldījumu.
8. Jānodrošina labdarības izsoles visu ieguldījumu atspoguļošanu, kas vizuāli parādā krājumu progresu.
9. Ja lietotājs ir izsoles dalībnikes, viņa rindai jābūt izceltai.
10. Izsoles vēsture ir jāparadās nolaižamā sarakstā.
    1. Izsoles vēsturei jābūt sadalītai pa 2 daļam – labdarības un komerciālas izsoles (*sk. att. 2.5.1.*).
    2. Izsoles informācijā jāiekļauj tas veidošanas un beigšanas datumi (*sk. att. 2.5.2.*).



Att. 2.5.1. Sākums



Att. 2.5.2. Izsole

# **Uzdevuma risināšanas līdzekļu izvēles pamatojums**

## Uzdevuma analīze

Uzdevuma mērķis ir izveidot ērti izmantojāmo vietni, kurā lietotājs var ātri un neaizdomīgi pārvietoties starp platfromas sekcijam. Platforma sastāv no trīm lielām daļām, kurus lietotājam ir iespēja apmēklēt bez informācijas zaudēšanas. Apmēklēšana un atspoguļošana nenotiek vienlaicīgi, bet, pārvietojoties starp nodaļam, stāvoklis saglabājas un lietotājam ir iespēja sēkot līdzi informācijai bez pārladēšanas.

## Rīku analīze

Lai realizētu uzdevumu, ir nepieciešami izmantot:

* + SPA (*Single Page Application*) principu priekš atspoguļošanai;
  + ērtu servera programmatūru (API) priekš pieprasījumu apstrādi;
  + datu glabāšanas struktūru (datubāze).

Parasta HTML / CSS izmantošana neatbilst šī mērķa reailēšanai, jo tas pārladē visu dokumentu (DOM) pēc katra pieprasījuma un tāda veidā nesaglabā lokālu informāciju. Lai to realizētu, ir nepieciešami izmantot ietvaru, kas izmanto virtuālo dokumentu (virtual DOM) un pārlādē tikai nepieciešamus interfeisa daļas, nepārlādējot visu dokumentu, jo tas aizņem daudz resursus. Uz platformas realizāciju ir iespējami izmantot sēkojošus ietvarus:

* + jQuery;
  + AngularJS;
  + VueJS;
  + ReactJS;
  + BackboneJS.

VueJS 2.0 ir galīga iavēle priekš šī uzdevuma realizācijai, jo, salīdzinot ar citiem ietvariem, tādiem kā AngularJS vai BackboneJS, VueJS ir ļoti viegls un modulārs ietvars, kas ir tieši piemērots šim uzdevumam. VueJS 2.0:

* + ir open-source;
  + ļauj veidot un pieslēgt komponentus priekš konkrētiem mērķim, nepārslēdzot programmēšanas vidi;
  + ir komponentu atkartota izmantošanai;
  + ir sabiedrība, kas attīsta ietvaru jebkuram mērķim (no sīkiem moduļiem (*[timers, counters, ...]*, *DOM elementiem*) līdz lielam integrācijam (*papildus* *stila ietvari, Oauth, ‘flux’ informācijas plūsma*)).

Lai nodrošinātu lietotāju autorizāciju, bija nolemts izmantot integrēt risinājumu *@websanova/vue-auth*, kas veido lokālo sesiju un lietotāja datu apmaiņu, izmantojot JSON Web Tokens.

Tā kā vietne ir realizēta ar SPA pieeju, bet maršrutēšana joprojām ir nepieciešama, lai efektīvi un ērti izmantotu vietni, bija nolemts izmantot *Vue Router*, kas imitē parastu maršrutētāja darbību, kur katram maršrutēšanas ceļam ir atsauce un konkrētu vietnes dokumenta komponentu.

Priekš dizainu izveidošanai bija nolemts izmantot *Vuetify*, kas satur daudz komponentu, kurus ir viegli modificēt pēc vajadzībām.

Lai nodrošinātu tūlītēju pieeju pie datiem dažas projekta daļās, bija nolemts izmantot *Vuex*, kas implementē centrālu krātuvi, kuru ir iespējāms piekļūt no jebkura programmas komponenta un nesūtīt datus tur un atkal atpakaļ pa visu komponentu vīrkni.

Lai apstrādātu informāciju no servera puses, ir nepieciešami izmantot PHP balstotu ietvaru vai parastu PHP kodu. Tā kā bija pieņemts lēmums ietvarus, lai paātrinātu procesu un palielināt efektivitāti, šī uzdevuma ietvaros ir iespēja izmantot vienu no sēkojošiem ietvariem:

* + Larvel;
  + Symfony;
  + Cake PHP.

Laravel 6.18.3 ir galīga iavēle priekš šī uzdevuma realizācijai no *back-end* puses, jo tas ir vispopularākais PHP ietavrs, kuram ir atbalsts ne tikai no ražotājiem, bet arī no sabiedrības. Laravel izmanto PHP >= 7.2.0. versiju.

Lai pilnīgi izmantotu JSON Web Token implementāciju no lietotājas saskrnes pusēs, bija nolemts izmantot *tymon/jwt-auth*, lai autorizētu lietotāju no servera puses.

Lai glabātu sistēmas datus bija pieņemts lēmums izmantot MySQL balstotu datubāzi, jo tas ir vispazistamāka un elastīga priekš pieprasījumu veidošanu un datu glabāšanu. Izmantojamā MySQL versija ir 8.0.

## Izmantojamie spraudņi

Interfeiss:

* VueJS: 2.6.11,
* @websanova/vue-auth: 2.21.14-beta,
* Vue-Router: 3.0.3,
* Vuetify: 2.1.0,
* Vuex: 3.0.1

Serveris:

* Laravel: 6.18.3
* tymon/jwt-auth: 1.0.0

# **Programmatūras modelēšana un projektēšana**

## Sistēmas struktūras modelis

### ***Sistēmu arhitektūra***

Funkcionālas dekompozīcijas diagramma sastāv no 4 līmeņiem (*sk. 2. pielikumā*):

1. līmenis – paša platfromas sistēma.
2. līmenis – platformas apakšsistēmas:
   1. Sludinājumu sistēma
   2. Izsoles sistēma
   3. Foruma sistēma
   4. Lietotāju sistēma
3. līmenis – funckionālie moduļi iepriekš aprakstītam sistēmam.
4. līmenis – moduļi, kas paplašinā lietotāju saskarnes iespējas ar sistēmu.

### ***Sistēmas ER modelis***

ER diagramma (*sk. 1 pielikuma*) sastāv no zemāk mīnētājiem entitijam:

* *“****Sludinājums****”* – apraksta lietotāja piedāvājumu. Tās atribūtu kopums sevī ietvēr virsrakstu, saturu, tālruni, statusu un absolūto ceļu uz bildi.
* ***“Detaļa”*** – apraksta detaļu būtību. Tās atribūtu kopums sevī ietvēr nosaukumu.
* ***“Detaļas Kategorija”*** – apraksta iespējāmas detaļas tipu / kategoriju. Tās atribūtu kopums sevī ietvēr nosaukumu.
* ***“Labdarības Izsole”*** – apraksta labdarības izsoles tipu. Tās atribūtu kopums sevī ietvēr krājamo mērķi, statusu, sākuma datumu un beigšanas datumu.
* ***“Komerciālā Izsole”*** – apraksta komerciālas izsoles tipu. Tās atribūtu kopums sevī ietvēr sākumcenu, piedalīšanos beigu datumu, statusu, sākuma datumu un beigšanas datumu.
* *“****Eksemplārs****”* – apraksta izsoles pievienoto eksemplāru. Tās atribūtu kopums sevī ietvēr nosaukumu un absolūto ceļu uz bildi.
* ***“Komentārs”*** – apraksta lietotāju komentārus. Tās atribūtu kopums sevī ietvēr saturu.
* ***“Lietotājs”*** – apraksta platformas lietotājus. Tās atribūtu kopums sevī ietvēr vārdu, uzvārdu, absolūto ceļu uz bildi, e pastu, paroli, dzimumu, lomu, lietotājvārdu un statusu.
* ***“Ziņa”*** – apraksta ziņas tēmu. Tās atribūtu kopums sevī ietvēr virskrakstu un saturu.
* ***“Ziņu Kategorija”*** – apraksta ziņas kategoriju. Tās atribūtu kopums sevī ietvēr nosaukumu.

Datu bāzes relācijas parāda kā savstarpēji saistītas divas vai vairākas entītijas:

* + - starp lietotājiem un labdarības izsolei ir attiecības daudz pret daudziem, jo viens lietotājs var piedalīties vairākas izsolēs, un vienā izsolē var piedalīties vairāki lietotāji; tajā rezultātā tiek apreķināts lietotāja solījums;
    - starp lietotājiem un komerciālai izsolei ir attiecības daudz pret daudziem, jo viens lietotājs var piedalīties vairākas izsolēs, un vienā izsolē var piedalīties vairāki lietotāji; tajā rezultātā tiek apreķināts lietotāja solījums un lietotājs ar vislielāko solījumu;
    - starp lietotājiem un sludinājumu ir attiecības viens pret daudziem, jo vienam lietotājam pieder vairāki sludinājumi, bet vienam sludinājumam ir tikai viens autors (lietotājs); tajā rezultātā tiek apreķināts sludinājuma izveidošanas datums;
    - starp lietotājiem un ziņam ir attiecības viens pret daudziem, jo vienam lietotājam var piederēt vairākas ziņas, bet vienai ziņai ir tikai viens autors (lietotājs); tajā rezultātā tiek apreķināts ziņas izveidošanas datums;
    - starp ziņu ketegorijam un ziņam ir attiecības viens pret daudziem, jo vienai ziņu ketegorijai var piederēt vairākas ziņas, bet vienai ziņai ir tikai viena ziņu ketegorija;
    - starp lietotājiem un komentāriem ir attiecības viens pret daudziem, jo vienam lietotājam var piederēt vairāki komentāri, bet vienam komentāram ir tikai viens autors (lietotājs); tajā rezultātā tiek apreķināts komentāra izveidošanas datums;
    - starp ziņam un komentāriem ir attiecības viens pret daudziem, jo vienai ziņai var piederēt vairāki komentāri, bet vienam komentāram ir tikai viena ziņa;
    - starp detaļam un sludinājumiem ir attiecības viens pret daudziem, jo vienam sludinājumam varētu būt vairākas detaļas, bet viena detaļa pieder vienam sludinājumam;
    - starp detaļam un detaļu kategorijām ir attiecības viens pret daudziem, jo vienai detaļu kategorijai var piederēt vairākas detaļas, bet viena detaļa pieder tikai vienai detaļu kategorijai;
    - starp komerciālām un labdarības izsolēm un eksemplāriem ir attiecības viens pret vienu, jo vienā izsolē tiek izspēlēts viens eksemplārs, un viens eksemplārs tiek piešķirts tikai vienai izsolei.

## Funkcionālais sistēmas modelis

### ***Datu plūsmu modelis***

Lietotāja reģistrācija (*sk. 3. pielikumā*) – šī posma notiek lietotāja reģistrācija ar ievadīto datu validāciju.

Ievadot lietotāja vārdu un uzvārdu, sistēma pārbauda, vai tie sastāv tikai no burtiem un neparsniedz 24 simbolu garumu. Gadījumā, ja vārds un / vai uzvārds ir pārak garšs / sastāv no pārējiem simboliem, sistēma izvada kļūdu. Abi lauki nav obligāti aizpildami.

Ievadot lietotājvārdu, sistēma pārbauda, vai tas vispār ir ievadīts, atrodas 5-16 simbolu skaitu diapazonā un to unikalitāti. Gadījumā, ja lietotājvārds nav ievadīts / pārak garšs / ir ļoti īss / lietotājvārds jau ir aizņemts, sistēma izvada kļūdu.

Ievadot e-pastu, sistēma pārbauda, vai tas vispār ir ievadīts, atbilst ‘\*@\*.\*’ formātam un nebija izmantots sistēmā āgrāk. Gadījumā, ja epasts nav ievadīts / neatbilst formātam / e-pasts jau ir aizņemts, sistēma izvada kļūdu.

Ievadot paroli, sistēma pārbauda, vai tas vispār ir ievadīta, ir sarežģīta un atrodas 8-16 simbolu skaitu diapazonā. Gadījumā, ja parole nav ievadīta / pārak garša / ir ļoti īsa, sistēma izvada kļūdu.

Ievadot paroles apstiprinājumu, sistēma pārbauda, vai tas vispār ir ievadīts un sakrīt ar iepriekš ievadīto paroli. Gadījumā, ja paroles apstiprinājumu nav ievadīts / nesakrīt ar iepriekš ievadīto paroli, sistēma izvada kļūdu.

Lietotājam ir nepieciešami izvēleēt vienu no dzimumiem. Ja dzimums netika izvēlēts, sistēma izvada kļūdu.

Pēc visām validācijam notiek datu ierakstīšana datubāzē, kur lietotājam pievienojas vēl viens lauks *role* ar vērtību *user*, kurš apzīme viņu kā parasto lietotāju un lietotājs tiek pāradresēts uz savu profilu.

Lietotāja pieteikšanās (*sk. 4. pielikumā*) – šī posma notiek lietotāja pieteikšanās ar ievadīto datu validāciju. Šeit lietotājs var izmantot savu lietotājvārdu vai epastu, lai pieteiktos sistēmā.

Ievadot lietotājvārdu vai epastu , sistēma pārbauda, vai tas vispār ir ievadīts un pārbauda to datubāzē. Gadījumā, ja lietotājs nav ievadīts / neeksistē datubāzē, sistēma izvada kļūdu.

Ievadot paroli, sistēma pārbauda, vai tas vispār ir ievadīta un atbilst izvēlētājiem lietotājam. Gadījumā, ja parole nav ievadīta / nav pareiza, sistēma izvada kļūdu.

Pēc visām validācijam notiek lietotāja statusa pārbaude. Gadījumā, ja lietotāja profils ir aptūrēts, sistēma izvada kļūdu. Ja lietotājs ir aktīvs, notiek sesiju izveide, kur ģenerējas *token*, kurš glāba informāciju par sesiju un lietotājs tiek autorizēts un pāradresēts uz savu profilu. Tas ir objekts, kurš glābā sesijas datus, tādus kā unikālais numurs, izveidošanas laiks, derīguma termiņš, un ir izmantots priekš lietotāja autorizācijai. *Token* tiks saglabāts lokālā datubāzē (local storage). Local storage – lokālā datu uzglābšanas bāze, kas atrodas paša parlūkprogrammā.

Lietotāja atteikšanās (*sk. 5. pielikumā*) – šī posma notiek lietotāja atteikšanās no sistēmas.

Pēc pogas nospiešanas, notiek lietotāja atteikšanās. no sistēmas, kurā notiek sesiju iznīcināšana un *token* atzīšana par spēkā neesošu. Pēc tam notiek lietotāju pāradresēts uz sākumlapu.

Lietotāja datu rediģēšana (*sk. 6. pielikumā*) – šī posma notiek lietotāja datu rediģēšana ar ievadīto datu validāciju.

Ievadot lietotāja vārdu un uzvārdu, sistēma pārbauda, vai tie sastāv tikai no burtiem un neparsniedz 24 simbolu garumu. Gadījumā, ja vārds un / vai uzvārds ir pārak garšs / sastāv no pārējiem simboliem, sistēma izvada kļūdu.

Ievadot lietotājvārdu, sistēma pārbauda, vai tas vispār ir ievadīts, atrodas 5-16 simbolu skaitu diapazonā un to unikalitāti. Gadījumā, ja lietotājvārds nav ievadīts / pārak garšs / ir ļoti īss / lietotājvārds jau ir aizņemts, sistēma izvada kļūdu.

Ievadot e-pastu, sistēma pārbauda, vai tas vispār ir ievadīts, atbilst ‘\*@\*.\*’ formātam un nebija izmantots sistēmā āgrāk. Gadījumā, ja epasts nav ievadīts / neatbilst formātam / e-pasts jau ir aizņemts, sistēma izvada kļūdu.

Ievadot paroli, sistēma pārbauda, vai tas vispār ir ievadīta, ir sarežģīta un atrodas 8-16 simbolu skaitu diapazonā. Gadījumā, ja parole nav ievadīta / pārak garša / ir ļoti īsa, sistēma izvada kļūdu.

Ievadot paroles apstiprinājumu, sistēma pārbauda, vai tas vispār ir ievadīts un sakrīt ar iepriekš ievadīto paroli. Gadījumā, ja paroles apstiprinājumu nav ievadīts / nesakrīt ar iepriekš ievadīto paroli, sistēma izvada kļūdu.

Lietotājam ir iespēja nomainīt dzimumu.

Pēc visām validācijam notiek datu rediģēšana datubāzē, kur lietotāja dati aizvietojas ar jauniem ievadītiem datiem.

Lietotāja bildes maiņa (*sk. 7. pielikumā*) – šī posma notiek lietotāja bildes.

Lietotājam ir iespēja nomainīt profila bildi, uzspiežot uz savu tekošo attēlu. Pēc bildes notiek bildes datu rediģēšana datubāzē, kur lietotājam aizvietojas bildes ceļš.

Lietotāja profila apturēšana (*sk. 8. pielikumā*) – šī posma notiek lietotāja profila aptūrēšana, kas ir pieejāma tikai lietotājiem ar lomu Administrators un Virs Administrators.

No sākuma sistēma sagatavo sarakstu ar visiem lietotājiem, kuriem ir piešķirta loma *user* un sadalās tos 2 sarakstos – aktīvie un aptūrēti lietotāji.

Administratoram ir iespēja nospiest pogu *Suspend*, lai apturētu lietotāju darbību vai atjaunotu, nospiežot *Activate*.

Pēc administratora ievades, izvēlēta lietotāja status mainās vai uz *active*, vai uz *suspended* atkārībā no izvēles.

Administratoru pievienošana (*sk. 9. pielikumā*) – šī posma notiek lietotāja profila paaugstināšana līdz administratora statusam vai tas pazemināšana līdz lietotājam, kas ir pieejāma tikai lietotājiem ar lomu Virs Administrators.

No sākuma sistēma sagatavo sarakstu ar visiem aktīviem lietotājiem, kuriem ir piešķirta loma *user* un *admin* un sadalās tos 2 sarakstos – lietotāji un administratori.

Virs Administratoram ir iespēja nospiest pogu *Promote*, lai paaugstinātu lietotāju līdz administratoram vai, prētēji, pazemināt līdz lietotāja statusam, nospiežot *Remove*.

Pēc administratora ievades, izvēlēta lietotāja loma mainās vai uz *admin*, vai uz *user* atkārībā no izvēles.

Sesiju izveide (*sk. 10. pielikumā*) – šī posma notiek automātiska pārbaude no sistēmas puses, kas pārbauda, vai lietotājs ir autorizēts.

Ieladējot mājas lapu, sistēma pārbauda, vai *token* eksistē lokālā datubāzē (uz ierīces) un pārbauda to derīguma tērmiņū. Gadījumā, ja *token* neeksitē / beidzās tas derīguma termiņš, sistēma pāradresē uz sākumlapu. Taču, ja *token* eksistē un tas derīguma termiņs nav beidzies, sistēma autorizē lietotāju un atgriež uz pēdējo apmēklētu lapu, atjaunojot *token*.

Lietotāja lomas pārbaude (*sk. 11. pielikumā*) – šī posma notiek automātiska lietotāja lomas pārbaude, kas kontrolē vietnes sadaļas atspoguļošanu atkarība no lietotāja tiesībam.

Pietiektos profilā, sistēma pārbauda, kāda loma ir piešķirta lietotājam. Gadījumā, ja loma ir regulārais lietotājs, vietnē parādās lietotāja paneļa sadaļa un ar to iespēja nodarboties ar sludinājumu pārvalid un izsoļu pieteikšanu. Ja lietotāja loma ir administrators, viņam ir pieejami visi regulārā lietotāj iespējas un arī vietnē parādās administratoru panelis ar iespēju nodarboties ar lietotāju, sludinājumu un izsoļu pārvaldi. Ja lietotāja loma ir virs administrators, tad viņam ir pieejāmas visi administratoru pārvaldes rīki un administratoru pārvalde.

Sludinājumu mēklēšana (*sk. 12. pielikumā*) – šī posma notiek sludinājumu mēklēšana pēc ievadītiem filtrācijas datiem..

Pēc interesējošas filtru aizpildīšanas, sistēma pārbauda vai kādi no sludinājumiem datubāzē atbilst pieprasītam filtriem un izvada tos, ja tādi ir.

Sludinājumu pievienošana (*sk. 13. pielikumā*) – šī posma notiek sludinājuma pievienošana ar ievadīto datu validāciju, kas ir pieejama tikai reģistrētājiem lietotājiem.

Ievadot virsrakstu, sistēma pārbauda, vai tas vispār ir ievadīts un nav pārak garšs. Gadījumā, ja lietotājs nav ievadīts / ir pārak garšs, sistēma izvada kļūdu.

Ievadot aprakstu, sistēma pārbauda, vai tas vispār ir ievadīts un nav pārak garšs. Gadījumā, ja lietotājs nav ievadīts / ir pārak garšs, sistēma izvada kļūdu.

Tālāk lietotājam ir iespēja ierakstīt nosaukumu detaļai un izvēlēt tas kategoriju. Pēc noklusījuma sludinājumam tiek pievienota 1 detaļa.

Ievadot detaļas nosaukumu, sistēma pārbauda, vai tas vispār ir ievadīts un nav pārak garšs. Gadījumā, ja lietotājs nav ievadīts / ir pārak garšs, sistēma izvada kļūdu.

Izvēlējot detaļas kategoriju, sistēma pārbauda, vai tā tika pievienota katrai detaļai. Gadījumā, ja pielukmu vismaz vienai detaļai trūkst kategoija, sistēma izvada kļūdu.

Ievadot tālruņa numuru, sistēma pārbauda, vai tas vispār ir ievadīts , nav pārak garšs un vaibija ievadīti tikai cipai.. Gadījumā, ja tālrunis neatbilst validācijas notikumiem, sistēma izvada kļūdu.

Izvēlējot pielikumus (failus, bildes), sistēma pārbauda, vai ir pievienots vizmaz 1 pielikums. Gadījumā, ja pielikumu nav, sistēma izvada kļūdu.

Pēc visām validācijam notiek katras bildes nosaukuma ģenerēšana, lai to ērti un bez konfliktiem saglabāt uz servera. Tālāk notiek sludinājuma datu ierakstīšana datubāzē. Beidzot, lietotājs tiek paziņots par pievienotu sludinājumu.

Sludinājumu pievienošana pie grāmatzīmēm (*sk. 14. pielikumā*) – šī posma notiek sludinājumu pievienošana pie grāmatzīmēm, kas ir pieejāma tikai reģistrētājiem lietotājiem.

Sistēma pārbauda, vai sludinājuma autors ir cits lietotājs, lai autors nevarētu pievienot savus izveidotus sludinājumus pie grāmatzīmēm.

Lietotājam ir iespēja nospiest pogu ‘Sirds’, kas atrodas blakus ar sludinājuma virsrakstu. Ja poga ir ieskicēta, pēc uzspiešanas sludinājums pievienojas pie lietotāja grāmatzīmēm; ja ir aizpildīta – sludinājums tiek noņemts no grāmatzīmēm.

Sludinājumu arhivēšana (*sk. 15. pielikumā*) – šī posma notiek sludinājumu arhivēšana pēc kuras tas nebūs pieejāma publiskai pieejai, kas ir pieejāma tikai reģistrētājiem lietotājiem.

Sistēma pārbauda, vai sludinājuma autors sakrīt ar pieslēgto sistēmai lietotājāju un dod iespēju arhivēt sludinājumu. Gadījumā, ja lietotājam ir loma Administrators vai Virs Anministrators, viņam ir tiesības arhivēt jebkuru sludinājumu neatkarīgi no autora.

Nospiežot pogu *Delete*/*Archive*, kas atrodas blakus ar sludinājuma virsrakstu sludinājuma status mainās no *active* uz *archived* un ir pieejams tikai administratora paneļā.

Tēmu (topic) pievienošana forumā (*sk. 16. pielikumā*) – šī posma notiek ziņas pievienošana forumā ar ievadīto datu validāciju, kas ir pieejama tikai reģistrētājiem lietotājiem.

No sākuma lietotājam jāizvelas kategorija no saraksta. Tālāk notiek ievadītu datu pārbaude un validācija.

Ievadot virsrakstu, sistēma pārbauda, vai tas vispār ir ievadīts un nav pārak garšs. Gadījumā, ja virsraksts neatbilst validācijas notikumiem, sistēma izvada kļūdu.

Ievadot aprakstu, sistēma pārbauda, vai tas vispār ir ievadīts un nav pārak garšs. Gadījumā, ja apraksts neatbilst validācijas noteikumiem, sistēma izvada kļūdu.

Pēc validācijam notiek tēmas datu ierakstīšana datubāzē. Beidzot, lietotājs tiek paziņots par pievienotu tēmu.

Komentāru pievienošana ziņai (topic) forumā (*sk. 17. pielikumā*) – šī posma notiek komentāru pievienošana ziņai ar ievadīto datu validāciju, kas ir pieejama tikai reģistrētājiem lietotājiem.

No sākuma lietotājam jāizvelas kategorija no saraksta un interesejoša ziņa. Tālāk notiek ievadītu datu pārbaude un validācija.

Ievadot aprakstu, sistēma pārbauda, vai tas vispār ir ievadīts un nav pārak garšs. Gadījumā, ja apraksts neatbilst validācijas noteikumiem, sistēma izvada kļūdu.

Pēc validācijam notiek komentāra datu ierakstīšana datubāzē un tas atspoguļošana diskusijā.

Labdarības izsoles pievienošana (*sk. 18. pielikumā*) – šī posma notiek labdarības izsoles pievienošana ar ievadīto datu validāciju, kas ir pieejama tikai reģistrētājiem lietotājiem ar lomu Administrators vai Virs Aministrators.

No sākuma sistēma pārbauda, vai dotajā momentā vēl nav aktīvas labdarības izsoles. Ja tādas nav, lietotājam ir iespēja pievienot jauno. Gadījumā, ja eksistē aktīva labdarības izsole, sistēma neļaut pievienot izsoli.

Ievadot eksemplāra nosaukumu, sistēma pārbauda, vai tas vispār ir ievadīts un nav pārāk garšs. Gadījumā, ja eksemplāra nosaukums neatbilst validācijas noteikumiem, sistēma izvada kļūdu.

Ievadot izsoles mērķi, sistēma pārbauda, vai tas vispār ir ievadīts, ir izmantoti tikai skaitļi un mērķis ir lielāks par nulli.. Gadījumā, ja izsoles mērķis neatbilst validācijas noteikumiem, sistēma izvada kļūdu.

Izvēlējot eksemplāra bildi, sistēma pārbauda, vai tas ir pievienots. Gadījumā, ja bildes nav, sistēma izvada kļūdu.

Pēc validācijam notiek izsoles datu ierakstīšana datubāzē un paziņojums no sistēmas par pievienotu izsoli.

Komerciālas izsoles pievienošana (*sk. 19. pielikumā*) – šī posma notiek komerciālas izsoles pievienošana ar ievadīto datu validāciju, kas ir pieejama tikai reģistrētājiem lietotājiem ar lomu Administrators vai Virs Aministrators.

No sākuma sistēma pārbauda, vai dotajā momentā vēl nav aktīvas komerciālas izsoles. Ja tādas nav, lietotājam ir iespēja pievienot jauno. Gadījumā, ja eksistē aktīva komerciālas izsole, sistēma neļaut pievienot izsoli.

Ievadot eksemplāra nosaukumu, sistēma pārbauda, vai tas vispār ir ievadīts un nav pārak garšs. Gadījumā, ja eksemplāra nosaukums neatbilst validācijas noteikumiem, sistēma izvada kļūdu.

Izvēlējot eksemplāra bildi, sistēma pārbauda, vai tas ir pievienots. Gadījumā, ja bildes nav, sistēma izvada kļūdu.

Ievadot izsoles sākumcenu, sistēma pārbauda, vai tas vispār ir ievadīts, ir izmantoti tikai skaitļi un mērķis ir lielāks par nulli. Gadījumā, ja izsoles sākumcena neatbilst validācijas noteikumiem, sistēma izvada kļūdu.

Izvēlējot izsoles beigu piedalīšanas datumu, sistēma pārbauda, vai tas vispār ir ievadīts. Gadījumā, ja izsoles beigu piedalīšanas datums nav norādīts, sistēma neļauj pievienot izsoli.

Pēc validācijam notiek izsoles datu ierakstīšana datubāzē un paziņojums no sistēmas par pievienotu izsoli.

Piedalīšanas labdarības izsolē (*sk. 20. pielikumā*) – šī posma notiek lietotāja piedalīšanas izsolē ar ievadīto datu validāciju, kas ir pieejāma tikai reģistrētajiem lietotājiem.

Ievadot solījumu, sistēma pārbauda, vai tas vispār ir ievadīts, ir izmantoti tikai skaitļi un solījums ir lielāks par nulli. Gadījumā, ja solījums neatbilst validācijas noteikumiem, sistēma izvada kļūdu un neļauj pievienot solījumu.

Pēc validācijas notiek tranzakcijas autorizācija caur PayPal servisu. Ja tranzakcija tika apstiprināta, lietotājs tiks pievienots pie konkrētas izsoles dalībniekiem. Gadījumā, ja tranzakcijas procesa notiek kļūda, sistēma izvada paziņojumu.

Piedalīšanas komerciālā izsolē (*sk. 21. pielikumā*) – šī posma notiek lietotāja piedalīšanas izsolē ar ievadīto datu validāciju, kas ir pieejāma tikai reģistrētajiem lietotājiem.

Ievadot solījumu, sistēma pārbauda, vai tas vispār ir ievadīts, ir izmantoti tikai skaitļi un solījums ir lielāks par sākumacenu vai par lielāko solījumu, ja tas jau eksistē. Gadījumā, ja solījums neatbilst validācijas noteikumiem, sistēma izvada kļūdu un neļauj pievienot solījumu.

Pēc validācijas notiek solījuma pievienošana pie izsoles, aizvietojot iepriekšējo lielāko solījumu, ja tas eksistē.

Labdarības izsoles beigšana (*sk. 22. pielikumā*) – šī posma notiek labdarības izsoles beigšana, kas ir pieejama tikai reģistrētājiem lietotājiem ar lomu Administrators vai Virs Aministrators.

No sākuma sistēma pārbauda, vai dotajā momentā aktīvs labdarības izsoles mērķis ir sasniegts. Ja mērķis ir sasniegts, lietotājam ir iespēja pabeigt izsoli.

Pēc lietotāja manuālas izsoles pabeigšanas notiek izsoles datu rediģēšana datubāzē un izsoles status mainās no *active* uz *dismissed*.

Komerciālas izsoles beigšana (*sk. 23. pielikumā*) – šī posma notiek komerciālas izsoles beigšana, kas ir pieejama tikai reģistrētājiem lietotājiem ar lomu Administrators vai Virs Aministrators.

No sākuma sistēma pārbauda, vai dotajā momentā tekošais datums tika pārsniegts aktīvas komerciālas izsoles piedalīšanas datumu. Ja datums ir sasniegts, lietotājam ir iespēja pabeigt izsoli.

Pēc lietotāja manuālas izsoles pabeigšanas notiek izsoles datu rediģēšana datubāzē un izsoles status mainās no *active* uz *dismissed*.

# **Datu struktūru apraksts**

## Datubāzes aprakstītas tabulas

Tālāk tika aprakstītas visu tabulu un starptabulu struktūras, kas eksistē platformas datubāze, ar paramētriem, iekļaujot nosaukumu, tipu, garumu un aprakstu ar piezīmēm (*sk. 24. pielikumā*).

Tabulā “auctions” (sk. 5.1.1. tabulā) glabājas vispārīgie dati par izsoli.

* + 1. tabula

Tabulas *“***auctions***”* struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Primārā atslēga |
| 2. | object\_id | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘auction\_object’ |
| 3. | started\_at | timestamp | ––– | CURRENT\_TIMESTAMP |
| 4. | finished\_at | timestamp | ––– | ––– |
| 5. | status | enum | ––– | [active, dismissed]  Default: active |
| 6. | type | enum | ––– | [charity, commercial] |

Tabulā “auction\_object” (sk. 5.1.2. tabulā) glabājas dati par izsoles eksemplāriem.

* + 1. tabula

Tabulas “**auction\_object**” struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Primārā atslēga |
| 2. | object\_type\_id | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘object’\_type’ |
| 3. | name | varchar | 24 | ––– |
| 4. | preview\_image | text | ––– | ––– |

Tabulā “auction\_participants” (sk. 5.1.3. tabulā) glabājas dati par izsoles dalībniekiem.

* + 1. tabula

Tabulas *“***auction\_** **participants***”* struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Primārā atslēga |
| 2. | user\_id | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘users’ |
| 3. | auction\_id | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘auctions’ |
| 4. | amount | double | 8, 2 | ––– |

Tabulā “charity\_auction” (sk. 5.1.4. tabulā) glabājas dati par labdarības izsolēm.

* + 1. tabula

Tabulas *“***charity\_auction***”* struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Primārā atslēga |

5.1.4 . tabulas turpinājums

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 2. | auction\_id | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘auctions’ |
| 3. | goal | double | 8, 2 | ––– |

Tabulā “commercial\_auction” (sk. 5.1.5. tabulā) glabājas dati par komerciālam izsolēm.

* + 1. tabula

Tabulas *“***commercial\_auction***”* struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Primārā atslēga |
| 2. | auction\_id | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘auctions’ |
| 3. | start\_bid | double | 8, 2 | ––– |
| 4. | highest\_bid\_user\_id | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘users’ |
| 5. | end\_date | datetime | ––– | ––– |

Tabulā “favorite\_offers” (sk. 5.1.6. tabulā) glabājas dati par lietotāja atzīmētājiem sludinājumiem.

* + 1. tabula

Tabulas *“***favorite\_offers***”* struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Primārā atslēga |
| 2. | user\_id | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘users’ |
| 3. | offer\_bid | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘offers’ |

Tabulā “object\_type” (sk. 5.1.7. tabulā) glabājas dati par izsoles eksemplāra tipu.

* + 1. Tabula

Tabulas *“***favorite\_offers***”* struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Primārā atslēga |
| 2. | label | varchar | 32 | ––– |

Tabulā “offers” (sk. 5.1.8. tabulā) glabājas dati par lietotāju izveidotājiem sludinājumiem.

* + 1. tabula

Tabulas *“***offers**”struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Primārā atslēga |
| 2. | author\_id | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘users’ |
| 3. | title | varchar | 64 | ––– |
| 4. | body | text | ––– | ––– |
| 5. | contact\_phone | varchar | 12 | ––– |
| 6. | created\_at | timestamp | ––– | ––– |
| 7. | updated\_at | timestamp | ––– | ––– |
| 8. | status | enum | ––– | [active, archived] Default: active |

5.1.8 . tabulas turpinājums

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 9. | preview\_image | text | ––– | ––– |

Tabulā “offers\_media” (sk. 5.1.9. tabulā) glabājas dati par pievienotiem failiem (bildēm) katram sludinājumam.

* + 1. tabula

Tabulas *“***offers\_media***”* struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Primārā atslēga |
| 2. | offer\_id | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘offers’ |
| 3. | photo\_path | text | ––– | ––– |
| 4. | file\_name | text | ––– | ––– |

Tabulā “offers\_parts” (sk. 5.1.10. tabulā) glabājas dati par pievienotiem detaļam katram sludinājumam.

* + 1. tabula

Tabulas *“***offers\_parts***”* struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Primārā atslēga |
| 2. | part | varchar | 32 | ––– |
| 3. | category | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘part\_category’ |
| 4. | offer\_id | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘offers’ |

Tabulā “part\_category” (sk. 5.1.11. tabulā) glabājas dati par detaļas kategoriju, kura tiks piešķļirta katrai detaļai.

* + 1. tabula

Tabulas *“***part\_category**” struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Primārā atslēga |
| 2. | label | varchar | 32 | ––– |

Tabulā “topics” (sk. 5.1.12. tabulā) glabājas dati par foruma ziņam.

* + 1. tabula

Tabulas “**topics**” struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Primārā atslēga |
| 2. | author\_id | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘users’ |
| 3. | category\_id | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘topics\_category’ |
| 4. | title | varchar | 64 | ––– |

5.1.12 . tabulas turpinājums

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 5. | body | text | ––– | ––– |
| 6. | created\_at | timestamp | ––– | ––– |
| 7. | updated\_at | timestamp | ––– | ––– |

Tabulā “topics\_category” (sk. 5.1.13. tabulā) glabājas dati par ziņu kategoriju, kura tiks piešķļirta katrai ziņai.

* + 1. tabula

Tabulas “**topics\_category**” struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Primārā atslēga |
| 2. | label | varchar | 32 | ––– |

Tabulā “topics\_comments” (sk. 5.1.14. tabulā) glabājas dati par uzrakstītājiem komentāriem kādai ziņai (postam).

* + 1. tabula

Tabulas *“***topics\_comments***”* struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Primārā atslēga |
| 2. | author\_id | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘users’ |
| 3. | topic\_id | bigint | 20 | Saite uz lauku ‘id’ tabulā ‘topics’ |
| 4. | body | text | ––– | ––– |
| 6. | created\_at | timestamp | ––– | ––– |
| 7. | updated\_at | timestamp | ––– | ––– |

Tabulā “users” (sk. 5.1.15. tabulā) glabājas dati par katru lietotāju. Lietotājam tiek piešķirta loma, kas nosaka lietotāja tiesības platformas ietvaros.

* + 1. tabula

Tabulas “u**sers**” struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | bigint | 20 | Primārā atslēga |
| 2. | email | varchar | 255 | Unikāla vērtība |
| 3. | username | varchar | 16 | Unikāla vērtība |
| 4. | avatar\_path | text | ––– | ––– |
| 5. | name | varchar | 24 | ––– |
| 6. | surname | varchar | 24 | ––– |
| 7. | gender | enum | ––– | [male, female] |
| 8. | password | varchar | 255 | ––– |
| 9. | created\_at | timestamp | ––– | ––– |
| 10. | updated\_at | timestamp | ––– | ––– |
| 11. | role | enum | ––– | [super\_admin, admin, user]  Default: user |

5.1.15 . tabulas turpinājums

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 12. | status | enum | ––– | [active, suspended]  Default: active |

Tabulā “migrations” (sk. 5.1.16. tabulā) glabājas dati par datubāzes izveidošanas pieprasījumiem, kas ir izveidoti ar Laravel ietvara palīdzību un kas ļauj sēkot līdzi datubāzes struktūras izmaiņām.

* + 1. tabula

Tabulas “**migrations**” struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | id | int | 10 | Primārā atslēga |
| 2. | migration | varchar | 255 | ––– |
| 3. | batch | int | 11 | ––– |

# **Lietotāja ceļvedis**

## Sistēmas prasības aparatūrai un programmatūrai

Lai izmantotu šo projekta programmatūru, zemāk (sk. tabula 8.1.1.) tika aprakstīti minimālas prasības priekš tās lietošanu.

8.1.1 tabula

Tabula **“Minimālas sistēmu prasības”**

|  |  |
| --- | --- |
| **Sistēmas daļa** | **Minimālā prasība** |
| Operētājsistēmā | Windows 7/ OS X Yosemite 10.10 / 64-bit Ubuntu 16.04 |
| Procesors | Intel Pentium 4 / AMD Athlon 64 |
| Operatīvā atmiņa | 512 MB RAM |
| Brīvas vietas uz diska | 128 MB |

Programma izmanto pārlūkprogrammu kā galvēnu palaišanas vīdi. Zemāk tika aprakstīti atbalstītas pārlūkprogrammas:

* Google Chrome Blink 78
* Firefox 38+
* Opera 64.0.3417.73
* Microsoft Edge 17

## Sistēmas instalācija un palaišana

Lai palaistu projekta programmatūru, ierīcei ir nepieciešami atbilst minimāliem sistēmas prasībām un izmantot vienu no pārlūkprogrammām. Pēc tam ir nepieciešami izpildīt sekojošus soļus.

1. Lejupielādējam Git[4].
2. Instalējam git.
3. Lejupielādējam WAMP[10], lai varētu izveidot webserveri.
4. Instalējam WAMP.
5. Pārliecinamies par WAMP darbību atverot adresi http://127.0.0.1
6. Jāizveido tukšo datubāzi ar nosaukumu **retro\_platform**
7. Dodamies uz WAMP atrašanās vietu (piemēram, c:\wamp{64}\www).
8. Veicam labo klikšķi un izvēlamies opciju "git bash here" un izpildam zemāk raksīto komandu:

* git clone https://github.com/rvtprog-kvalifikacija-20/d42-PavelsZuravlovs-RetroPlatforma.git

1. Lejupielādējam un instalējam Composer[1].
2. Lejupielādējam un instalējam NodeJS kopā ar NPM (LTS)[6]
3. Dodamies uz projekta atrašanās vietu (piemēram, c:\wamp{64}\www\d42-PavelsZuravlovs-RetroPlatforma).
4. Izveidojam **.env** failu
5. Dodamies uz PasteBin[7], kopējam visas rindas un ievietojam kodu **.env** failā
6. Taisām vaļā 2 termināļus (CMD / bash / PowerShell)
7. Izpildam sekojošas komandas pēc kārtas 1. terminālā (ja būs prompti ar [yes/no], atbildam 'yes'):
   1. composer i
   2. php artisan migrate
   3. php artisan jwt:secret
   4. php artisan storage:link
   5. php artisan serve
8. Izpildam sekojošas komandas pēc kārtas 2. terminālā (ja būs prompti ar [yes/no], atbildam 'yes'):
   1. cd ./frontend/
   2. npm i
   3. npm run serve
9. Neaizverot terminālus, atveram adresi http://localhost:8080/

### ***Datubāze***

Lai ērtāk strādāt ar portālu un gribat aizpildīto datubāzi, iztukšojiet **retro\_platform** datubāzi (neizdzēsiet pašu DB) un importējiet šo failu:

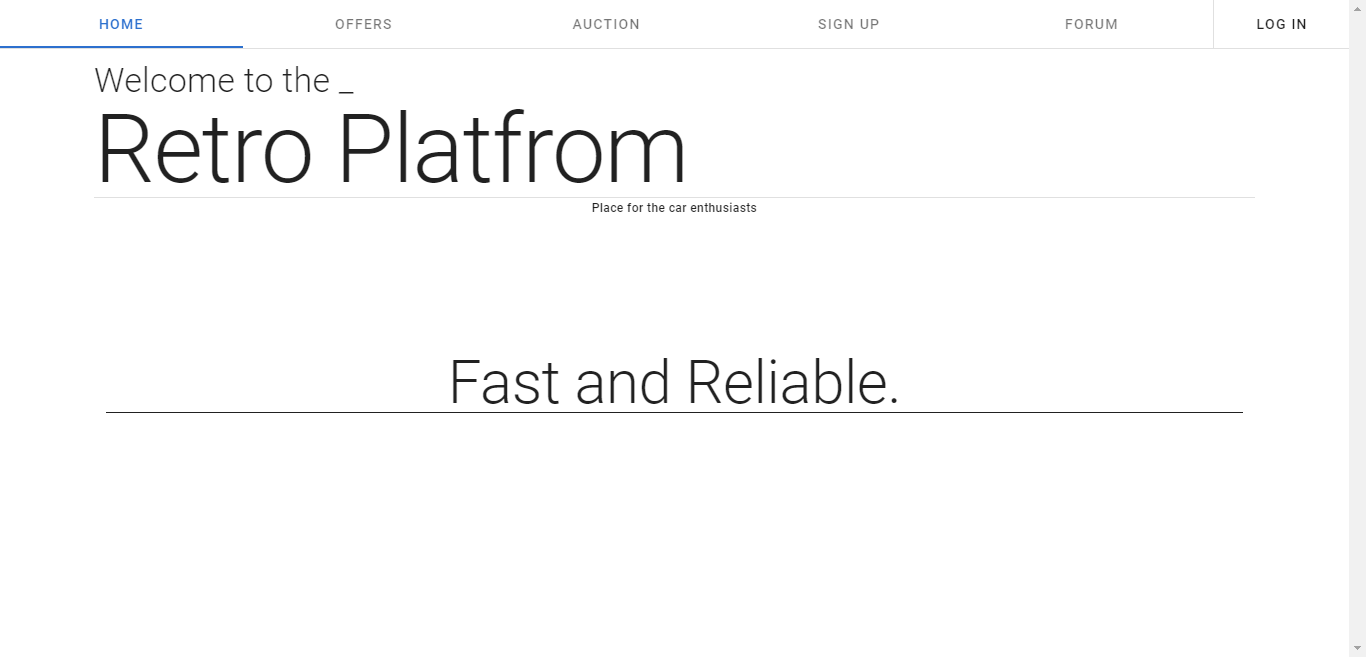
* Database dump (tables + contents): https://pastebin.com/raw/0a5dKt4q Lai piesliegties sistēmā, drīkst izmantot vai lietotājvārdu, vai e-pastu; visiem izveidotājiem lietotājiem parole ir 12345678.

### ***PayPal***

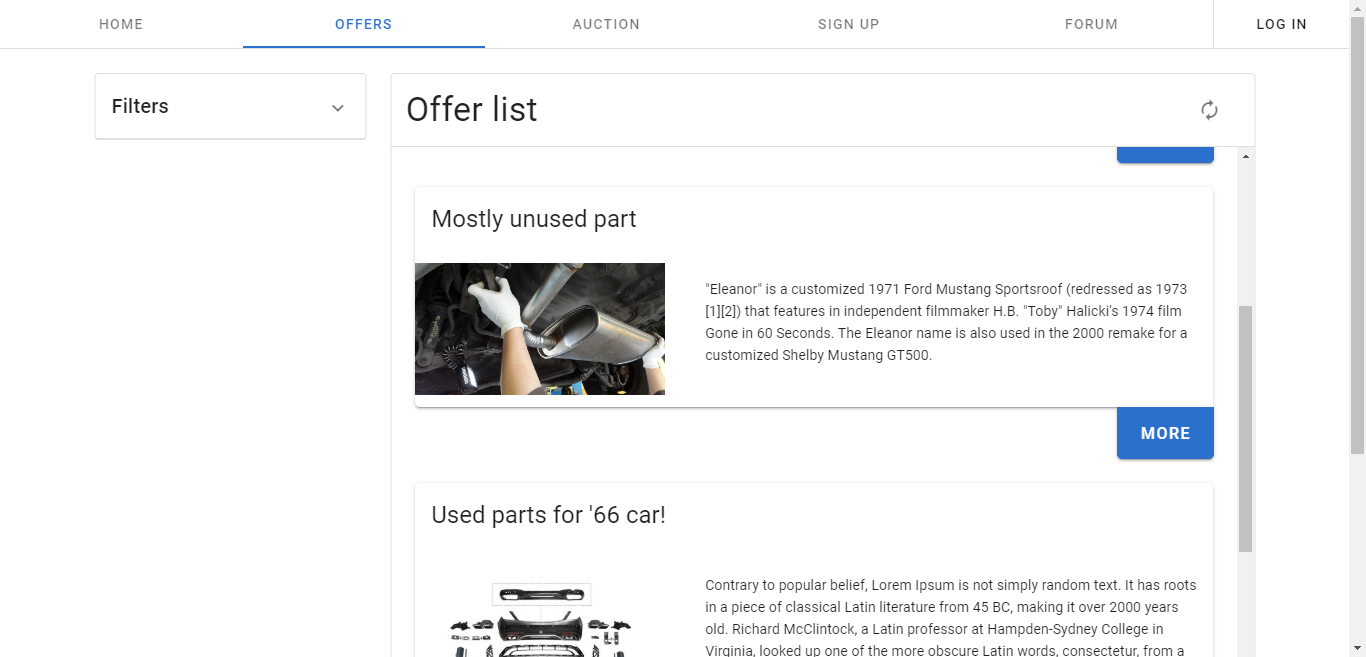
Sistēma izmanto **sandbox** režīmu, tas nozīme, ka sistēmā tagad atrodas tikai **development** konti un ir nepieciešami izmantot zemāk piedāvāto kontu.

* PayPal login: projektsTest@mail.com.
* PayPal password: 12345678.
* Naudas apjoms: 100 000.

## Programmas apraksts

Pēc vietnes palaīsanas, lietotājs redzēs škumlapu ar projekta nosaukumu un navigāciju lapas augšdaļā (*sk. att. 6.3.1 Galvēnais skats)*, kur būs izvēlēta *Home* sadaļa.

Att. 6.3.1. Galvēnais skats

Izmantojot navigāciju un uzspiežot un sadaļu *Offers*, lietotājam būs redzams skats ar sludinājumiem un tas filtriem (*sk. att. 6.3.2. Sludinājumu skats)*.

Att. 6.3.2. Sludinājumu skats

Šeit lietotājam tiek piedavāta iespēja apskatīt visus sludinājumus, izmantjot pgināciju lapas lejā (*sk. att. 6.3.3.)*

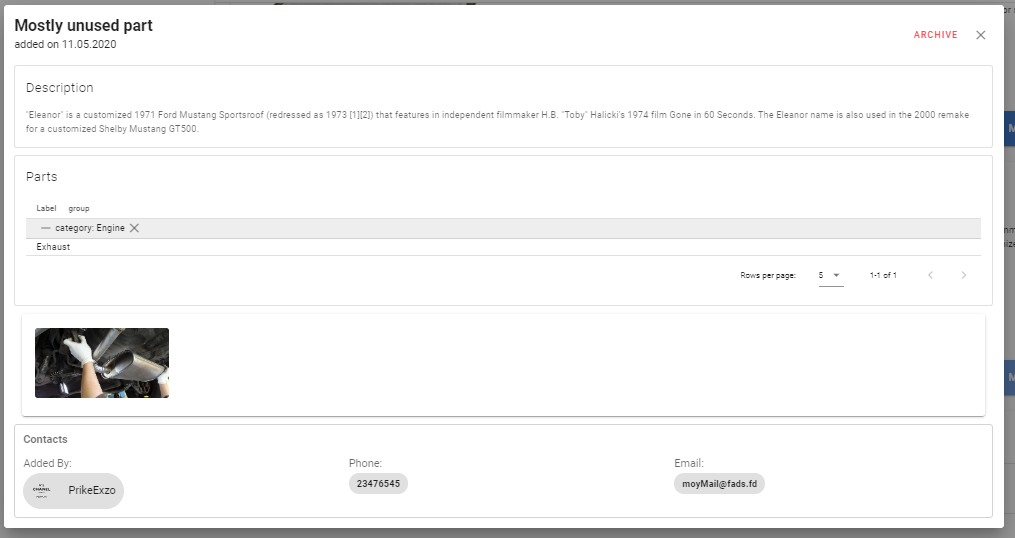
**un katru no sludinājumiem atsevišķi, nospiežot pogu *More* (*sk. att. 6.3.4.)*,

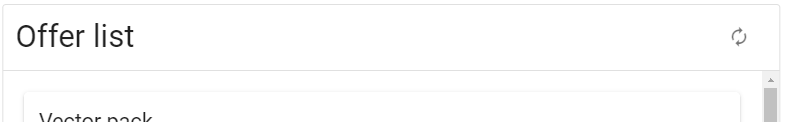
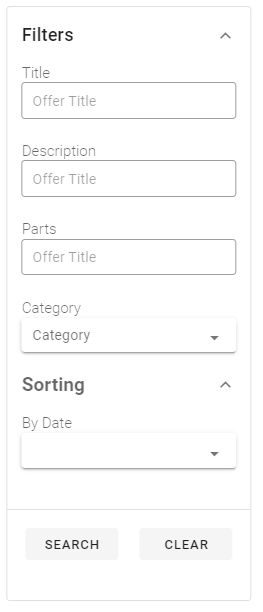
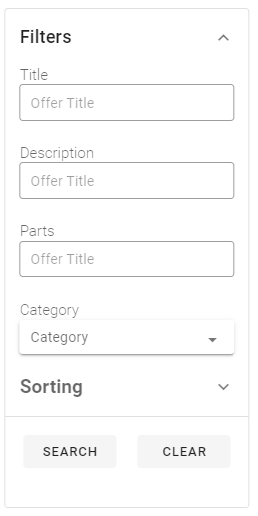
Att. 6.3.4. Poga “More”

Att. 6.3.3. Sludinājumu paginācija

pēc kuras lietotājs var redzēt visu informāciju par sludinājumu (*sk. att. 6.3.5.)*,

Att. 6.3.5. Sludinājumu informācija

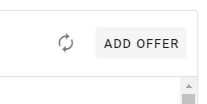
iekļaujot sludinājuma aprakstu ar detaļas informāciju un autora kontaktinformāciju. Šeit lietotājam ir iespēje izmantot filtrāciju, nospiežot uz izvēlni *Filters* (*sk. att. 6.3.6.)*, kur ir iespējāmi filtrēt sludinājumu sarakstu pēc tas virsraksta, apraksta, detaļas nosaukuma, detaļas kategorijas un sludinājuma izveidošanas datuma, uzspiežot uz izvēlni *Sorting* (*sk. att. 6.3.7.)*. Lai uzsāktu sludinājumu mēklēšanu pēc noradītajām filtriem, lietotājam jānospiež pogu *Search* un, lai izdzēstu visus filtrus – pogu *Clear* (*sk. att. 6.3.6.*).

*C:\Users\zdkan\Documents\ShareX\Screenshots\2020-05\chrome_B5mUr3wsTp.png*Lietotājma ir iespēja manuāli atjaunot sludinājumu sarakstu, nospiežot pogu (*sk. att. 6.3.8.*).

Att. 6.3.8. Sludinājumu saraksta atjaunošana

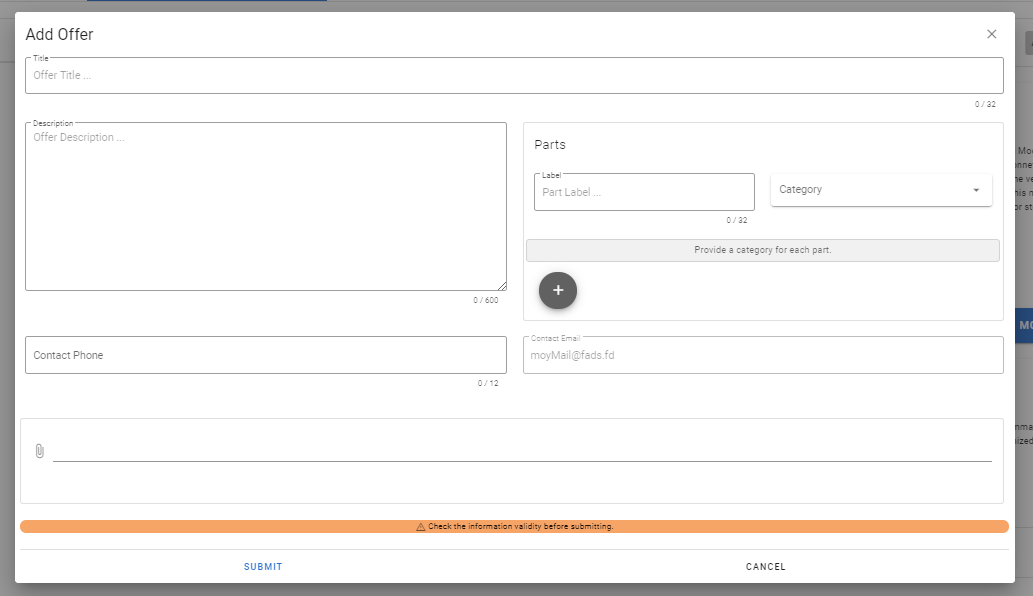
Att. 6.3.6. Sludinājumu filtrācija

Att. 6.3.7 Sludinājumu kārtošana

Ja lietotājs ir piereģisrēts, viņam būs iespēja pievienot sludinājumu, izmantojot pogu *Add Offer* (*sk. att. 6.3.9.*),

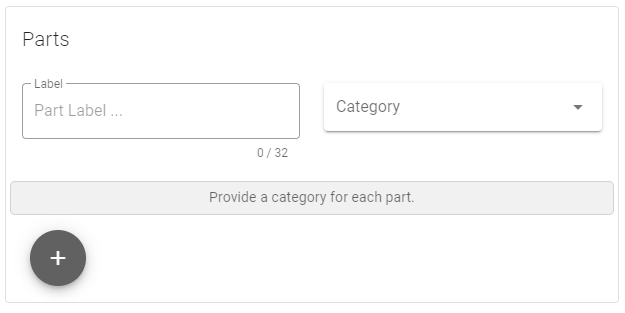
Att. 6.3.9. Sludinājumu pievienošanas poga

kur viņam jāaizpildā sludinājumu forma (*sk. att. 6.3.10.*).

C:\Users\zdkan\Documents\ShareX\Screenshots\2020-05\chrome_5AqKeVOX2C.pngC:\Users\zdkan\Documents\ShareX\Screenshots\2020-05\chrome_lpkpbMSUXQ.pngLai aizvērtu formu bez sludinājuma pievienošanas, lietotājam jāuzspiež poga vai poga *Cancel* sludinājumu formas apakšā (*sk. att. 6.3.11.*).

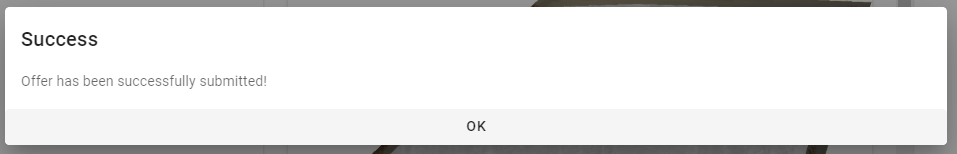
Att. 6.3.11. Sludinājumu formas aizvēršanas poga

Att. 6.3.10. Sludinājumu pievienošanas forma

*C:\Users\zdkan\Documents\ShareX\Screenshots\2020-05\chrome_DaRJpYOP1r.png*Lai pievienotu sludinājumu, lietotājam jāizpilda virsraksts, apraksts, tālrunis, jāpievieno bildes un detaļas. Lietotājs var pievienot vairākas detaļas, nospiežot pogu, kur katrai no tiem ir jāuzrādā kategorija (*sk. att. 6.3.12.*).

Att. 6.3.12. Detaļu ievienošana

Nospiežot pogu *Submit* (*sk. att. 6.3.13.*),

ja visas validācijas bija veiksmīgas, sludinājums tiks pievienots un sistēma paziņo lietotāju par veiksmīgu pievienošanu (*sk. att. 6.3.14.*).

Att. 6.3.14. Sludinājumu veiksmīga pievienošana

Att. 6.3.13. Sludinājumu atsūtīšanas poga

C:\Users\zdkan\Documents\ShareX\Screenshots\2020-05\chrome_drjd2uJd5n.pngC:\Users\zdkan\Documents\ShareX\Screenshots\2020-05\chrome_8t45UsRmNE.pngC:\Users\zdkan\Documents\ShareX\Screenshots\2020-05\chrome_8t45UsRmNE.pngC:\Users\zdkan\Documents\ShareX\Screenshots\2020-05\chrome_drjd2uJd5n.pngJa lietotājs ir piereģisrēts, viņam ir iespēja pievienot sludinājumu no citiem lietotājiem pie savam grāmatzīmēm, nospiežot pogu , un, lai noņemtu no tās – pogu.

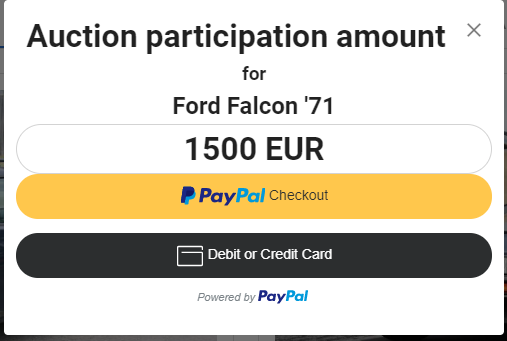
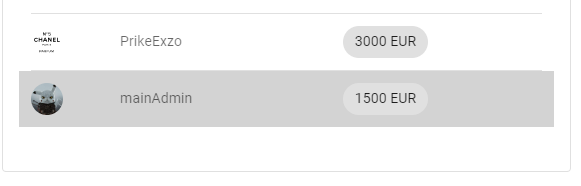
Piereģistrētājiem lietotājiem ir iespēja arhivēt savu sludinājumu (ja lietotāju loma ir administrators vai virs administrators, tad viņam ir iespēja to izdarīt ar jebkura lietotāja sludinājumu) nospiežot pogu *Delete* (*Archive*, ja lietotāju loma ir administrators vai virs administrators) (*sk. att. 6.3.15.*).

Att. 6.3.15. Arhivēšanas pogas

Izmantojot navigāciju un uzspiežot un sadaļu *Auction*, lietotājam būs redzams skats ar pēdējas izsolēm un tas vēsturi (*sk. att. 6.3.16.)*.

Ja lietotājs ir piereģistrēts, viņam ir iespēja piedalīties kādā no izsolēm.

Labdarības izsole (*sk. att. 6.3.17.)* prasa lietotājuuzrādīt naudas apjomu un apstiprināt ar pogu *Participate* (*sk. att. 6.3.18.*), pēc kuras lietotājam būs parādīts dialogs ar visu informāciju par viņa transakciju un maksājuma veidiem (*sk. att. 6.3.19.*).

Lai apstiprinātu maksājumu, lietotājam jānospiež *PayPal* *Checkout* vai *Debit or Credit Card* (*sk. att. 6.3.19.*) poga un sēkot PayPal servisa instrukcijam. Ja operācija izpildas, lietotājs būs paziņots par veiksmīgu transakciju un pievienots pie izsoles dalībniekiem kas ir redzami zem izsoles eksemplāra (*sk. att. 6.3.20.*).

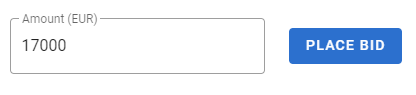
Att. 6.3.20. Izsoles dalībnieki

Att. 6.3.17. Labdarības izsole

Att. 6.3.19. Transakcijas informācija

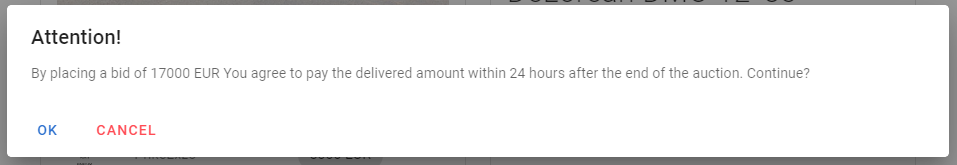
Att. 6.3.18. Labdarības izsoles piedalīšanas

Att. 6.3.16. Izsoles skats

Komericālā izsole (*sk. att. 6.3.21.)* prasa lietotājuuzrādīt naudas apjomu

Att. 6.3.22. Komericālas izsoles piedalīšanas

Att. 6.3.21. Komericālā izsole

un apstiprināt ar pogu *Place Bid* (*sk. att. 6.3.22.*), pēc kuras lietotājam būs parādīts dialogs par piedalīšanas apstiprināšanu (*sk. att. 6.3.23.*).

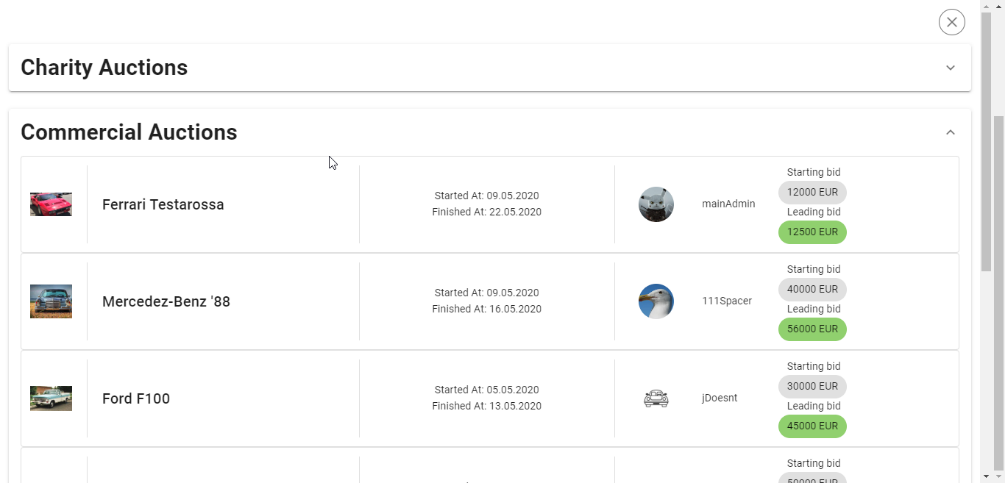
Att. 6.3.23. Piedalīšanas noteikumi

Lietotājam ir jāuzspiež poga *Ok*, lai piekrītu vai *Canvel*, lai atteikties (*sk. att. 6.3.23.*). Ja lietotājs piekrīt noteikumiem, viņš būs paziņots par veiksmīgu dalībniecību izsolē un pievienots pie izsoles dalībniekiem kas ir redzami zem izsoles eksemplāra (*sk. att. 6.3.20.*).

Lai apskatītu izsoles vēsturi, lietotājam ir nepieciešami nospiezt pogu *History*, kas atrodas lapas apakšā (*sk. att. 6.3.24.*).

Att. 6.3.24. Izsoles vēstures poga

Šeit lietotājam būs pieejams saraksts ar visiem iepriekšējam izsolēm (*sk. att. 6.3.25.*).

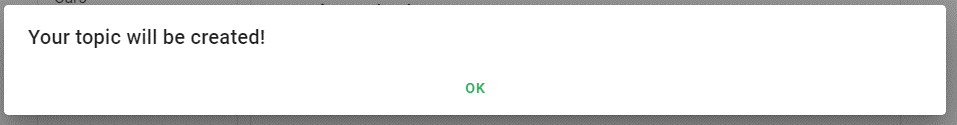
Izmantojot navigāciju un uzspiežot un sadaļu *Forum*, lietotājam būs redzams skats forumu (*sk. att. 6.3.26.)*.

Att. 6.3.26. Forumu skats

Att. 6.3.25. Izsoļu vēsture

Lai apskatītu foruma ziņas, lietotājam ir jāizvēlas kategoriju no saraksta (*sk. att. 6.3.27.)*, pēc kuras lietotājam būs redzami visas ziņas atteicīgajā kategorijā (*sk. att. 6.3.28.)*.

Ja lietotājs ir piereģistrēts, viņam ir iepēja pievienot ziņu, nospiežot pogu *Create Topic* (*sk. att. 6.3.29.*), pēc kuras parādās forma, kuru ir nepieciešami aizpildīt un nospiest *Submit*, lai pievienotu ziņu (*sk. att. 6.3.30.*). Pēc veiksmīgām validācijam, sistēma pzaiņo lietotāju par pievienotu ziņu (*sk. att. 6.3.31.).*

C:\Users\zdkan\Documents\ShareX\Screenshots\2020-05\chrome_4ONPxUR9GK.png Piereģistrētājiem lietotājiem ir iespēja nodzēst savu ziņu vai komentāru (ja lietotāju loma ir administrators vai virs administrators, tad viņam ir iespēja to izdarīt ar jebkura lietotāja ziņu un / vai komentāru) nospiežot pogu.

Att. 6.3.31. Ziņu veiksmīga pieveinošana

Att. 6.3.30. Ziņu pievienošanas forma

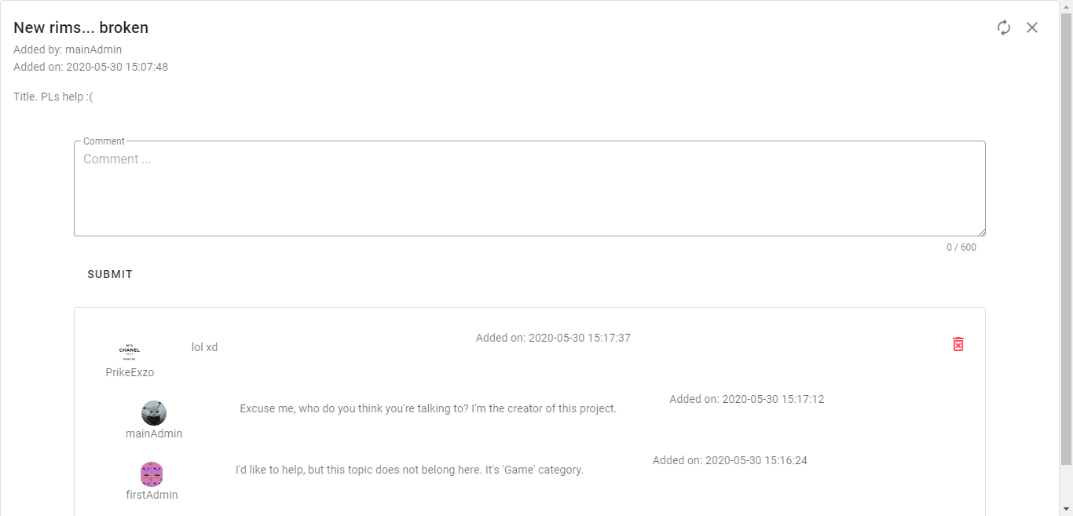
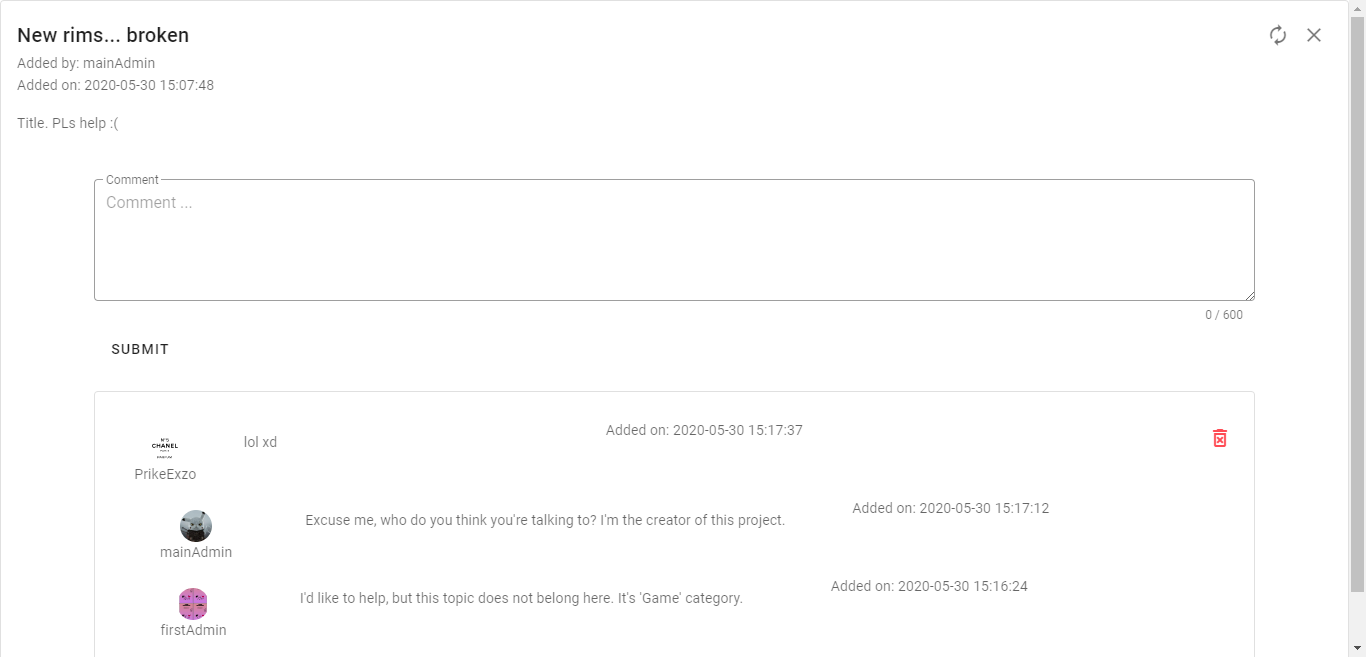
Att. 6.3.29. Ziņu pievienošanas poga

Att. 6.3.28. Forumu ziņas

Att. 6.3.27. Forumu kategorijas

Nospiežot pogu *Expand* blakus ar ziņu, parādas ziņas skats (*sk. att. 6.3.32.*), kur lietotājam ir iespēja apskatīt ziņas aprakstu un komentārus.

Ja lietotājs ir piereģistrēts, viņam ir iespēja pievienot komentāru ziņam, izmantojot teksta laukumu un pogu *Submit*, lai pieivenotu komentāru (*sk. att. 6.3.33.*).

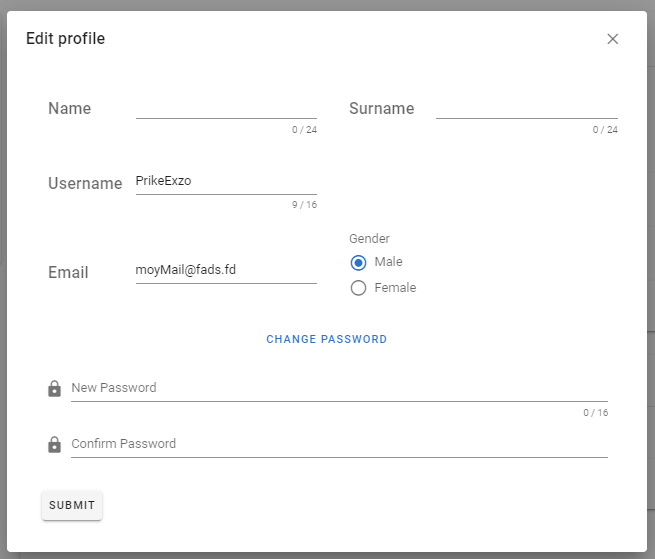
 Ja lietotājs ir piereģistrēts, tad, zmantojot navigāciju un uzspiežot un sadaļu *Dashboard*, lietotājam būs redzams viņa profila skats (*sk. att. 6.3.34.)*.

Att. 6.3.34. Profila skats

Att. 6.3.33. Komentāru pievienošana

Att. 6.3.32. Ziņas skats

Šeit lietotājam ir iespēja rediģēt savu profilu, uzspiežot uz pogu *Edit* uz viņu kartītes (*sk. att. 6.3.35.)*.

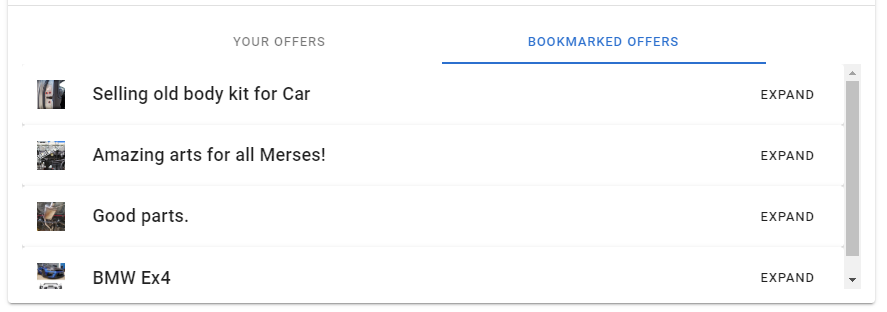
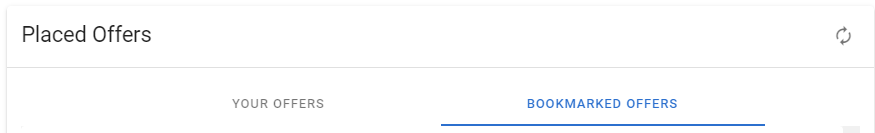
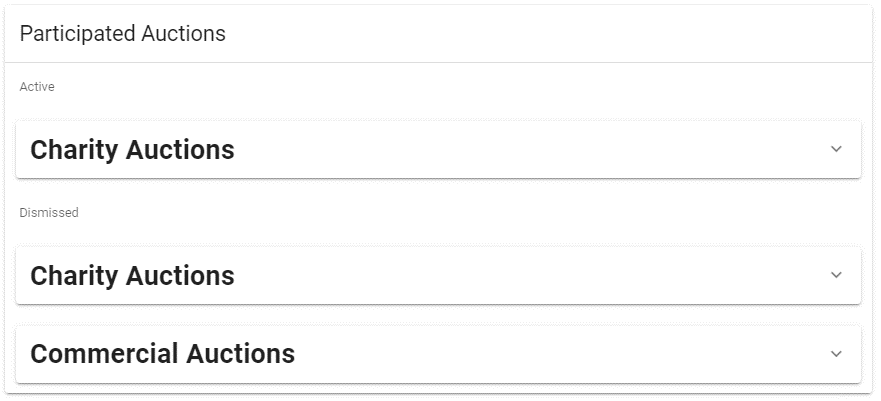
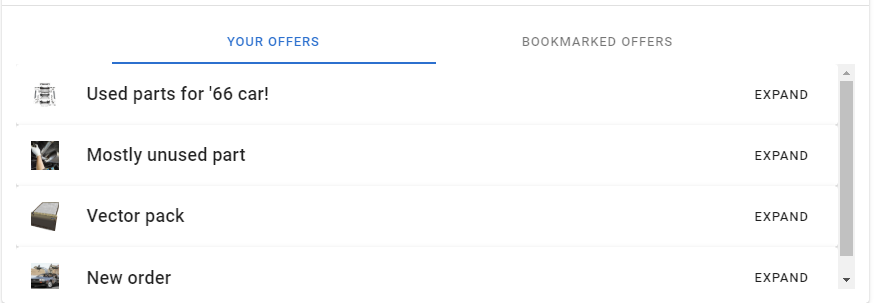
Rediģēšanas formā (*sk. att. 6.3.36.*) lietotājam ir iespēja nomainīt savu informāciju, sekojot validācijas noteikumiem. Ja validācija bija veiksmīga pēc pogas *Submit* nospiešanas, lietotāja dati būs izmanīti.

Att. 6.3.36. Rediģēšanas forma

Att. 6.3.35. Profila kartīte

Lai nomainītu profila bildi, lietotājam jānospiež uz savu tekošo bildi profila kartītē (*sk. att. 6.3.35.*), izvēlēt bildi un uzspiezt *Upload* pogu.

Lietotāja paneļā ir pieejāmi visi lietotāja pievienoti sludinājumi, kas atrodas sadaļā *Your Offers* (*sk. att. 6.3.37.*) un sludinājumi, kas ir pievienoti pie grāmatzīmēm, kas atrodas sadaļā *Bookmarked Offers* (*sk. att. 6.3.38.*) zem *Placed Offers* nodaļas (*sk. att. 6.3.39.*).

 Lietotāja paneļā ir pieejāmi visas izsoles, kuros viņs piedalījas. Tie dalas uz labdarības un komerciālām un aktīvām / neaktīvām izsolēm, kas atrodas sadaļā *Active* un *Dismissed* zem *Participated Auctions* nodaļas (*sk. att. 6.3.40.*).

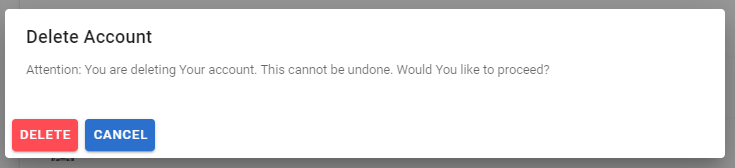
Att. 6.3.40. Lietotāja izsoles

Att. 6.3.39. Profila sludinājumi

Att. 6.3.38. Sludinājumu grāmatzīmes

Att. 6.3.37. Lietotāja sludinājumi

Lietotājam ir iespēja nodzēst savu profilu, izmantojot pogu *Delete* (*sk. att. 6.3.35.*), pēc kura sistēma izvada paziņojumu par lietotāja operāciju (*sk. att. 6.3.41.*). Ja lietotājs gribētu nodzēst profilu, viņam būtu jāspiež pogu *Delete* vai *Cancel*, lai atteikties no tā.

 Ja lietotājs ir piereģistrēts un viņu loma ir Administrators vai Virs Administrators, tad, zmantojot navigāciju un uzspiežot un sadaļu *Control Panel*, lietotājam būs redzams skats ar sistēmas pārvaldes rīkiem (*sk. att. 6.3.42.)*.

Att. 6.3.42. Administratora paneļa skats

Att. 6.3.41. Profila dzēšana

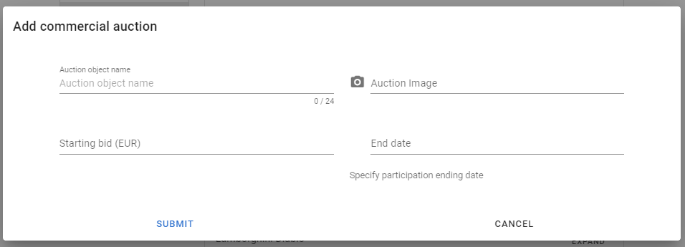
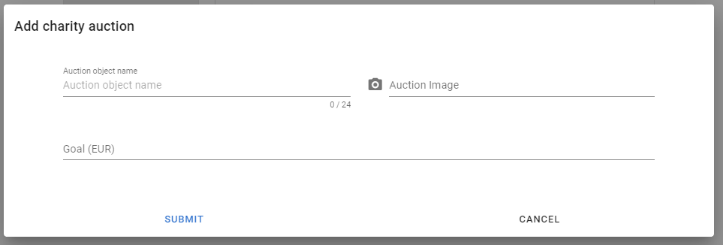
 Kreisajā pusē atrodas izvēlne (*sk. att. 6.3.43.*), kur katrs punkts atbild par konkrēto platformas sadaļas pārvaldi.

Att. 6.3.43. Paneļa izvēlne

Punkts *Auctions* atbild par izsoles pievienošanu un to beigšanu (*ir redzams att. 6.3.42.*).

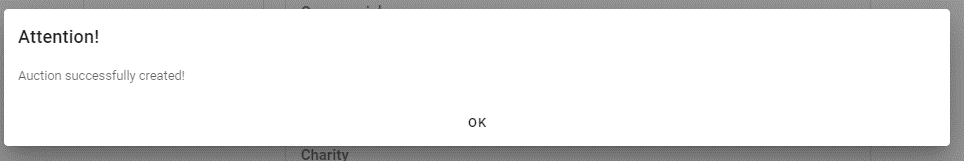
Ja izsole vēl nav pievienota (labdarības un / vai komerciālā), administratoram ir iespēja pievienot to, nospižot pogu *Add Auction* (*sk. att. 6.3.44.*) atbilstošā izsoles kategorijā (*charity / commercial*).

Att. 6.3.44. Izsoles pievienošanas poga

Sistēmā vienlaicīgi varētu būt tikai 1 aktīva izsole kartā no kategorijām. Labdarības izsoles pievienošanas forma (*sk. att. 6.3.45.*) atšķiras no komerciālas izsoles pievienošanas formas forma (*sk. att. 6.3.46.*).

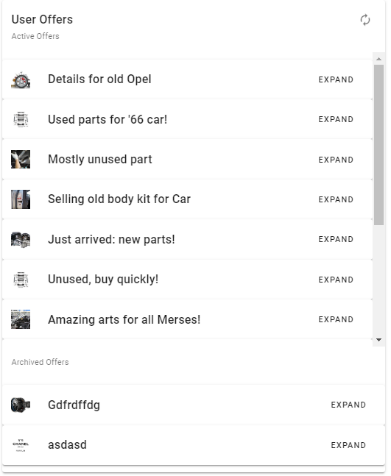
Att. 6.3.45. Labdarības izsoles pievienošanas forma

Att. 6.3.46. Komerciālas izsoles pievienošanas forma

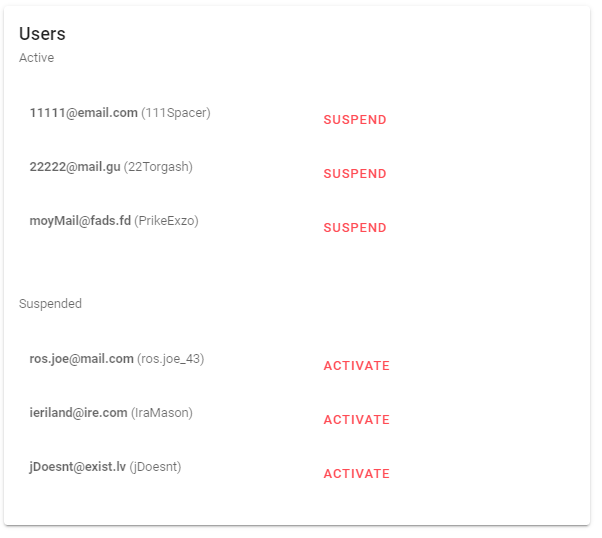
Pēc formas aizpildīšanas, administratoram jāuzspiež poga *Submit* lai pievienotu jaunu izsoli. Pēc tam administrators būs paziņots par pie vienoto izsoli (*sk. att. 6.3.47.*).

Att. 6.3.47. Izsoles veiksmīga pievienošana

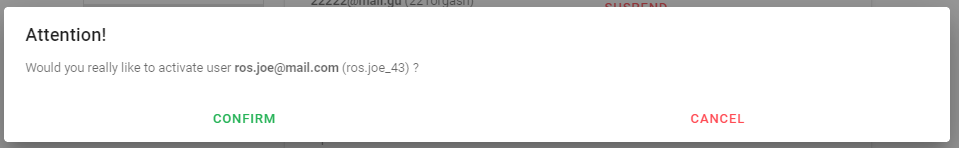
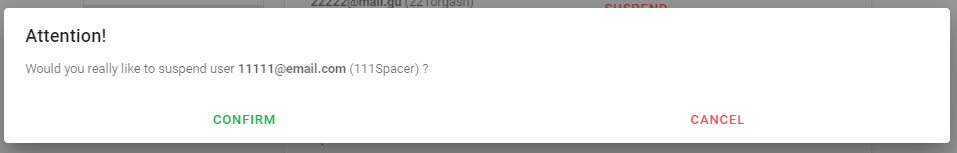
Punkts *Offers* atbild par sludinājumu arhivēšanu (*sk. att. 6.3.48.*).

Nospiežot pogu *Expand* blakus ar sludinājuma virsrakstu, administrators var redzēt sludinājuma informāciju. Gadījumā, kad sludinājums atrodas sadaļā *Active Offers* (*ir redzams att. 6.3.48.*), sludinājuma informāciju logā administratoram ir iespēja arhivēt to, nospiežot pogu *Archive* (*sk. att. 6.3.15.*), pievienotjot pie arhivētiem sludinājumiem (*ir redzams att. 6.3.48.*).

Att. 6.3.48. Sludinājumu pārvalde

Punkts *Users* atbild par lietotāju pārvaldi (*sk. att. 6.3.49.*).

Att. 6.3.49. Lietotāju pārvalde

Lietotāji ar lomu lietotājs tiek sadalīti 2 sarakstos – aktīvie un apturēti. Administratoram ir iespēja aptūrēt lietotāju, nospiežot pogu *Suspend*, pēc kuras sistēma prasa apstiprināt lēmumu (*sk. att. 6.3.50.*) vai atjaunot lietotāja darbību, nospiežot pogu *Activate*, pēc kuras sistēma prasa apstiprināt lēmumu (*sk. att.* *6.3.51.*).

Att. 6.3.51. Lietotāju aktivācijas dialogs

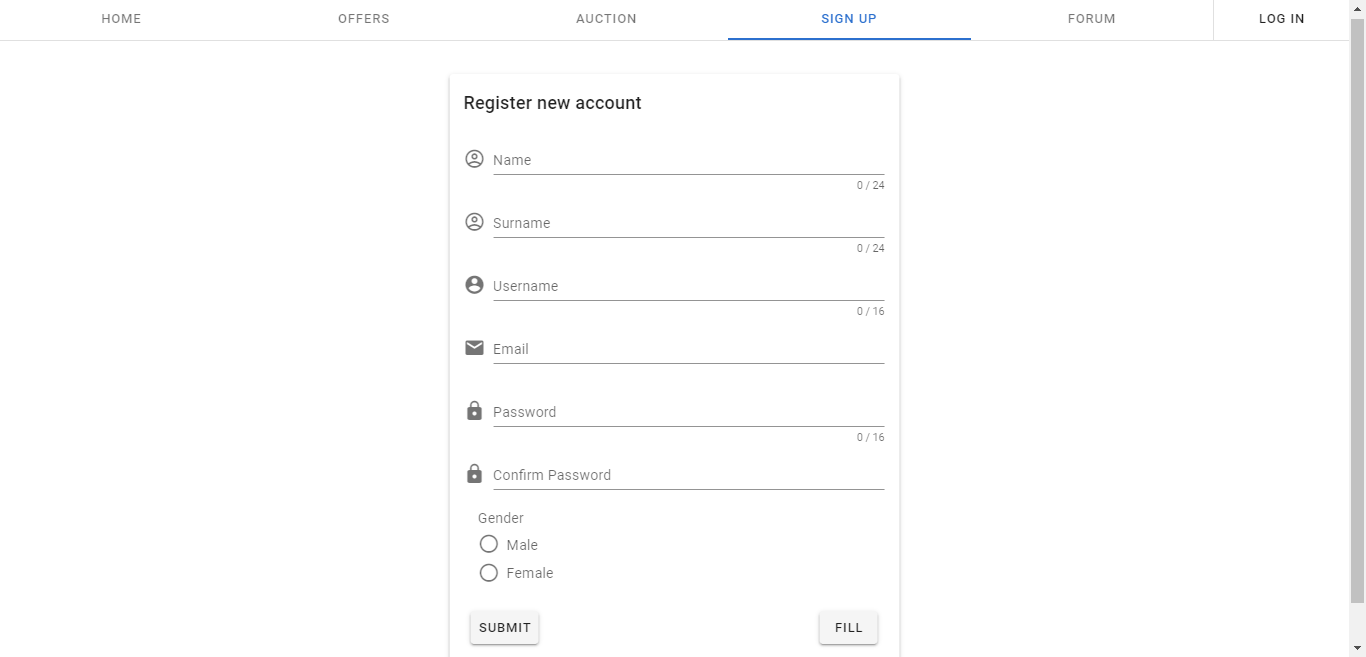
Att. 6.3.50. Lietotāju aptūrēšanas dialogs

 Punkts *Admins* atbild par administartoru pārvaldi (*sk. att. 6.3.52.*) un ir pieejāms tikai lietotājiem ar lomu virs administrators.

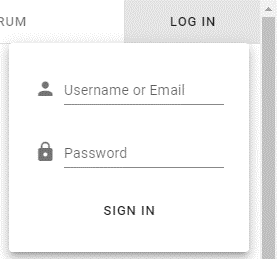
Att. 6.3.52. Administartoru pārvalde

Lietotāji ar lomu tiek sadalīti 2 sarakstos – aktīvie lietotāji un administratori. Virs Administratoram ir iespēja paaugstināt lietotāju līdz administratoram, nospiežot pogu *Promote*, pēc kuras sistēma prasa apstiprināt lēmumu (*sk. līdzīgs att. 6.3.50.*) vai pazemināt administratoru līdz parastam lietotājam, nospiežot pogu *Remove*, pēc kuras sistēma prasa apstiprināt lēmumu (*sk. līdzīgs att. 6.3.51.*).

Ja lietotājs nav pieteikts sistēmā, viņam būs pieejāma *Sign Up* sadaļa (*sk. att. 6.3.53.)*.

 Šeit lietotājam ir nepieciešami aizpildīt nepieciešamus laukus un nospiest *Submit*, lai apstiprinātu reģistrāciju. Ja validācija bija veiksmīga, sistēma pāradresēs lietotāju uz sākumlapu (*sk. att. 6.3.1*).

Att. 6.3.53. Reģistrācijas skats

Lai pieteikties sistēmā ir nepieciešami izmantot pogu *Log In* (*sk. att. 6.3.54.*), ievadot savu lietotājvārdu vai e-pastu un paroli. Ja pieteikšanas bija veiksmīga, lietotājs tiks pāradresēts uz profila skatu (*sk. att. 6.3.34.)*.

Att. 6.3.54. Pieteikšanas forma

Lai atteikties no sistēmas ir nepieciešami izmantot pogu *Log Out* (*sk. att. 6.3.55*), pēc kuras sistēma pāradresēs lietotāju uz sākumlapu (*sk. att. 6.3.1*).



Att. 6.3.55. Atteikšanas poga

## Testa piemērs

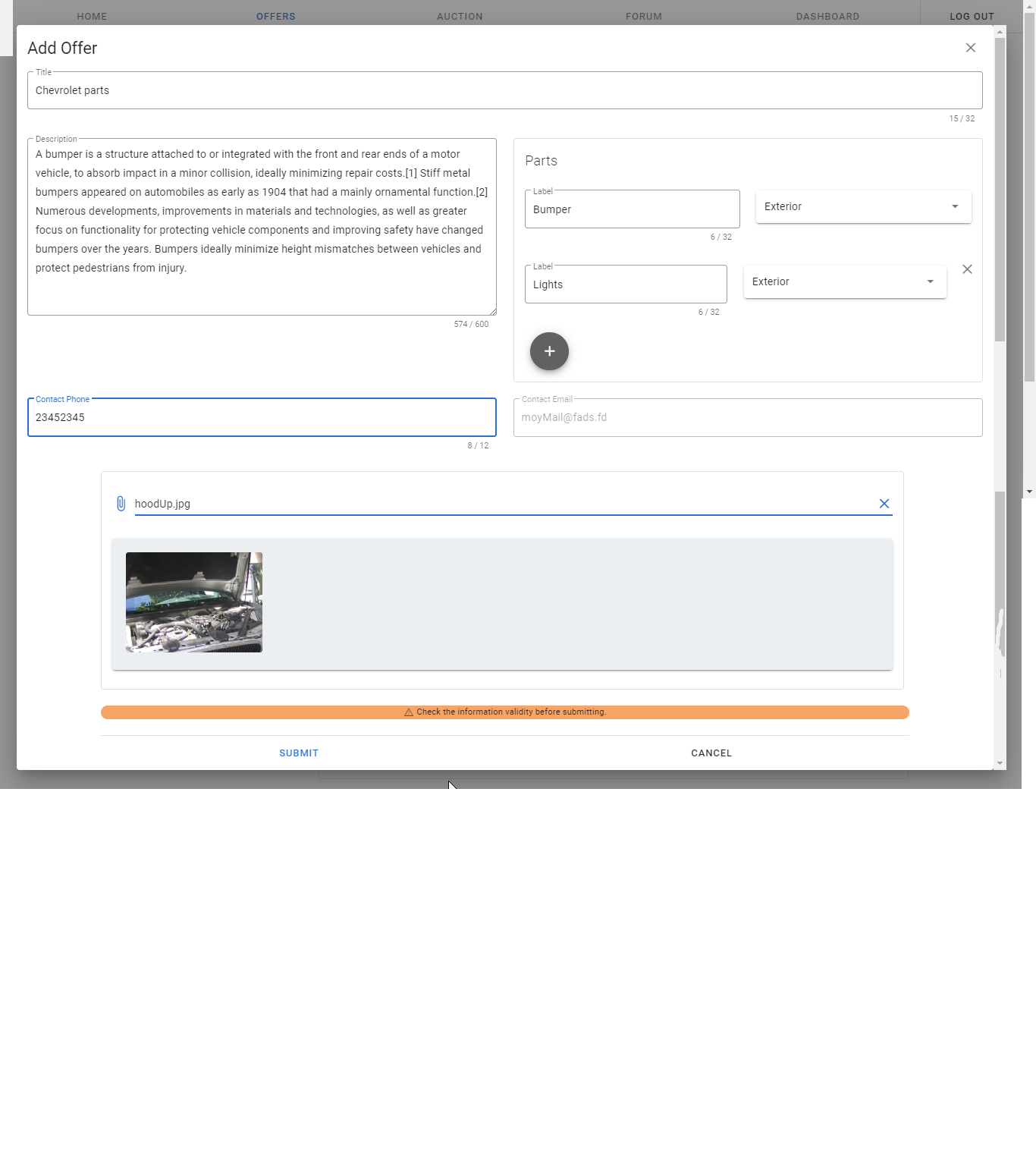
Zemāk aprakstītājs testa piemērs ietvēr sevī lietotāja reģistrāciju, kam seko pieteikšanas sistēmā ar sludinājumu pievienošanu, iziešanu no sistēmas un pievienota sludinājuma aplūkošanu.

1. etaps. Lietotājam jāpiereģistrējas, izmantojot Sign Up sadaļu.

2. etaps. Lietotājam ir nepieciešams pieteikties sistēmā, izmantojot iepriekš izveidoto kontu vai izmantojot eksistējošo profilu.

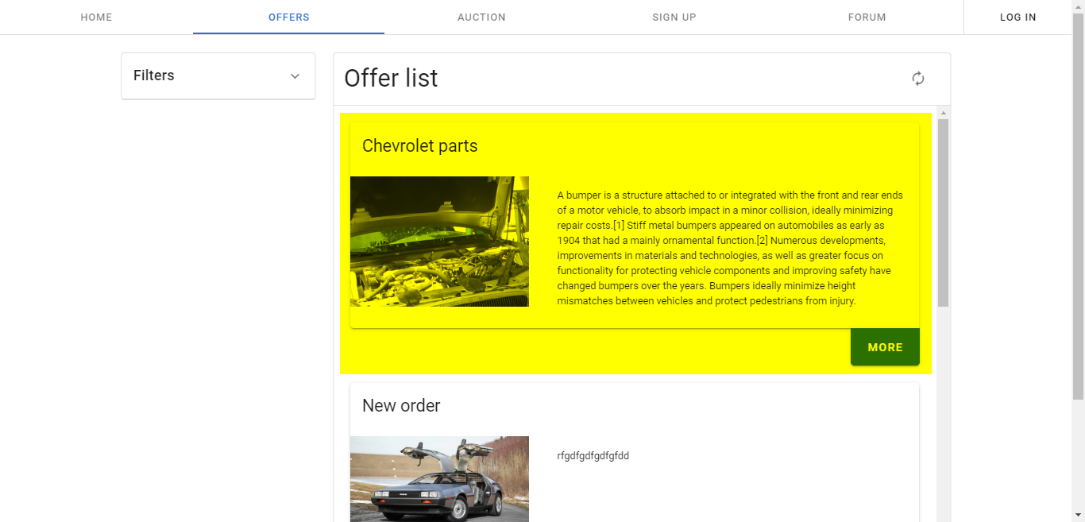
3. etaps. Lietotājam jāaizej uz sadaļu Offers un nospiest pogu Add Offer.

4. etaps. Pēc nosacījumiem aipildīt sludinājuma pievienošanas formu un ievēlēt failus (*sk. att. 6.4.1*).

5 .etaps. Nospiezt pogu Submit, iesniedzot sludinājuma datus.

Att. 6.4.1. Sludinājuma pievienošana

6. etaps. Sludinājums tika pievienots un lietotājs var iziet no sava profila, nospiežot pogu Log Out.

7. etaps. Tagad sludinājums atrodas brīvā pieejā un to ir iespējāmi apskatīt Offers sadaļa bez reģistrācijas (*sk. att. 6.4.2*).

Att. 6.4.2. Pievienots sludinājums

# **Nobeigums**

Šī projekta mērķis ir sasniegts. Tiek realizēts viss, kas bija ieplānots – no sludinājumiem un ziņas pievienošanai, līdz sistēmas pārvaldei un iespējam piedalīties izsolēs.

Sistēmas izmantotas tehnoloģijas – Single Page Application un API pieprasījumi – nodrošina ātrdarbību un ērtāku izmantošanu ne tikai regulāriem lietotājiem, bet arī tas administrācijai.

Projekts ir uztaisīts ar domu turpināt to attīstību, jo sistēma strādāja modulāri. Tas nozīme, ka, pievienojot jauno vai noņemot veco moduli vai funkcionalitāti, aplikācija nesabojājas, jo komponenti ir neatkarīgi viens no otrā. Tas nodrošina vieglāku projekta modifikāciju un to uzlabošanu.

Taisot projektu, galvēna problēma bija pareizi sadalīt laiku uz kātru no portāla daļām. Pieslēdzot transakcijas veikšanas risinājumu no PayPal, kurš apstiprinā naudas operācijas, bija domāts, ka tieši tas aizņems lielāku daļu no projekta darba, bet, subjektīvi, tas programmatūras interfeiss bija saprotamāks, salīdzinot, piemēram, ar Google API un neaizņema tik daudz laika, kā bija plānots no sākuma. Ar to, prioritātes ir mainījušās un tas ļāva labāk atkļūdot darbu un patērēt vairāk laika uz projekta dizainu.

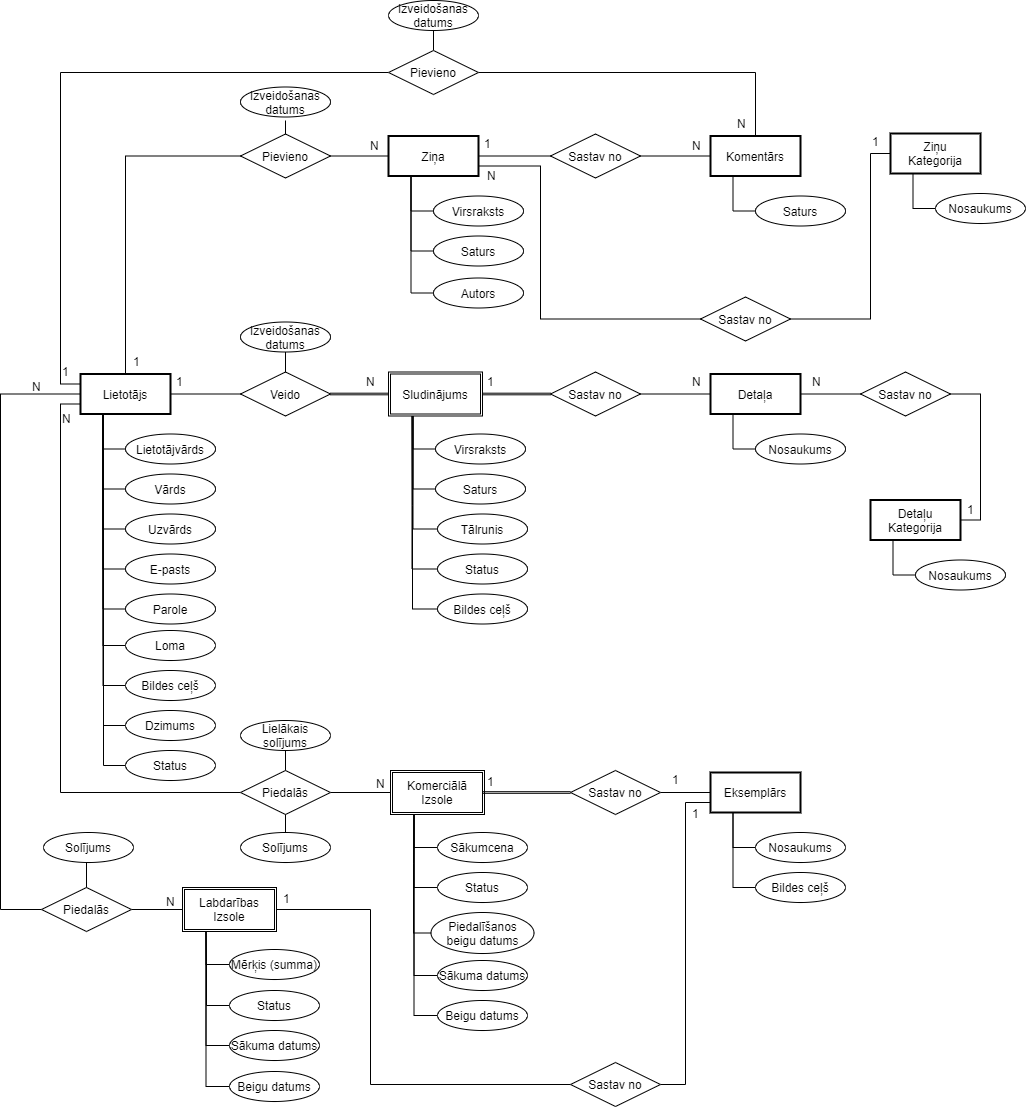
# **Informācijas avoti**

1. Composer programmatūra priekš PHP bibliotēkam – [https://getcomposer.org/download/](https://getcomposer.org/download/%20) – (Resurs apskatīts 30.05.2020).
2. ER diagrammas veidošanas principi –<https://www.estudijas.rvt.lv/pluginfile.php/14954/mod_resource/content/1/DB_logiska_projektesana.pdf> – (Resurs apskatīts 30.05.2020).
3. ER diagrammas, datu plūsmu un funkcionālas dekompozīcijas diagrammu veidošana – [https://www.draw.io](https://www.draw.io/) – (Resurs apskatīts 30.05.2020).
4. Git programmatūra – <https://git-scm.com/downloads> – (Resurs apskatīts 30.05.2020).
5. MySql pieprasījumu veidošanas dokumentācija – <https://www.mysql.com/> – (Resurs apskatīts 30.05.2020).
6. NodeJS kopā ar NPM (LTS) programmatūra priekš JavaScript bibliotēkam – [https://nodejs.org/en/](https://nodejs.org/en/%20) –– (Resurs apskatīts 30.05.2020).
7. PasteBin – .env faila glābšanas vieta – <https://pastebin.com/Pq1BLe1w> – (Resurs apskatīts 30.05.2020).
8. Termiņu pārtulkošana – <http://termini.lza.lv/> – (Resurs apskatīts 30.05.2020).
9. Vietnes skices izveidošanas rīki – <https://balsamiq.com/> – (Resurs apskatīts 30.05.2020).
10. WAMP virtuālā servera programmatūra – [https://sourceforge.net/projects/wampserver/](https://sourceforge.net/projects/wampserver/%20) – (Resurs apskatīts 30.05.2020).

# **Pielikumi**

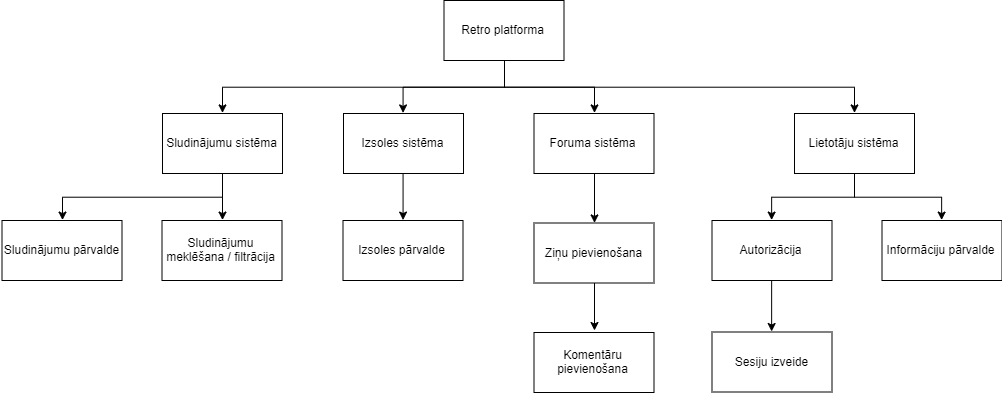
## **1. pielikums**

**ER diagramma**



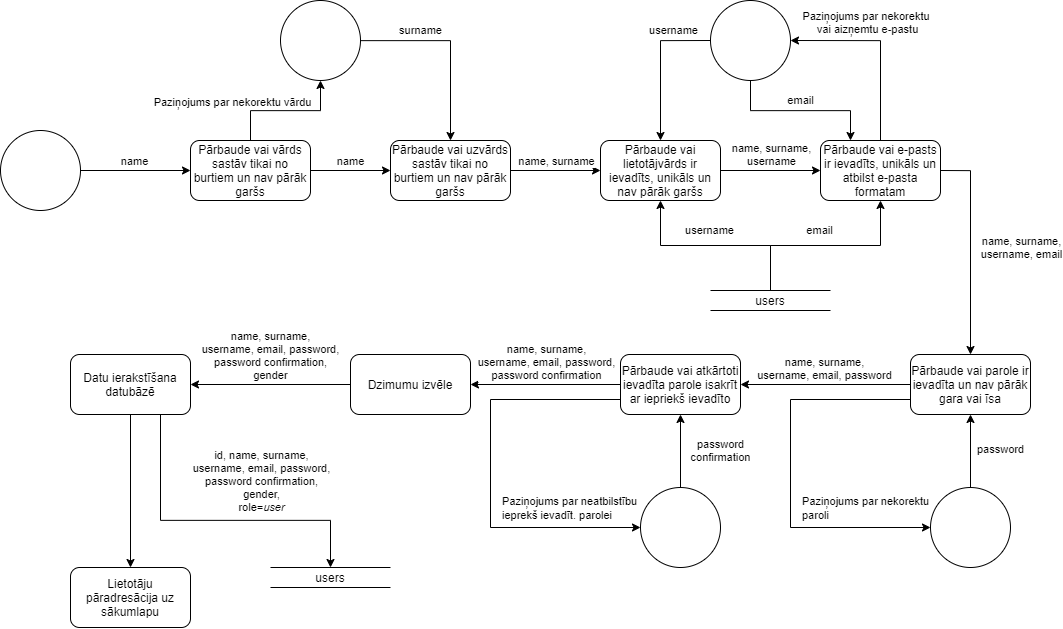
## **2. pielikums**

**Funkcionālas dekompozicijas diagramma**



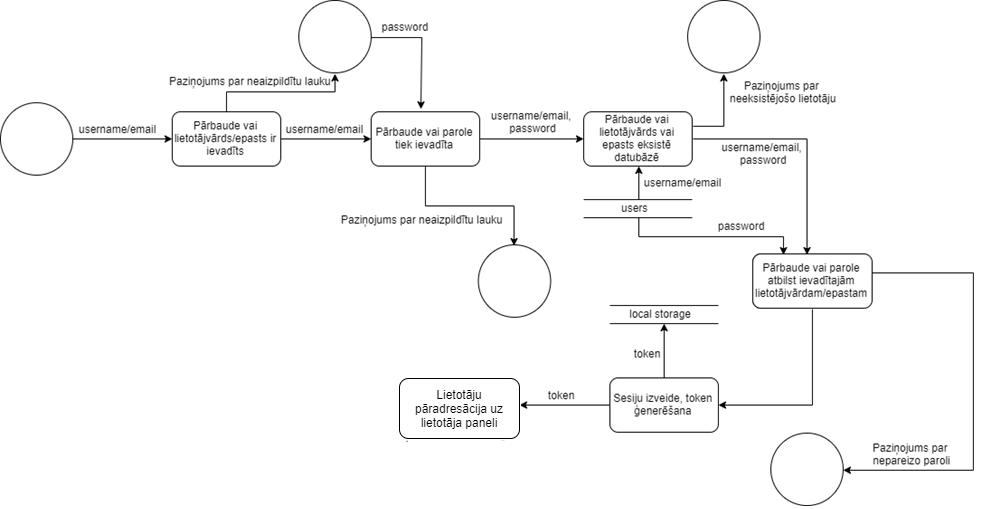
## **3. pielikums**

**Datu plūsmu diagramma – lietotāja reģistrācija**



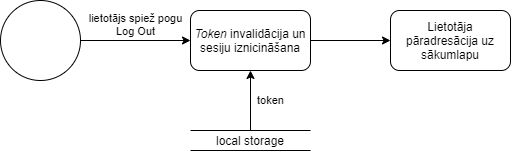
## **4. pielikums**

**Datu plūsmu diagramma – lietotāja pieteikšanas**



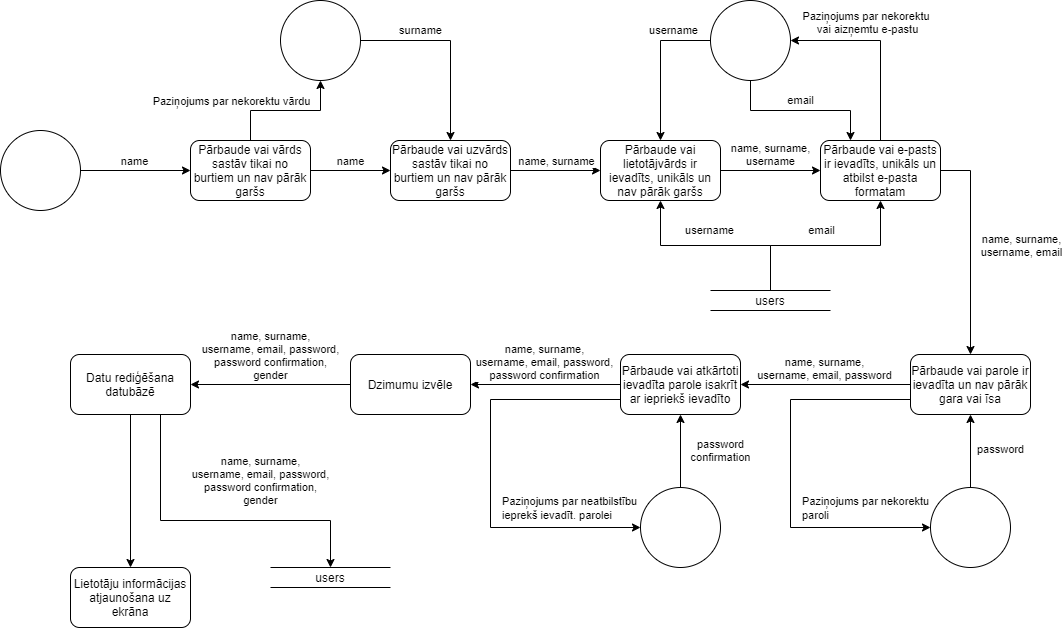
## **5. pielikums**

**Datu plūsmu diagramma – lietotāja atteikšanas**

****

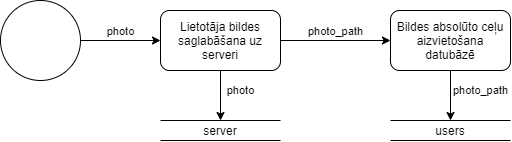
## **6. pielikums**

**Datu plūsmu diagramma – lietotāja datu rediģēšana**

****

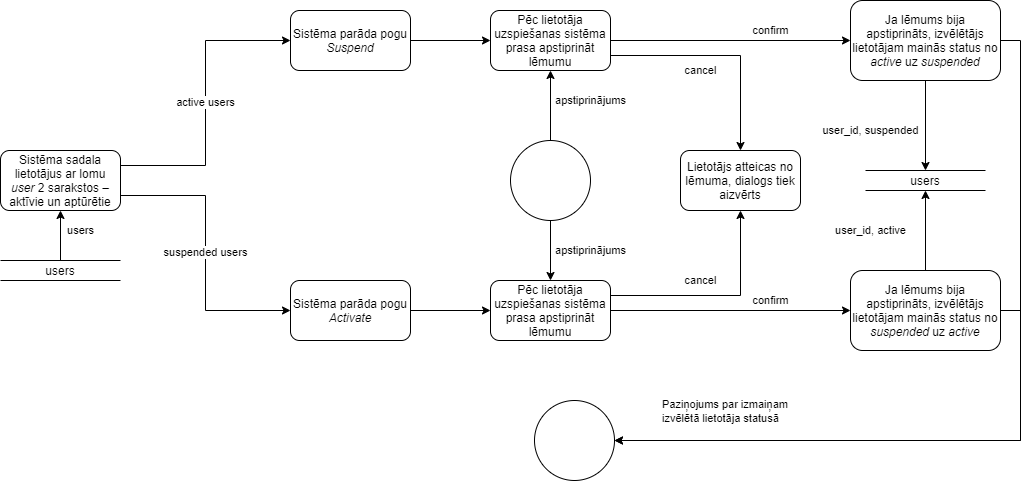
## **7. pielikums**

**Datu plūsmu diagramma – lietotāja bildes maiņa**

****

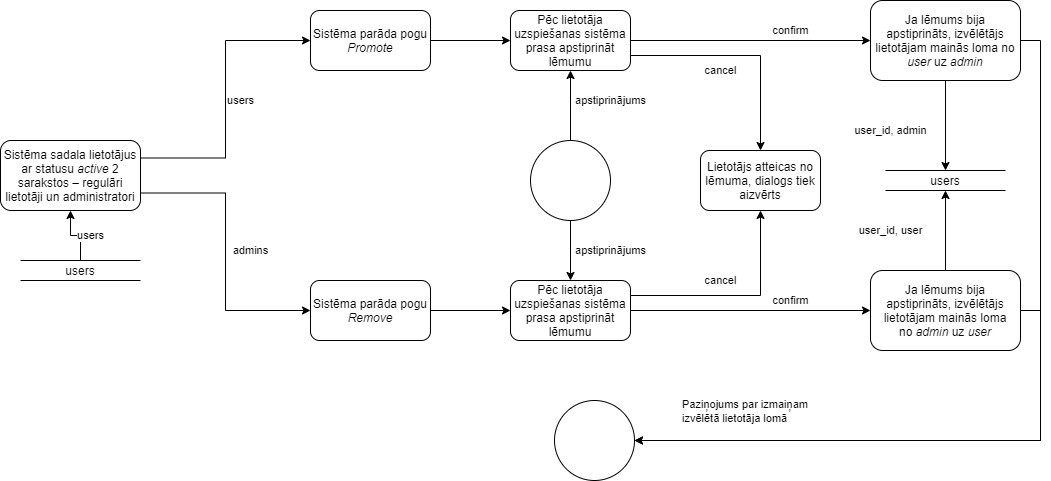
## **8. pielikums**

**Datu plūsmu diagramma – lietotāja profila aptūrēšana**

****

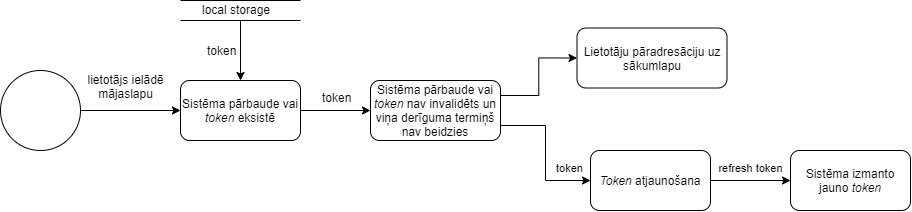
## **9. pielikums**

**Datu plūsmu diagramma – administratoru pievienošana**

****

## **10. pielikums**

**Datu plūsmu diagramma – sesiju izveide**

****

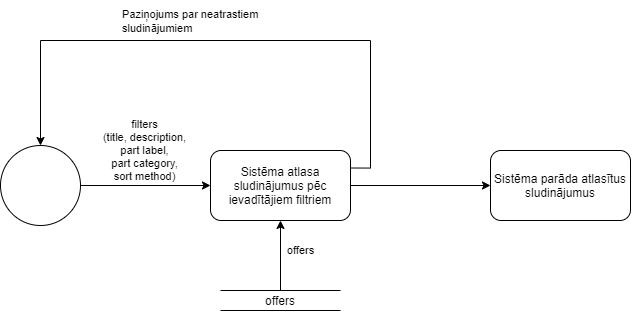
## **11. pielikums**

**Datu plūsmu diagramma – lietotāja lomas pārbaude**

****

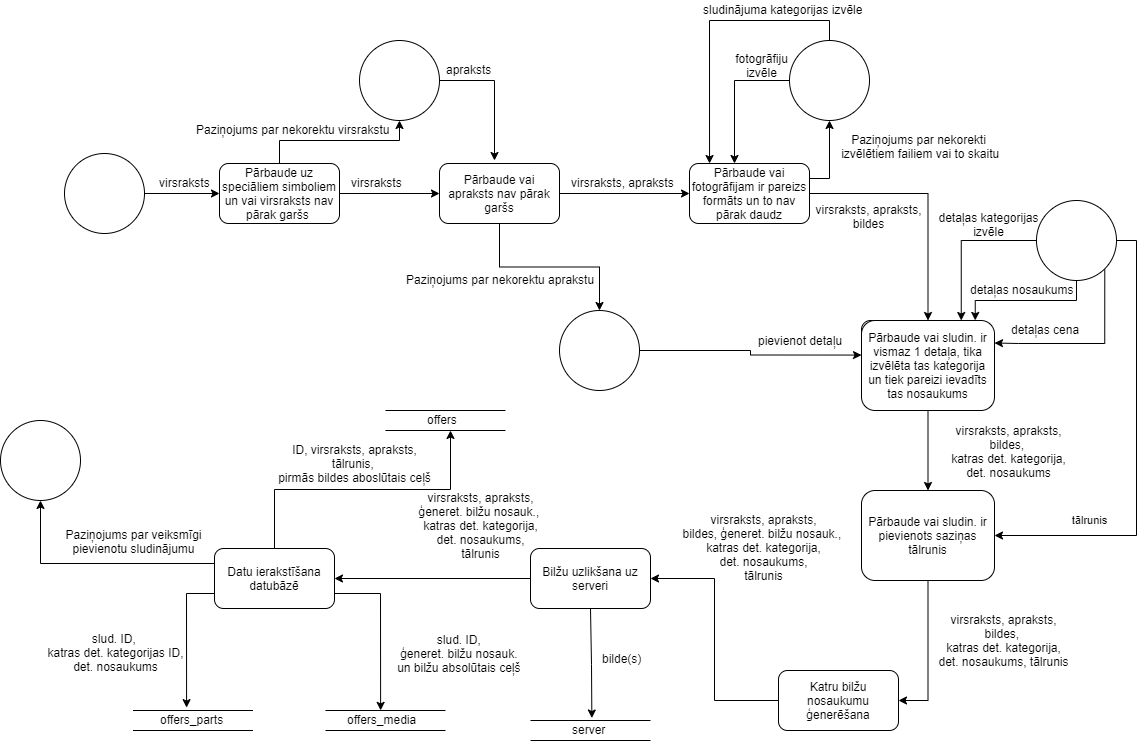
## **12. pielikums**

**Datu plūsmu diagramma – sludinājumu mēklēšana**

****

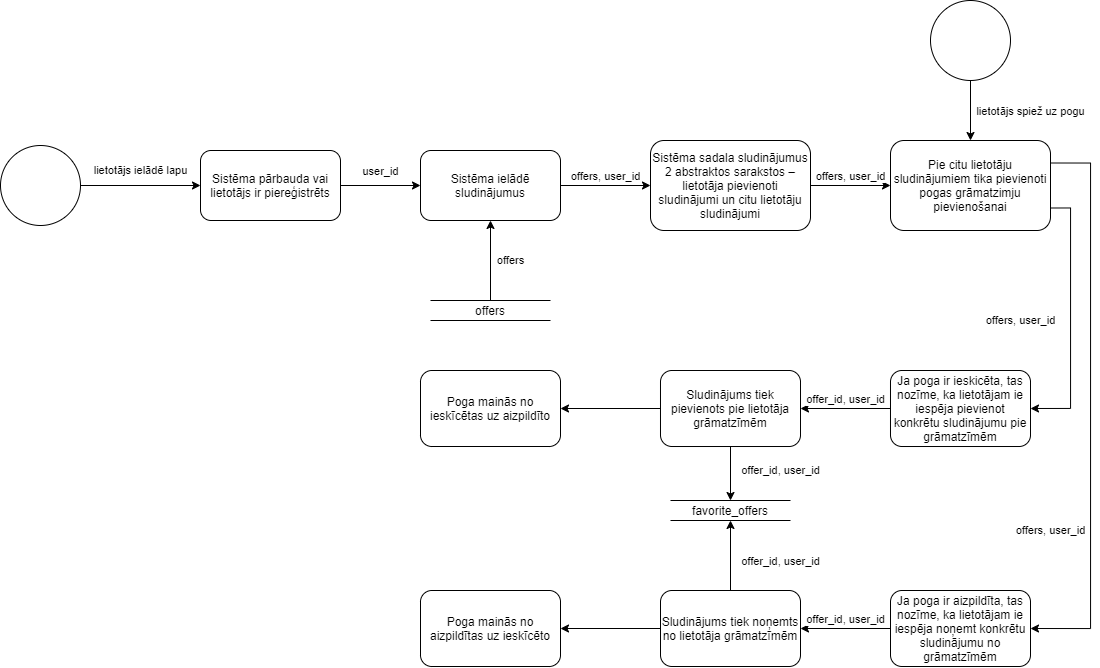
## **13. pielikums**

**Datu plūsmu diagramma – sludinājumu pievienošana**



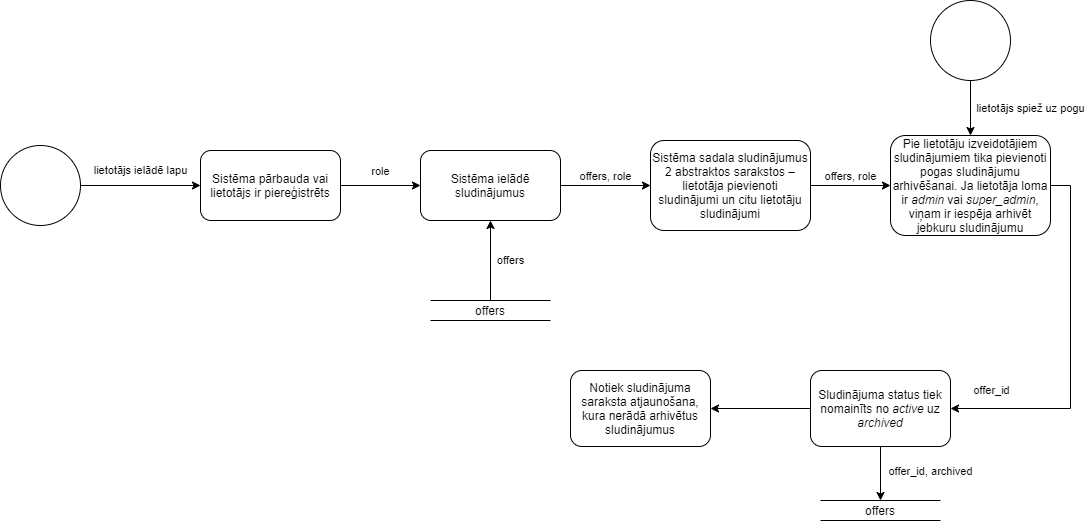
## **14. pielikums**

**Datu plūsmu diagramma – sludinājumu pievienošana pie grāmatzīmēm**

****

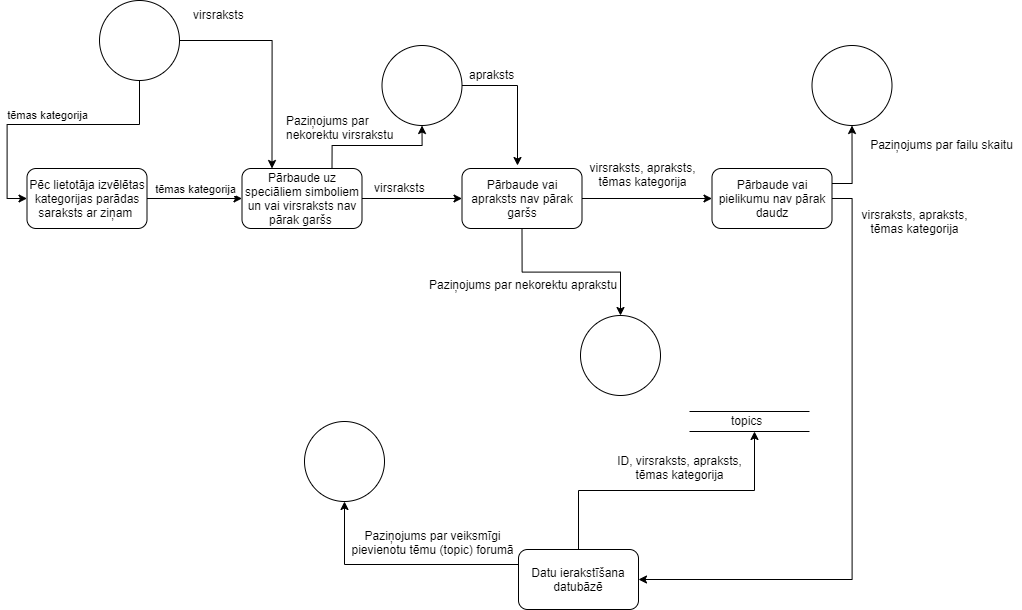
## **15. pielikums**

**Datu plūsmu diagramma – sludinājumu arhivēšana ‘**

****

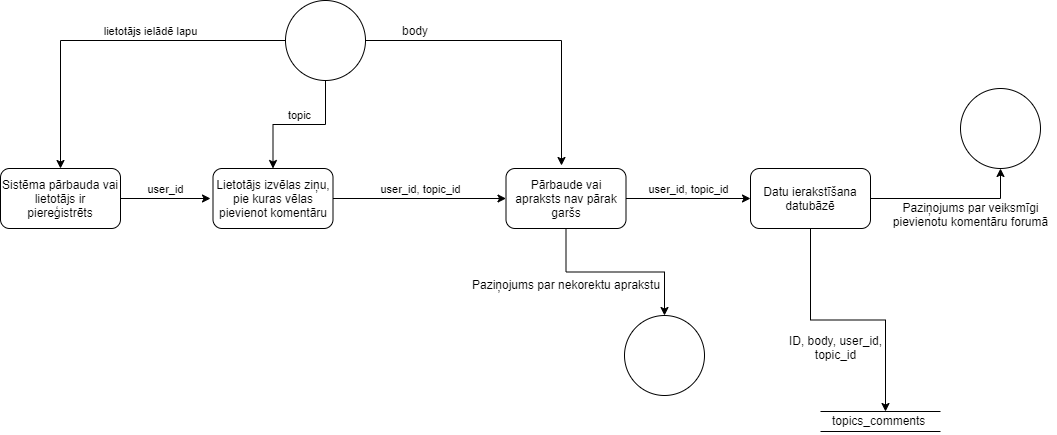
## **16. pielikums**

**Datu plūsmu diagramma – tēmu (topic) pievienošana forumā ‘**

****

## **17. pielikums**

**Datu plūsmu diagramma – komentāru pievienošana ziņai (topic) forumā ‘**

****

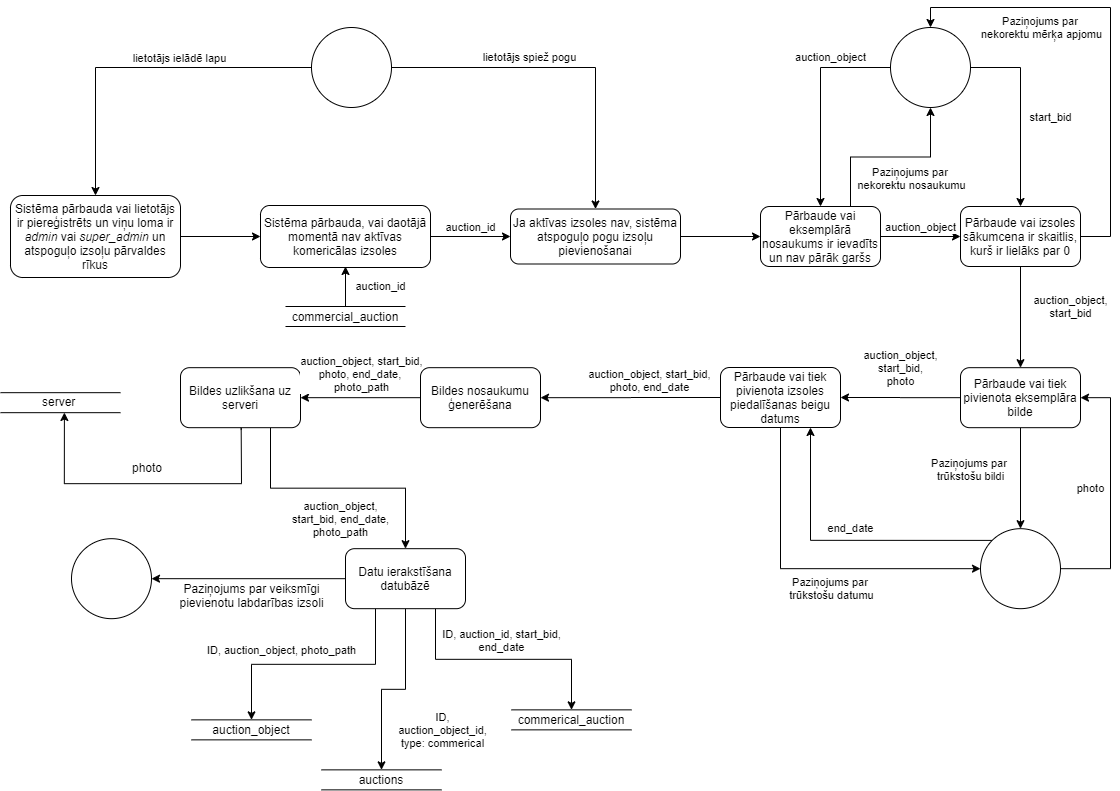
## **18. pielikums**

**Datu plūsmu diagramma – labdarības izsoles pievienošana ‘**

****

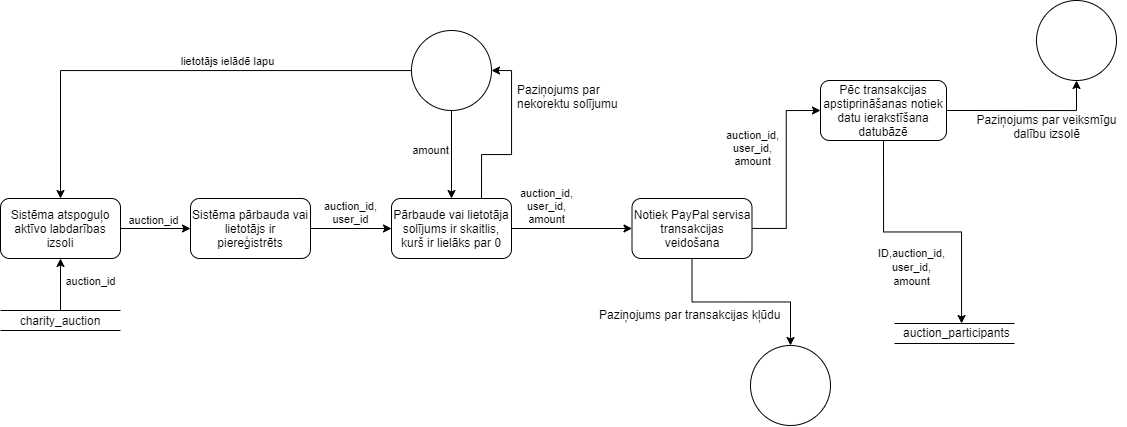
## **19. pielikums**

**Datu plūsmu diagramma – komerciālas izsoles pievienošana ‘**

****

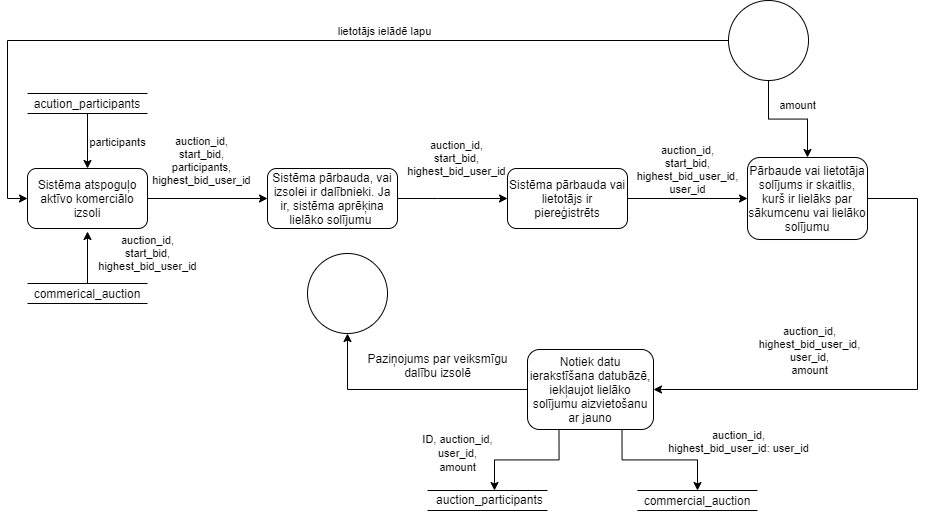
## **20. pielikums**

**Datu plūsmu diagramma – piedalīšanas labdarības izsolē ‘**

****

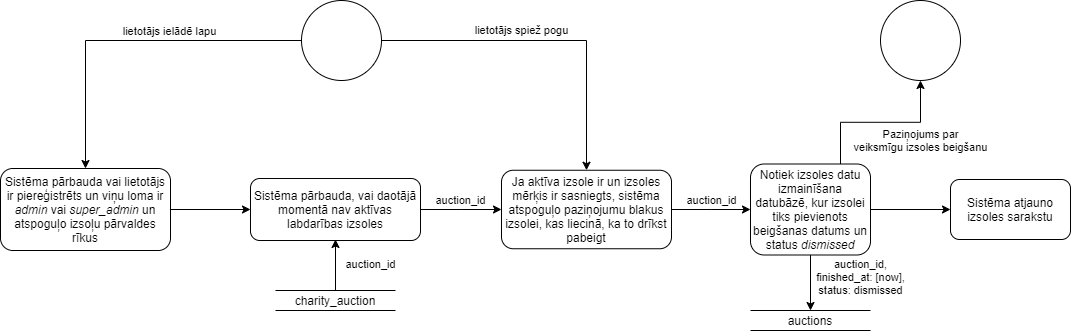
## **21. pielikums**

**Datu plūsmu diagramma – piedalīšanas komerciālā izsolē ‘**

****

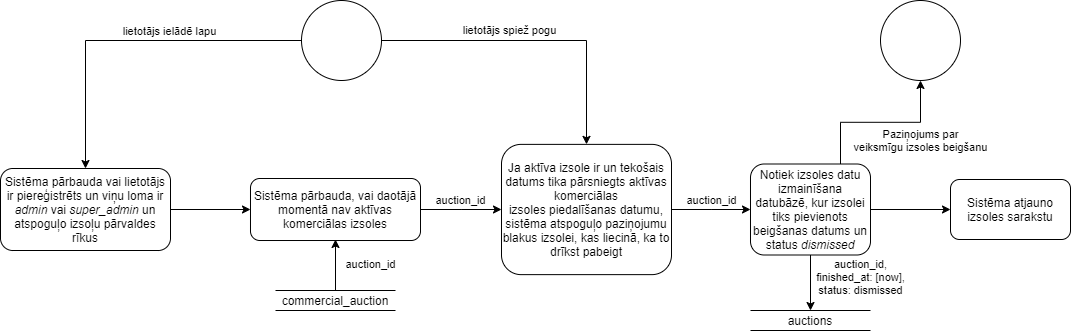
## **22. pielikums**

**Datu plūsmu diagramma – labdarības izsoles beigšana ‘**

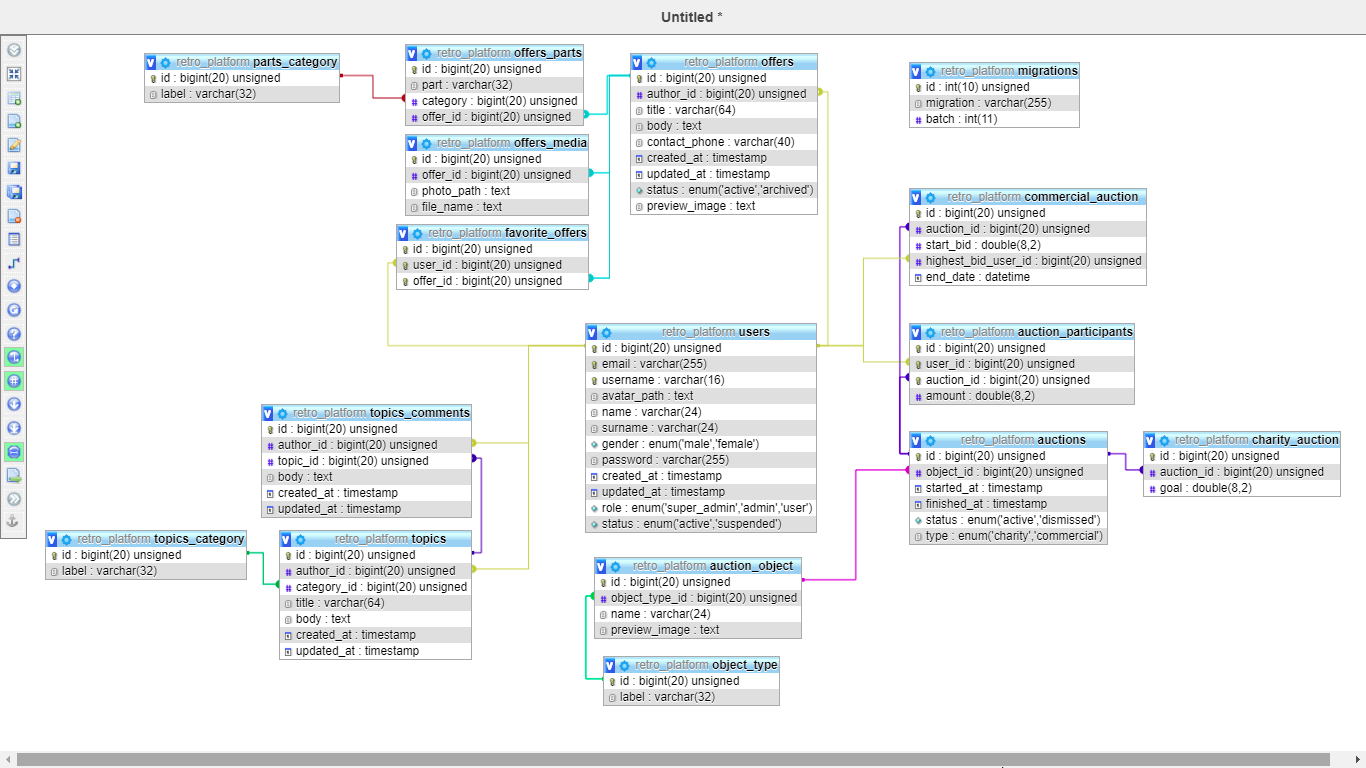
****

## **23. pielikums**

**Datu plūsmu diagramma – labdarības izsoles beigšana ‘**

****

## **24. pielikums**

**ER diagrammas pārveidošana par tabulu relācijām**

## **25. pielikums**

**Programmas pirmkods**

**AuctionController.php**

<?php

namespace App\Http\Controllers;

use App\Auction;

use App\AuctionObject;

use App\AuctionParticipants;

use App\Http\Resources\Auction\AuctionFullDataSet as AuctionFullDataSetResources;

use App\Traits\AuctionTraits;

use Illuminate\Http\Request;

use Illuminate\Support\Collection;

class AuctionController extends Controller

{

    use AuctionTraits;

    /\*\*

     \* Display a listing of the resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function index($status = 'active')

    {

        $raw\_auctions = new Auction;

        $decoded\_auctions = json\_decode($raw\_auctions->all());

        $filteredAuction = array\_filter($decoded\_auctions, function($auction) use ($status) {

            return $auction->status == $status;

        });

        if (sizeof($filteredAuction) == 0) {

            return [];

        }

        foreach ($filteredAuction as $auction) {

            $objectIds[] = $auction->object\_id;

        }

        $auction\_objects = json\_decode(json\_encode($this->getAuctionObjects(AuctionObject::all(), $objectIds)), true);

        foreach ($filteredAuction as $auction) {

            foreach ($auction\_objects as $object) {

                if ($object['id'] == $auction->object\_id) {

                    $auction->auction\_object = $object;

                }

            }

            $auction->participants = $this->getAuctionParticipants(AuctionParticipants::all(), $auction->id);

            $auction->auction\_data = $this->getAuctionTypeData($auction->id, $auction->type);

        }

        return new AuctionFullDataSetResources($filteredAuction);

    }

    public function getUserAuctions(Request $request) {

        $activeAuctions = $this->index();

        $dismissedAuctions = $this->index('dismissed');

        $newAuctions = [];

        for ($i=0; $i < 2; $i++) {

            $filteredAuctions = [];

            if ($i == 0) {

                $auctionBuffer = $activeAuctions;

            } else {

                $auctionBuffer = $dismissedAuctions;

            }

            $count = count(json\_decode(json\_encode($auctionBuffer)));

            for ($j=0; $j < $count; $j++) {

                $toAdd = false;

                if (isset($auctionBuffer[$j])) {

                    foreach ($auctionBuffer[$j]->participants as $participant) {

                        if ($participant->user\_id == $request->user\_id) {

                            $toAdd = true;

                        }

                    }

                    if ($toAdd) {

                        $filteredAuctions[] = $auctionBuffer[$j];

                    }

                } else {

                    $count++;

                }

            }

            $newAuctions[] = $filteredAuctions;

        }

        return $newAuctions;

    }

    public function finishAuction(Request $request) {

        $auctionToFinish = Auction::find($request->auction\_id);

        $auctionToFinish->finished\_at = now();

        $auctionToFinish->save();

        return;

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return \Illuminate\Http\Response

     \*/

    public function store(Request $request)

    {

        $auctionObject = $this->createAuctionObject($request->object\_name, $request->preview\_image);

        $auction = new Auction();

        $auction->object\_id = $auctionObject->id;

        $auction->type = $request->type;

        $auction->save();

        switch ($request->type) {

            case 'charity':

                $this->createCharityAuction($auction->id, $request->goal);

                break;

            case 'commercial':

                $this->createCommercialAuction($auction->id, $request->start\_bid, $request->end\_date);

                break;

        }

        return;

    }

}

**AuctionObjectController.php**

<?php

namespace App\Http\Controllers;

use App\AuctionObject;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Storage;

class AuctionObjectController extends Controller

{

    /\*\*

     \* Show the form for creating a new resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function create($objectName, $previewImage)

    {

        $newLabel = md5(time()+rand()).'.'.$previewImage->getClientOriginalExtension();

        $previewImage->storeAs(

            'public/uploads/auctions/auction\_objects', $newLabel

        );

        $imagePath = Storage::disk('public')->url('uploads/auctions/auction\_objects/'.$newLabel);

        $auctionObject = new AuctionObject();

        $auctionObject->object\_type\_id = 1; // car

        $auctionObject->name = $objectName;

        $auctionObject->preview\_image = $imagePath;

        $auctionObject->save();

        return $auctionObject;

    }

}

**AuctionParticipantsController.php**

<?php

namespace App\Http\Controllers;

use App\AuctionParticipants;

use App\Http\Resources\Auction\AuctionParticipants as AuctionParticipantsResources;

use Illuminate\Http\Request;

use Illuminate\Support\Collection;

class AuctionParticipantsController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function index()

    {

        return AuctionParticipantsResources::collection(Auctionparticipants::all());

    }

    public function indexByAuctionType($auctionType)

    {

        return AuctionParticipantsResources::collection(Auctionparticipants::where('auction\_type', $auctionType));

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return \Illuminate\Http\Response

     \*/

    public function store(Request $request)

    {

        $participant = new AuctionParticipants;

        $participant->auction\_id = $request->auction\_id;

        $participant->user\_id = $request->user\_id;

        $participant->amount = $request->amount;

        $participant->save();

        return response()->json(['status' => 'success'], 200);

    }

    /\*\*

     \* Display the specified resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function show($auctionId, $userId)

    {

        return (new AuctionParticipants)->where('auction\_id', $auctionId)->where('user\_id', $userId)->first();

    }

}

**AuthController.php**

<?php

namespace App\Http\Controllers;

use JWTAuth;

use App\User;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Storage;

use Illuminate\Support\Facades\Validator;

class AuthController extends Controller

{

     /\*\*

     \* Create a new AuthController instance.

     \*

     \* @return void

     \*/

    public function \_\_construct()

    {

        $this->middleware('auth:api', ['except' => ['login']]);

    }

    public function register (Request $request) {

        $v = Validator::make($request->all(), [

            'username' => 'unique:users',

            'email' => 'unique:users',

        ]);

        if ($v->fails()) {

            return response()->json([

                'status' => 'error',

                'errors' => $v->errors()

            ], 422);

        }

        $user = new User;

        $user->name = $request->name;

        $user->surname = $request->surname;

        $user->avatar\_path = Storage::disk('public')->url('service/user\_placeholder.jpg');

        $user->username = $request->username;

        $user->gender = $request->gender;

        $user->email = $request->email;

        $user->password = bcrypt($request->password);

        $user->save();

        return response()->json(['status' => 'success'], 200);

    }

    public function login (Request $request) {

        try{

            $credentials = $request->only('email', 'password');

            if (!($token = auth()->attempt($credentials))) {

                $credentials = $request->only('username', 'password');

                if (!($token = auth()->attempt($credentials))){

                    return response()->json([

                        'error' => 'login\_error'

                    ], 401);

                }

            }

        } catch (JWTException $e) {

            return response()->json(['error' => 'could\_not\_create\_token'], 500);

        }

        $user = Auth::user();

        if ($user->status == 'active') {

            return response([

                'data' => $user

            ])

            ->header('Authorization', 'Bearer '.$token);

        } else {

            return response()->json(['error' => 'account\_suspended'], 401);

        }

    }

    public function logout () {

        auth()->invalidate();

        return response()->json([

            'status' => 'success',

            'msg' => 'Logged out successfully'

        ], 200);

    }

    public function user (Request $request) {

        $user = User::find(Auth::user()->id);

        if ($user->status == 'active') {

            return response()->json([

                'status' => 'success',

                'data' => $user

            ]);

        } else {

            return response()->json(['error' => 'account\_suspended'], 401);

        }

    }

    public function refresh (Request $request) {

        try {

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

            if ($user) {

                if ($user->status == 'active') {

                    $newToken = auth()->refresh();

                    return response()->json(['status' => 'success'], 200)->header('Authorization', $newToken);

                } else {

                    return response()->json(['error' => 'account\_suspended'], 401);

                }

            }

            return response()->json(['error' => 'user\_error'], 401);

        } catch (\Tymon\JWTAuth\Exceptions\UserNotDefinedException $e) {

            return response()->json(['error' => 'refresh\_token\_error'], 401);

        }

    }

    private function guard() {

        return Auth::guard();

    }

}

**CharityAuctionController.php**

<?php

namespace App\Http\Controllers;

use App\CharityAuction;

use App\Http\Resources\Auction\CharityAuction as CharityAuctionResources;

use Illuminate\Http\Request;

use Illuminate\Support\Collection;

class CharityAuctionController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function index()

    {

        return CharityAuctionResources::collection(CharityAuction::all());

    }

    public function getDetailsByAuctionId($auctionId) {

        return CharityAuctionResources::collection(CharityAuction::where('auction\_id', $auctionId)->get());

    }

    /\*\*

     \* Show the form for creating a new resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function create($auctionId, $goal)

    {

        $charityAuction = new CharityAuction();

        $charityAuction->auction\_id = $auctionId;

        $charityAuction->goal = $goal;

        $charityAuction->save();

    }

}

**CommercialAuctionController.php**

<?php

namespace App\Http\Controllers;

use App\CommercialAuction;

use App\Http\Resources\Auction\CommercialAuction as CommercialAuctionResources;

use App\Traits\AuctionTraits;

use Illuminate\Http\Request;

class CommercialAuctionController extends Controller

{

    use AuctionTraits;

    /\*\*

     \* Display a listing of the resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function index()

    {

        return CommercialAuctionResources::collection(CommercialAuction::all());

    }

    public function getDetailsByAuctionId($auctionId) {

        return CommercialAuctionResources::collection(CommercialAuction::where('auction\_id', $auctionId)->get());

    }

    public function checkBid(Request $request) {

        $auction = new CommercialAuction;

        $auction = $auction->where('auction\_id', $request->auction\_id)->first();

        if (is\_null($auction->highest\_bid\_user\_id)) {

            $auction->highest\_bid\_user\_id = $request->user\_id;

            $auction->save();

            return response()->json([

                'status' => 'success',

                'msg' => 'Bid was null'

            ], 200);

        }

        $currentBiggestBidUser = $this->findAuctionParticipant($request->auction\_id, $auction->highest\_bid\_user\_id);

        if ($request->amount > $currentBiggestBidUser->amount) {

            $auction->highest\_bid\_user\_id = $request->user\_id;

            $auction->save();

            return response()->json([

                'status' => 'success',

                'msg' => 'Bid replaced'

            ], 200);

        }

        return response()->json([

            'status' => 'success',

            'msg' => 'No changes to the bid'

        ], 200);

    }

    /\*\*

     \* Show the form for creating a new resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function create($auctionId, $startBid, $endDate)

    {

        $commercialAuction = new CommercialAuction();

        $commercialAuction->auction\_id = $auctionId;

        $commercialAuction->start\_bid = $startBid;

        $commercialAuction->end\_date = $endDate;

        $commercialAuction->save();

    }

}

**OfferCategoryController.php**

<?php

namespace App\Http\Controllers;

use App\OfferCategory;

use App\Http\Resources\Offers\OfferCategory as OfferCategoryResources;

use Illuminate\Http\Request;

class OfferCategoryController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function index()

    {

        return OfferCategoryResources::collection(OfferCategory::all());

    }

}

**OfferController.php**

<?php

namespace App\Http\Controllers;

use DB;

use App\User;

use App\Offer;

use App\Http\Resources\Offers\Offer as OfferResources;

use App\Traits\OffersTraits;

use Illuminate\Http\Request;

use Illuminate\Support\Collection;

use Illuminate\Support\Facades\Storage;

use Illuminate\Support\Facades\Validator;

use JWTAuth;

class OfferController extends Controller

{

    use OffersTraits;

    /\*\*

     \* Display a listing of the resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function index(Request $request)

    {

        $offersQuery = Offer::query();

        // offer sorting by creation date

        if (!is\_null($request->date)) {

            switch ($request->date) {

                case 'accending':

                    $offersQuery->orderBy('created\_at','asc');

                    break;

                case 'descending':

                    $offersQuery->orderBy('created\_at','desc');

                    break;

            }

        } else {

            $offersQuery->orderBy('created\_at','desc');

        }

        // conditions from route query parameters

        if(!is\_null($request->title)) {

            $offersQuery->where('title', 'like', '%'.$request->title.'%');

        }

        if(!is\_null($request->description)) {

            $offersQuery->where('body', 'like', '%'.$request->description.'%');

        }

        $offers = Offer::where('status', 'active');

        if(!is\_null($request->parts)) {

            $offerIds = DB::table('offers\_parts')

                ->where('part', 'like', '%'.$request->parts.'%')

                ->groupBy('offer\_id')

                ->pluck('offer\_id')

                ->toArray();

            $offersQuery->whereIn('id', $offerIds);

        }

        $filterOfferIds = [];

        if(!is\_null($request->category)) {

            $offerIds = DB::table('offers\_parts')

                ->whereIn('category', (array) $request->category)

                ->groupBy('offer\_id')

                ->pluck('offer\_id');

            $offersQuery->whereIn('id', $offerIds);

        }

        $offersQuery->where('status', 'active');

        $offers = $offersQuery;

        $offers = $offers->paginate(5);

        foreach ($offers as $offer) {

            $offer->author\_info = $this->getOfferAuthor($offer->author\_id);

        }

        return OfferResources::collection($offers)->additional([

            'meta' => [

                'version' => '1.0.0',

                'API\_base\_url' => url('/')

            ]

        ]);

    }

    public function getAll () {

        $offers = Offer::where('status', 'active')->get();

        foreach ($offers as $offer) {

            $offer->author\_info = $this->getOfferAuthor($offer->author\_id);

        }

        return OfferResources::collection($offers);

    }

    public function getServiceOffers () {

        $offers = Offer::all();

        foreach ($offers as $offer) {

            $offer->author\_info = $this->getOfferAuthor($offer->author\_id);

        }

        return OfferResources::collection($offers);

    }

    public function getFavoriteOffers (Request $request) {

        return DB::table('favorite\_offers')->where('user\_id', $request->user\_id)->get();

    }

    public function changeFavorite (Request $request) {

        if ($request->action == 'set') {

            DB::table('favorite\_offers')->insert(

                ['user\_id' => $request->user\_id, 'offer\_id' => $request->offer\_id]

            );

        }

        if ($request->action == 'delete') {

            DB::table('favorite\_offers')

            ->where('user\_id', $request->user\_id)

            ->where('offer\_id', $request->offer\_id)

            ->delete();

        }

    }

    public function getUserOffers (Request $request) {

        $offers = Offer::where('author\_id', '=', $request->author\_id)->where('status', 'active')->get();

        if (sizeof($offers) < 1) {

            return [];

        }

        if (sizeof($offers) > 1) {

            foreach ($offers as $offer) {

                $offer->author\_info = $this->getOfferAuthor($request->author\_id);

            }

        } else {

            $offers[0]->author\_info = $this->getOfferAuthor($request->author\_id);

        }

        try {

            return OfferResources::collection($offers);

        } catch (\Throwable $th) {

            return new OfferResources($offers);

        }

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return \Illuminate\Http\Response

     \*/

    public function store(Request $request)

    {

        $offer = new Offer;

        $offer->author\_id = $request->user\_id;

        $offer->title = $request->title;

        $offer->body = $request->body;

        $offer->contact\_phone = $request->contact\_phone;

        $offer->save();

        $this->storeOfferParts($request->parts, $offer->id);

        $previewImage = self::storeImages($offer->id, $request->images, $request->preview\_image\_id);

        $offer->preview\_image = $previewImage;

        $offer->save();

        return response()->json(['status' => 'success'], 200);

    }

    public function storeImages($offerId, $filesObj, $preview\_image\_id)

    {

        $previewImage = null;

        foreach ($filesObj as $fileId => $file) {

            $newLabel = null;

            $name\_passed = false;

            do {

                $newLabel = md5(time()+rand()).'.'.$file->getClientOriginalExtension();

                if (!DB::table('offers\_media')->where('file\_name', $newLabel)) {

                    $name\_passed = true;

                }

            } while ($name\_passed = false);

            $path = $file->storeAs(

                'public/uploads/offers\_media', $newLabel

            );

            DB::table('offers\_media')->insert(

                [

                    'offer\_id' => $offerId,

                    'photo\_path' => Storage::disk('public')->url('uploads/offers\_media/'.$newLabel),

                    'file\_name' => $newLabel

                ]

            );

            if ($fileId == $preview\_image\_id) {

                $previewImage = Storage::disk('public')->url('uploads/offers\_media/'.$newLabel);

            }

        }

        return $previewImage ? $previewImage : Storage::disk('public')->url('uploads/offers\_media/'.$newLabel);

    }

    /\*\*

     \* Display the specified resource.

     \*

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function show($id)

    {

        $offer = Offer::find($id);

        $offer['parts'] = $this->showOfferParts($id);

        return new OfferResources($offer);

    }

    public function media($id)

    {

        $mediaCollection = collect([]);

        $rawMedia = DB::table('offers\_media')->get()->where('offer\_id', $id);

        foreach ($rawMedia as $data) {

            $mediaCollection = $mediaCollection->concat(['file\_name' => Storage::disk('public')->url('uploads/offers\_media/'.$data->file\_name)]);

        }

        return response()->json([

            'status' => 'success',

            'photo\_path' => $mediaCollection->toArray()

        ], 200);

    }

    public function softDelete(Request $request) {

        $offer = Offer::find($request->id);

        $offer->status = 'archived';

        $offer->save();

        return;

    }

}

**OfferPartController.php**

<?php

namespace App\Http\Controllers;

use Log;

use App\Offer;

use App\OfferPart;

use App\Http\Resources\Offers\OfferPart as OfferPartResources;

use Illuminate\Http\Request;

use Illuminate\Support\Collection;

class OfferPartController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function index()

    {

        return OfferPartResources::collection(OfferPart::all());

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return \Illuminate\Http\Response

     \*/

    public static function store($arrayData, $offerId)

    {

        $decodedArrayData = json\_decode($arrayData);

        $insertArray = array();

        foreach ($decodedArrayData as $dataChunk) {

            array\_push($insertArray, array(

                'part' => $dataChunk->label,

                'category' => $dataChunk->category,

                'offer\_id' => $offerId

            ));

        }

        OfferPart::insert($insertArray);

    }

    /\*\*

     \* Display the specified resource.

     \*

     \* @param  \App\OfferPart  $offerPart

     \* @return \Illuminate\Http\Response

     \*/

    public static function show($offerId)

    {

        return OfferPartResources::collection(OfferPart::where('offer\_id', $offerId)->get());

    }

}

**TopicCategoryController.php**

<?php

namespace App\Http\Controllers;

use App\TopicCategory;

use App\Http\Resources\Topics\TopicCategory as TopicCategoryResources;

use Illuminate\Http\Request;

class TopicCategoryController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function index()

    {

        return TopicCategoryResources::collection(TopicCategory::all());

    }

}

**TopicCommentController.php**

<?php

namespace App\Http\Controllers;

use App\TopicComment;

use App\Traits\ForumTraits;

use App\Http\Resources\Topics\TopicComment as TopicCommentResources;

use Illuminate\Http\Request;

class TopicCommentController extends Controller

{

    use ForumTraits;

    /\*\*

     \* Display a listing of the resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function index(Request $request)

    {

        if ($request->topic) {

            $comments = TopicComment::where('topic\_id', $request->topic)->orderByDesc('created\_at')->get();

        } else {

            $comments = TopicComment::all()->sortByDesc('created\_at');

        }

        foreach ($comments as $comment) {

            $comment->author\_data = $this->getAuthor($comment->author\_id);

        }

        return TopicCommentResources::collection($comments);

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return \Illuminate\Http\Response

     \*/

    public function store(Request $request)

    {

        $comment = new TopicComment;

        $comment->author\_id = $request->user\_id;

        $comment->topic\_id = $request->topic\_id;

        $comment->body = $request->body;

        $comment->save();

    }

    public function delete(Request $request) {

        $commentToDelete = TopicComment::find($request->comment);

        $commentToDelete->delete();

        return;

    }

}

**TopicController.php**

<?php

namespace App\Http\Controllers;

use App\Topic;

use App\Traits\ForumTraits;

use App\Http\Resources\Topics\Topic as TopicResources;

use Illuminate\Http\Request;

class TopicController extends Controller

{

    use ForumTraits;

    /\*\*

     \* Display a listing of the resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function index(Request $request)

    {

        if ($request->category) {

            $topics = Topic::where('category\_id', $request->category)->get();

        } else {

            $topics = Topic::all();

        }

        foreach ($topics as $topic) {

            $topic->author\_data = $this->getAuthor($topic->author\_id);

        }

        return TopicResources::collection($topics);

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return \Illuminate\Http\Response

     \*/

    public function store(Request $request)

    {

        $topic = new Topic;

        $topic->author\_id = $request->user\_id;

        $topic->category\_id = $request->category\_id;

        $topic->title = $request->title;

        $topic->body = $request->body;

        $topic->save();

        return;

    }

    public function delete(Request $request) {

        $topicToDelete = Topic::find($request->topic);

        $topicToDelete->delete();

        return;

    }

}

**UserController.php**

<?php

namespace App\Http\Controllers;

use JWTAuth;

use App\User;

use App\Http\Controllers\Auth\LoginController;

use App\Http\Resources\Users\UserBasic as UserBasicResources;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Storage;

use Illuminate\Support\Facades\Validator;

class UserController extends Controller

{

    public function index() {

        $users = User::where('role', '=', 'user')->get();

        return response()->json([

            'status' => 'success',

            'users' => $users->toArray()

        ], 200);

    }

    public function show(Request $request, $id) {

        $user = User::find($id);

        return response()->json([

            'status' => 'success',

            'user' => $user->toArray()

        ], 200);

    }

    public function update(Request $request) {

        $user = Auth::user();

        $v = Validator::make($request->all(), [

            'username' => 'unique:users,username,' . $request->id,

            'email' => 'unique:users,email,' . $request->id

        ]);

        if ($v->fails()) {

            return response()->json([

                'status' => 'error',

                'errors' => $v->errors()

            ], 422);

        }

        $user->fill([

            'name' => $request->name,

            'surname' => $request->surname,

            'username' => $request->username,

            'email' => $request->email,

            'gender' => $request->gender

        ]);

        if ($request->password) {

            $user->fill([

                'password' => bcrypt($request->password)

            ]);

        }

        $user->save();

        return response()->json(['status' => 'success'], 200);

    }

    public function updateAvatar (Request $request) {

        $user = Auth::user();

        $newLabel = md5(time()+rand()).'.'.$request->avatar->getClientOriginalExtension();

        $request->avatar->storeAs(

            'public/uploads/users', $newLabel

        );

        $user->avatar\_path = Storage::disk('public')->url('uploads/users/'.$newLabel);

        $user->save();

        return response()->json(['status' => 'success'], 200);

    }

    public function delete (Request $request) {

        $user = Auth::user();

        $user->delete();

        return response()->json(['status' => 'success'], 200);

    }

    public static function findById ($userId) {

        return new UserBasicResources(User::find($userId));

    }

    public function findByIDs ($userIds) {

        return UserBasicResources::collection(User::whereIn('id', $userIds)->get());

    }

    public function setRole (Request $request) {

        $userToChange = User::find($request->user\_id);

        $userToChange->role = $request->role;

        $userToChange->save();

        return;

    }

    public function setStatus (Request $request) {

        $userToChange = User::find($request->user\_id);

        $userToChange->status = $request->status;

        $userToChange->save();

        return;

    }

    public function showAll () {

        $users = User::all();

        return response()->json([

            'status' => 'success',

            'users' => $users->toArray()

        ], 200);

    }

}

**Resources/Auction/Auction.php**

<?php

namespace App\Http\Resources\Auction;

use Illuminate\Http\Resources\Json\JsonResource;

class Auction extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array

     \*/

    public function toArray($request)

    {

        return [

            'id' => $this->id,

            'object\_id' => $this->object\_id,

            'started\_at' => $this->started\_at,

            'finished\_at' => $this->finished\_at,

            'status' => $this->status,

            'type' => $this->type

        ];

    }

}

**Resources/Auction/AuctionFullDataSet.php**

<?php

namespace App\Http\Resources\Auction;

use Illuminate\Http\Resources\Json\JsonResource;

class AuctionFullDataSet extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array

     \*/

    public function toArray($request)

    {

        return parent::toArray($request);

    }

}

**Resources/Auction/AuctionObject.php**

<?php

namespace App\Http\Resources\Auction;

use Illuminate\Http\Resources\Json\JsonResource;

class AuctionObject extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array

     \*/

    public function toArray($request)

    {

        return parent::toArray($request);

    }

}

**Resources/Auction/AuctionParticipants.php**

<?php

namespace App\Http\Resources\Auction;

use Illuminate\Http\Resources\Json\JsonResource;

class AuctionParticipants extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array

     \*/

    public function toArray($request)

    {

        return [

            'id' => $this->user\_id,

            'username' => $this->username,

            'email' => $this->email,

            'avatar\_path' => $this->avatar\_path,

            'amount' => $this->amount,

        ];

    }

}

**Resources/Auction/CharityAuction.php**

<?php

namespace App\Http\Resources\Auction;

use Illuminate\Http\Resources\Json\JsonResource;

class CharityAuction extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array

     \*/

    public function toArray($request)

    {

        return [

            'id' => $this->id,

            'goal' => $this->goal

        ];

    }

}

**Resources/Auction/CommercialAuction.php**

<?php

namespace App\Http\Resources\Auction;

use Illuminate\Http\Resources\Json\JsonResource;

class CommercialAuction extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array

     \*/

    public function toArray($request)

    {

        return [

            'id' => $this->id,

            'start\_bid' => $this->start\_bid,

            'highest\_bid\_user\_id' => $this->highest\_bid\_user\_id,

            'end\_date' => $this->end\_date,

        ];

    }

}

**Resources/Offers/Offer.php**

<?php

namespace App\Http\Resources\Offers;

use Illuminate\Http\Resources\Json\JsonResource;

class Offer extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array

     \*/

    public function toArray($request)

    {

        $arrayData = [

            'id' => $this->id,

            'title' => $this->title,

            'body' => $this->body,

            'contact\_phone' => $this->contact\_phone,

            'contact\_email' => $this->contact\_email,

            'created\_at' => $this->created\_at,

            'updated\_at' => $this->updated\_at,

            'preview\_image' => $this->preview\_image,

            'status' => $this->status

        ];

        if ($this->author\_info) {

            $arrayData['author\_info'] = $this->author\_info;

        }

        if ($this->author\_id) {

            $arrayData['author\_id'] = $this->author\_id;

        }

        if ($this->parts) {

            $arrayData['parts'] = $this->parts;

        }

        return $arrayData;

    }

}

**Resources/Offers/OfferCategory.php**

<?php

namespace App\Http\Resources\Offers;

use Illuminate\Http\Resources\Json\JsonResource;

class OfferCategory extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array

     \*/

    public function toArray($request)

    {

        return [

            'id' => $this->id,

            'label' => $this->label

        ];

    }

}

**Resources/Offers/OfferPart.php**

<?php

namespace App\Http\Resources\Offers;

use Illuminate\Http\Resources\Json\JsonResource;

class OfferPart extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array

     \*/

    public function toArray($request)

    {

        return [

            'id' => $this->id,

            'part' => $this->part,

            'category' => $this->category

        ];

    }

}

**Resources/Topics/Topic.php**

<?php

namespace App\Http\Resources\Topics;

use Illuminate\Http\Resources\Json\JsonResource;

class Topic extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array

     \*/

    public function toArray($request)

    {

        return parent::toArray($request);

    }

}

**Resources/Topics/TopicCategory.php**

<?php

namespace App\Http\Resources\Topics;

use Illuminate\Http\Resources\Json\JsonResource;

class TopicCategory extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array

     \*/

    public function toArray($request)

    {

        return [

            'id' => $this->id,

            'label' => $this->label

        ];

    }

}

**Resources/Topics/TopicComment.php**

<?php

namespace App\Http\Resources\Topics;

use Illuminate\Http\Resources\Json\JsonResource;

class TopicComment extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array

     \*/

    public function toArray($request)

    {

        return parent::toArray($request);

    }

}

**Resources/User/UserBasic.php**

<?php

namespace App\Http\Resources\Users;

use Illuminate\Http\Resources\Json\JsonResource;

class UserBasic extends JsonResource

{

    /\*\*

     \* Transform the resource into an array.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return array

     \*/

    public function toArray($request)

    {

        return [

            'id' => $this->id,

            'username' => $this->username,

            'email' => $this->email,

            'avatar\_path' => $this->avatar\_path

        ];

    }

}

**Traits/AuctionTraits.php**

<?php

namespace App\Traits;

use App\Http\Controllers\AuctionObjectController;

use App\Http\Controllers\AuctionParticipantsController;

use App\Http\Controllers\CharityAuctionController;

use App\Http\Controllers\CommercialAuctionController;

use App\Http\Controllers\UserController;

use App\Http\Resources\Auction\AuctionParticipants as AuctionParticipantsResources;

use App\Http\Resources\Auction\AuctionObject as AuctionObjectResources;

use Illuminate\Http\Request;

trait AuctionTraits

{

    public function indexActive() {

        return [

            'charity' => (new CharityAuctionController)->getActive(),

            'commercial' => (new CommercialAuctionController)->getActive()

        ];

    }

    public function getAuctionObjects($auctionsObject\_Model, $objectIds) {

        return AuctionObjectResources::collection($auctionsObject\_Model->whereIn('id', $objectIds));

    }

    public function findAuctionParticipant($auctionId, $userId) {

        return (new AuctionParticipantsController)->show($auctionId, $userId);

    }

    public function getAuctionParticipants($auctionParticipants\_Model, $auctionId) {

        $participantsInfo = $auctionParticipants\_Model->where('auction\_id', $auctionId);

        $userIds = [];

        foreach ($participantsInfo as $participant\_data\_chunk) {

            $userIds[] = $participant\_data\_chunk->user\_id;

        }

        $usersInfo = (new UserController)->findByIDs($userIds);

        try {

            foreach ($participantsInfo as $participant){

                foreach ($usersInfo as $userChunk) {

                    if (json\_encode($userChunk->id) == $participant->user\_id) {

                        $participant->username = $userChunk->username;

                        $participant->email = $userChunk->email;

                        $participant->avatar\_path = $userChunk->avatar\_path;

                    }

                }

            }

        } catch (\Throwable $err) {

            $participantsInfo->username = $usersInfo[0]->username;

            $participantsInfo->email = $usersInfo[0]->email;

            $participantsInfo->avatar\_path = $usersInfo[0]->avatar\_path;

        }

        return AuctionParticipantsResources::collection($participantsInfo);

    }

    public function getAuctionTypeData ($auctionId, $auctionType) {

        switch ($auctionType) {

            case 'charity':

                return (new CharityAuctionController)->getDetailsByAuctionId($auctionId);

            case 'commercial':

                return (new CommercialAuctionController)->getDetailsByAuctionId($auctionId);

        }

    }

    public function createAuctionObject ($objectName, $previewImage) {

        return (new AuctionObjectController)->create($objectName, $previewImage);

    }

    public function createCharityAuction ($auctionId, $goal) {

        (new CharityAuctionController)->create($auctionId, $goal);

    }

    public function createCommercialAuction ($auctionId, $startBid, $endDate) {

        (new CommercialAuctionController)->create($auctionId, $startBid, $endDate);

    }

}

**Traits/ForumTraits.php**

<?php

namespace App\Traits;

use App\Http\Controllers\UserController;

trait ForumTraits

{

    public function getAuthor($userId) {

        return UserController::findById($userId);

    }

}

**Traits/OffersTraits.php**

<?php

namespace App\Traits;

use App\Http\Controllers\OfferPartController;

use App\Http\Controllers\UserController;

/\*\*

 \*

 \*/

trait OffersTraits

{

    // Offer Parts

    public function showOfferParts($offerId) {

        return OfferPartController::show($offerId);

    }

    public function storeOfferParts($arrayData, $offerId) {

        return OfferPartController::store($arrayData, $offerId);

    }

    public function getOfferAuthor($userId) {

        return UserController::findById($userId);

    }

}

**Auction.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Auction extends Model

{

    /\*\*

     \* Indicates if the model should be timestamped.

     \*

     \* @var bool

     \*/

    public $timestamps = false;

}

**AuctionObject.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class AuctionObject extends Model

{

    /\*\*

    \* The table associated with the model.

    \*

    \* @var string

    \*/

   protected $table = 'auction\_object';

   /\*\*

    \* Indicates if the model should be timestamped.

    \*

    \* @var bool

    \*/

   public $timestamps = false;

}

**AuctionParticiapnts.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class AuctionParticipants extends Model

{

    /\*\*

    \* The table associated with the model.

    \*

    \* @var string

    \*/

   protected $table = 'auction\_participants';

   /\*\*

    \* Indicates if the model should be timestamped.

    \*

    \* @var bool

    \*/

   public $timestamps = false;

}

**CharityAuction.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class CharityAuction extends Model

{

    /\*\*

    \* The table associated with the model.

    \*

    \* @var string

    \*/

   protected $table = 'charity\_auction';

   /\*\*

    \* Indicates if the model should be timestamped.

    \*

    \* @var bool

    \*/

   public $timestamps = false;

}

**CommercialAuction.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class CommercialAuction extends Model

{

    /\*\*

    \* The table associated with the model.

    \*

    \* @var string

    \*/

   protected $table = 'commercial\_auction';

   /\*\*

    \* Indicates if the model should be timestamped.

    \*

    \* @var bool

    \*/

   public $timestamps = false;

}

**Offer.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Offer extends Model

{

    protected $fillable = [

        'author\_id', 'title', 'body'

    ];

    public $timestamps = true;

}

**OfferCategory.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class OfferCategory extends Model

{

    /\*\*

     \* The table associated with the model.

     \*

     \* @var string

     \*/

    protected $table = 'parts\_category';

    /\*\*

     \* Indicates if the model should be timestamped.

     \*

     \* @var bool

     \*/

    public $timestamps = false;

}

**OfferPart.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class OfferPart extends Model

{

    /\*\*

     \* The table associated with the model.

     \*

     \* @var string

     \*/

    protected $table = 'offers\_parts';

    /\*\*

     \* Indicates if the model should be timestamped.

     \*

     \* @var bool

     \*/

    public $timestamps = false;

}

**Topic.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Topic extends Model

{

    /\*\*

     \* The table associated with the model.

     \*

     \* @var string

     \*/

    protected $table = 'topics';

}

**TopicCategory.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class TopicCategory extends Model

{

    /\*\*

     \* The table associated with the model.

     \*

     \* @var string

     \*/

    protected $table = 'topics\_category';

    /\*\*

     \* Indicates if the model should be timestamped.

     \*

     \* @var bool

     \*/

    public $timestamps = false;

}

**TopicComment.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class TopicComment extends Model

{

    /\*\*

     \* The table associated with the model.

     \*

     \* @var string

     \*/

    protected $table = 'topics\_comments';

}

**User.php**

<?php

namespace App;

use Illuminate\Contracts\Auth\MustVerifyEmail;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

use Tymon\JWTAuth\Contracts\JWTSubject;

class User extends Authenticatable implements JWTSubject

{

    use Notifiable;

    /\*\*

     \* The attributes that are mass assignable.

     \*

     \* @var array

     \*/

    protected $fillable = [

        'email', 'username', 'avatar\_path', 'name', 'surname', 'gender', 'password',

    ];

    /\*\*

     \* The attributes that should be hidden for arrays.

     \*

     \* @var array

     \*/

    protected $hidden = [

        'password', 'remember\_token',

    ];

    /\*\*

     \* The attributes that should be cast to native types.

     \*

     \* @var array

     \*/

    protected $casts = [

        'email\_verified\_at' => 'datetime',

    ];

    public $timestamps = false;

    public function getJWTIdentifier() {

        return $this->getKey();

    }

    public function getJWTCustomClaims() {

        return [];

    }

}

**2014\_10\_12\_000000\_create\_users\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('users', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->string('email')->unique();

            $table->string('username', 16)->unique();

            $table->text('avatar\_path')->nullable();

            $table->string('name', 24)->nullable();

            $table->string('surname', 24)->nullable();

            $table->enum('gender', ['male', 'female']);

            $table->string('password');

            $table->timestamps();

            $table->enum('role', ['super\_admin', 'admin', 'user'])->default('user');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('users');

    }

}

**2019\_10\_13\_121742\_create\_offers\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateOffersTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('offers', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->unsignedBigInteger('author\_id');

            $table->string('title', 64);

            $table->text('body');

            $table->string('contact\_phone', 12)->nullable();

            $table->timestamps();

            $table->enum('status', ['active', 'archived']);

            $table->text('preview\_image')->nullable();

            $table->foreign('author\_id')->references('id')->on('users')->onDelete('cascade')->onUpdate('cascade');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('offers');

    }

}

**2020\_04\_28\_134009\_create\_object\_type\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateObjectTypeTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('object\_type', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->string('label', 32);

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('object\_type');

    }

}

**2020\_04\_28\_134853\_create\_parts\_category\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreatePartsCategoryTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('parts\_category', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->string('label', 32);

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('parts\_category');

    }

}

**2020\_04\_29\_131543\_create\_auction\_object\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateAuctionObjectTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('auction\_object', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->unsignedBigInteger('object\_type\_id');

            $table->string('name', 24);

            $table->text('preview\_image')->nullable();

            $table->foreign('object\_type\_id')->references('id')->on('object\_type');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('auction\_object');

    }

}

**2020\_04\_29\_131544\_create\_auctions\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateAuctionsTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('auctions', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->unsignedBigInteger('object\_id');

            $table->timestamp('started\_at')->useCurrent();

            $table->timestamp('finished\_at')->nullable();

            $table->enum('status', ['active', 'dismissed'])->default('active');

            $table->enum('type', ['charity', 'commercial']);

            $table->unique(['id', 'object\_id']);

            $table->foreign('object\_id')->references('id')->on('auction\_object');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('auctions');

    }

}

**2020\_04\_29\_133255\_create\_charity\_auction\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateCharityAuctionTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('charity\_auction', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->unsignedBigInteger('auction\_id');

            $table->double('goal', 8, 2)->default(0);

            $table->foreign('auction\_id')->references('id')->on('auctions')->onDelete('cascade')->onUpdate('cascade');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('charity\_auction');

    }

}

**2020\_04\_29\_133805\_create\_commercial\_auction\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateCommercialAuctionTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('commercial\_auction', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->unsignedBigInteger('auction\_id');

            $table->double('start\_bid', 8, 2)->default(0);

            $table->unsignedBigInteger('highest\_bid\_user\_id')->nullable();

            $table->dateTime('end\_date');

            $table->foreign('auction\_id')->references('id')->on('auctions')->onDelete('cascade')->onUpdate('cascade');

            $table->foreign('highest\_bid\_user\_id')->references('id')->on('users');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('commercial\_auction');

    }

}

**2020\_04\_29\_134043\_create\_auction\_participants\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateAuctionParticipantsTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('auction\_participants', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->unsignedBigInteger('user\_id');

            $table->unsignedBigInteger('auction\_id');

            $table->double('amount', 8, 2);

            $table->unique(['user\_id', 'auction\_id']);

            $table->foreign('user\_id')->references('id')->on('users')->onDelete('cascade')->onUpdate('cascade');

            $table->foreign('auction\_id')->references('id')->on('auctions')->onDelete('cascade')->onUpdate('cascade');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('auction\_participants');

    }

}

**2020\_04\_29\_134512\_create\_offers\_media\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateOffersMediaTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('offers\_media', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->unsignedBigInteger('offer\_id');

            $table->text('photo\_path');

            $table->text('file\_name');

            $table->foreign('offer\_id')->references('id')->on('offers')->onDelete('cascade')->onUpdate('cascade');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('offers\_media');

    }

}

**2020\_04\_29\_134630\_create\_offers\_parts\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateOffersPartsTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('offers\_parts', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->string('part', 32);

            $table->unsignedBigInteger('category');

            $table->unsignedBigInteger('offer\_id');

            $table->foreign('category')->references('id')->on('parts\_category')->onUpdate('cascade');

            $table->foreign('offer\_id')->references('id')->on('offers')->onDelete('cascade')->onUpdate('cascade');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('offers\_parts');

    }

}

**2020\_05\_11\_183450\_create\_auction\_trigger.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateAuctionTrigger extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        DB::unprepared('

        CREATE TRIGGER `change\_status` BEFORE UPDATE ON `auctions`

            FOR EACH ROW BEGIN

                IF NEW.finished\_at IS NOT NULL THEN

                    SET NEW.status = "dismissed";

                ELSE

                    SET NEW.status = "active";

                END IF;

            END

        ');

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        DB::unprepared('DROP TRIGGER `change\_status`');

    }

}

**2020\_05\_20\_145510\_create\_favorite\_offers\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateFavoriteOffersTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('favorite\_offers', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->unsignedBigInteger('user\_id');

            $table->unsignedBigInteger('offer\_id');

            $table->unique(['user\_id', 'offer\_id']);

            $table->foreign('user\_id')->references('id')->on('users')->onDelete('cascade')->onUpdate('cascade');

            $table->foreign('offer\_id')->references('id')->on('offers')->onDelete('cascade')->onUpdate('cascade');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('favorite\_offers');

    }

}

**2020\_05\_25\_075821\_add\_status\_field\_to\_users.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class AddStatusFieldToUsers extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::table('users', function (Blueprint $table) {

            $table->enum('status', ['active', 'suspended'])->default('active');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::table('users', function (Blueprint $table) {

            $table->dropColumn('status');

        });

    }

}

**2020\_05\_26\_095552\_create\_topics\_category\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateTopicsCategoryTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('topics\_category', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->string('label', 32);

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('topics\_category');

    }

}

**2020\_05\_26\_095632\_create\_topics\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateTopicsTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('topics', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->unsignedBigInteger('author\_id');

            $table->unsignedBigInteger('category\_id');

            $table->string('title', 64);

            $table->text('body');

            $table->timestamps();

            $table->foreign('author\_id')->references('id')->on('users')->onDelete('cascade')->onUpdate('cascade');

            $table->foreign('category\_id')->references('id')->on('topics\_category')->onDelete('cascade')->onUpdate('cascade');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('topics');

    }

}

**2020\_05\_26\_095844\_create\_topics\_comments\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateTopicsCommentsTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('topics\_comments', function (Blueprint $table) {

            $table->bigIncrements('id');

            $table->unsignedBigInteger('author\_id');

            $table->unsignedBigInteger('topic\_id');

            $table->text('body');

            $table->timestamps();

            $table->foreign('author\_id')->references('id')->on('users')->onDelete('cascade')->onUpdate('cascade');

            $table->foreign('topic\_id')->references('id')->on('topics')->onDelete('cascade')->onUpdate('cascade');

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('topics\_comments');

    }

}

**frontend/components/admin/AddAuction.vue**

<template>

  <div>

    <v-form

      v-if="!response"

      ref="aucForm"

    >

      <v-card>

        <v-card-title>

          Add {{ pType }} auction

        </v-card-title>

        <v-card-text>

          <v-container>

            <v-row>

              <v-col>

                <v-text-field

                  v-model="objectName"

                  label="Auction object name"

                  placeholder="Auction object name"

                  :counter="24"

                  :rules="[rules.required]"

                />

              </v-col>

              <v-col>

                <v-file-input

                  v-model="image"

                  label="Auction Image"

                  prepend-icon="mdi-camera"

                  :rules="[rules.required]"

                />

              </v-col>

            </v-row>

            <!-- /charity block -->

            <div

              v-if="pType === 'charity'"

            >

              <v-row>

                <v-col>

                  <v-text-field

                    v-model="amount"

                    label="Goal (EUR)"

                    :rules="[rules.required, scoped\_rules.numbersOnly, scoped\_rules.amount]"

                  />

                </v-col>

              </v-row>

            </div>

            <!-- /charity block -->

            <!-- commercial block -->

            <div

              v-if="pType === 'commercial'"

            >

              <v-row>

                <v-col>

                  <v-text-field

                    v-model="amount"

                    label="Starting bid (EUR)"

                    :rules="[rules.required, scoped\_rules.numbersOnly, scoped\_rules.amount]"

                  />

                </v-col>

                <v-col>

                  <v-menu

                    :rules="[rules.required]"

                    v-model="dateMenu"

                    offset-y

                  >

                    <template #activator="{ on }">

                      <v-text-field

                        v-model="date"

                        label="End date"

                        prepend-icon="mdi-event"

                        readonly

                        v-on="on"

                      />

                    </template>

                    <v-date-picker

                      v-model="date"

                      :min="new Date(new Date().getTime()+(1\*24\*60\*60\*1000)).toISOString().substr(0, 10)"

                    />

                  </v-menu>

                  <span

                    v-if="!date"

                  >

                    Specify participation ending date

                  </span>

                </v-col>

              </v-row>

            </div>

            <!-- /commercial block -->

          </v-container>

        </v-card-text>

        <v-card-actions>

          <v-btn

            text

            color="primary"

            width="50%"

            @click="submit()"

          >

            Submit

          </v-btn>

          <v-btn

            text

            width="50%"

            @click="closeDialog()"

          >

            Cancel

          </v-btn>

        </v-card-actions>

      </v-card>

    </v-form>

    <v-card

      v-if="response"

    >

      <v-card-title>

        Attention!

      </v-card-title>

      <v-card-text>

        {{ response }}

      </v-card-text>

      <v-card-actions>

        <v-btn

          text

          block

          @click="closeDialog()"

        >

          Ok

        </v-btn>

      </v-card-actions>

    </v-card>

  </div>

</template>

<script>

import axios from 'axios'

import rules from '../../plugins/rules/rules'

export default {

  props: {

    pType: {type: String, default: null}

  },

  data () {

    return {

      objectName: null,

      amount: null,

      date: null,

      dateMenu: null,

      image: null,

      response: null,

      rules: rules,

      scoped\_rules: {

        numbersOnly: v => !!v && /^[0-9]+\.?[0-9]+$/.test(v) || 'Incorrect amount',

        amount: v => this.amount > 0 || 'Incorrect amount'

      }

    }

  },

  beforeDestroy () {

    this.closeDialog()

  },

  methods: {

    closeDialog () {

      this.destroyDialog()

      this.$emit('closeDialog')

    },

    destroyDialog () {

      this.objectName = null

      this.amount = null

      this.date = null

      this.dateMenu = null

      this.image = null

      this.response = null

      try {

        this.$refs.aucForm.reset()

      } catch (error) {

        return

      }

    },

    submit () {

      if (this.$refs.aucForm.validate() && (this.pType === 'commercial' ? this.date : true)) {

        const newAuctionData = new FormData()

        newAuctionData.append('type', this.pType)

        newAuctionData.append('object\_name', this.objectName)

        newAuctionData.append('preview\_image', this.image, this.image.name)

        switch (this.pType) {

          case 'charity':

            newAuctionData.append('goal', this.amount)

            break

          case 'commercial':

            newAuctionData.append('start\_bid', this.amount)

            newAuctionData.append('end\_date', this.date)

            break

        }

        const config = {

          headers: {

            'Authorization': 'Bearer '+this.$auth.token(),

            'Content-Type': 'multipart/form-data'

          }

        }

        axios

          .post('http://127.0.0.1:8000/api/auth/auction/add', newAuctionData, config)

          .then (res => {

            this.destroyDialog()

            this.response = `Auction successfully created!`

          })

          .catch ((err) => {

            this.response = `Something went wrong!`

            console.log(err)

          })

      }

    }

  }

}

</script>

**frontend/components/admin/Admins.vue**

<template>

  <v-card>

    <v-card-title>

      Privilege Control

    </v-card-title>

    <v-card-subtitle>

      Administrators

    </v-card-subtitle>

    <v-card-text>

      <UserList

        :pUsersArray="admins"

        :pIsSuperAdmin="true"

        @update="fetchAllUsers()"

      />

    </v-card-text>

    <v-card-subtitle>

      Active Users

    </v-card-subtitle>

    <v-card-text>

      <UserList

        :pUsersArray="activeUsers"

        :pIsSuperAdmin="true"

        @update="fetchAllUsers()"

      />

    </v-card-text>

  </v-card>

</template>

<script>

import axios from 'axios'

export default {

  components: {

    UserList: () => import('./UserList')

  },

  data () {

    return {

      activeUsers: [],

      admins: [],

      sAdmins: []

    }

  },

  created () {

    this.fetchAllUsers()

  },

  methods: {

    fetchAllUsers () {

      axios

        .get('http://127.0.0.1:8000/api/auth/users/service')

        .then(res => {

          let incomingUsers

          if (res.data.users.length) {

            incomingUsers = res.data.users

          } else {

            incomingUsers = [res.data.users]

          }

          this.activeUsers = []

          this.admins= []

          this.sAdmins= []

          for (const user of incomingUsers) {

            user.dialog = false

            if (user.status === 'active') {

              switch (user.role) {

                case 'user':

                  this.activeUsers.push(user)

                  break

                case 'admin':

                  this.admins.push(user)

                  break

                case 'super\_admin':

                  this.sAdmins.push(user)

                  break

              }

            }

          }

        })

        .catch(err => {

          console.log(err)

        })

    }

  }

}

</script>

**frontend/components/admin/AuctionList.vue**

<template>

  <div>

    <v-list

      flat

    >

      <v-list-item-group

        v-if="pAuctions"

      >

        <v-list-item

          v-for="auction in pAuctions"

          :key="auction.id"

          @click="auction.showDialog = true"

        >

          <v-list-item-content>

            <v-list-item-title

              class="item-title"

            >

              <div>

                {{ auction.auction\_object.name }}

              </div>

              <div

                class="finished-caption"

                v-if="isFinished() && activeAuction"

              >

                <h3>Finished</h3>

              </div>

              <v-spacer />

              <v-dialog

                v-model="auction.showDialog"

              >

                <template #activator="{ on }">

                  <v-btn

                    v-on="on"

                    small

                    text

                    @click="auction.showDialog = true"

                  >

                    EXPAND

                  </v-btn>

                </template>

                <div

                  v-if="!showFinishDialog"

                  style="background-color: white;"

                >

                  <AuctionDisplay

                    :pAuction="auction"

                    :pType="auction.type"

                  />

                  <div

                    v-if="isFinished()"

                  >

                    <v-btn

                      text

                      block

                      color="primary"

                      @click="finishAuction(auction)"

                    >

                      Finish auction

                    </v-btn>

                  </div>

                </div>

                <v-card

                  v-if="showFinishDialog"

                >

                  <v-card-title>

                    Auction finished!

                  </v-card-title>

                  <v-card-actions>

                    <v-btn

                      text

                      @click="closeFisnishDialog()"

                    >

                      Ok

                    </v-btn>

                  </v-card-actions>

                </v-card>

              </v-dialog>

            </v-list-item-title>

          </v-list-item-content>

        </v-list-item>

      </v-list-item-group>

    </v-list>

  </div>

</template>

<script>

import axios from 'axios'

export default {

  components: {

    AuctionDisplay: () => import('../AuctionDisplay')

  },

  props: {

    pAuctions: {type: Array, default: null},

    pIsActive: {type: Boolean, default: false}

  },

  data () {

    return {

      activeAuction: null,

      showFinishDialog: false

    }

  },

  created () {

    if (this.pIsActive) {

      this.activeAuction = this.pAuctions[0]

    }

  },

  methods: {

    closeFisnishDialog() {

      this.$emit('updateAuctions')

    },

    finishAuction (auction) {

      const config = {

        headers: {

          'Authorization': 'Bearer '+this.$auth.token(),

          'Content-Type': 'multipart/form-data'

        }

      }

      const finishingAuctionData = new FormData()

      finishingAuctionData.append('auction\_id', auction.id)

      axios

        .post('http://127.0.0.1:8000/api/auth/auction/finish', finishingAuctionData, config)

          .then (res => {

            this.showFinishDialog = true

          })

          .catch ((err) => {

            console.log(err)

          })

    },

    isFinished () {

      if (this.activeAuction) {

        switch (this.activeAuction.type) {

          case 'charity':

            let sum = 0

            this.activeAuction.participants.map(participant => sum += participant.amount)

            if (sum >= this.activeAuction.auction\_data[0].goal) {

              return true

            }

            return false

          case 'commercial':

            if (new Date().getTime() > new Date(this.activeAuction.auction\_data[0].end\_date).getTime()) {

              return true

            }

            return false

        }

      }

      return false

    }

  }

}

</script>

<style scoped>

  .finished-caption {

    margin-left: 10px;

  }

  .item-title {

    display: flex;

  }

</style>

**frontend/components/admin/Auctions.vue**

<template>

  <v-card

    outlined

  >

    <v-card-title>Active Auctions</v-card-title>

    <v-divider />

    <v-card-subtitle>

      <h3>Charity</h3>

    </v-card-subtitle>

    <AuctionList

      v-if="activeCharityAuction"

      :pAuctions="[activeCharityAuction]"

      :pIsActive="true"

      @updateAuctions="closeDialog()"

    />

    <v-card-actions

      v-else

    >

      <v-dialog

        v-model="showCharityCreateDialog"

      >

        <template

          #activator="{ on }"

        >

          <v-btn

            depressed

            block

            color="primary"

            v-on="on"

          >

            Add Auction

          </v-btn>

        </template>

        <AddAuction

          :pType="'charity'"

          @closeDialog="showCharityCreateDialog = false"

        />

      </v-dialog>

    </v-card-actions>

    <v-card-subtitle>

      <h3>Commercial</h3>

    </v-card-subtitle>

    <AuctionList

      v-if="activeCommercialAuction"

      :pAuctions="[activeCommercialAuction]"

      :pIsActive="true"

      @updateAuctions="closeDialog()"

    />

    <v-card-actions

      v-else

    >

      <v-dialog

        v-model="showCommericalCreateDialog"

      >

        <template

          #activator="{ on }"

        >

          <v-btn

            depressed

            block

            color="primary"

            v-on="on"

          >

            Add Auction

          </v-btn>

        </template>

        <AddAuction

          :pType="'commercial'"

          @closeDialog="closeDialog()"

        />

      </v-dialog>

    </v-card-actions>

    <v-divider />

    <v-card-title>Dismissed Auctions</v-card-title>

    <v-divider />

    <v-card-subtitle>

      <h3>Charity</h3>

    </v-card-subtitle>

    <AuctionList

      :pAuctions="charityAuctions"

    />

    <v-card-subtitle>

      <h3>Commercial</h3>

    </v-card-subtitle>

    <AuctionList

      :pAuctions="commercialAuctions"

    />

  </v-card>

</template>

<script>

import fetchAuctions from '../../plugins/fetchAuctions'

export default {

  components: {

    AddAuction: () => import('./AddAuction'),

    AuctionList: () => import('./AuctionList')

  },

  data () {

    return {

      activeCharityAuction: null,

      activeCommercialAuction: null,

      charityAuctions: [],

      commercialAuctions: [],

      showCharityCreateDialog: false,

      showCommericalCreateDialog: false

    }

  },

  created () {

    this.fetchActiveAuctions()

    this.fetchDismissedAuctions()

  },

  methods: {

    closeDialog() {

      this.showCharityCreateDialog = false

      this.showCommericalCreateDialog = false

      this.fetchActiveAuctions()

      this.fetchDismissedAuctions()

    },

    fetchActiveAuctions() {

      this.activeCharityAuction = null

      this.activeCommercialAuction = null

      fetchAuctions('http://127.0.0.1:8000/api/auctions')

        .then(res => {

          this.activeCharityAuction = res.charityAuctions

          this.activeCommercialAuction = res.commercialAuctions

        })

    },

    fetchDismissedAuctions() {

      fetchAuctions('http://127.0.0.1:8000/api/auctions/dismissed', 'dismissed')

      .then(res => {

        this.charityAuctions = res.charityAuctions

        this.commercialAuctions = res.commercialAuctions

        for (const auction of this.charityAuctions) {

          auction['showDialog'] = false

        }

        for (const auction of this.commercialAuctions) {

          auction['showDialog'] = false

        }

      })

    }

  }

}

</script>

<style scoped>

  .v-card\_\_subtitle {

    padding-top: 20px;

    padding-bottom: 0;

  }

</style>

**frontend/components/admin/Offers.vue**

<template>

  <v-card>

    <v-card-title>

      User Offers

      <v-spacer />

      <v-btn

        icon

        @click="fetchAll()"

      >

        <v-icon>mdi-autorenew</v-icon>

      </v-btn>

    </v-card-title>

    <v-card-subtitle>

      Active Offers

    </v-card-subtitle>

    <v-container

      class="offers-container"

    >

      <v-row

        v-for="offer in activeOffers"

        :key="offer.id"

      >

        <OffersDisplay

          class="offer"

          :pOffer="offer"

          :pCategories="categories"

          :pIsTiled="true"

          :pIsAdmin="true"

          @reloadOffers="fetchAll"

        />

      </v-row>

    </v-container>

    <v-card-subtitle>

      Archived Offers

    </v-card-subtitle>

    <v-container

      class="offers-container"

    >

      <v-row

        v-for="offer in archivedOffers"

        :key="offer.id"

      >

        <OffersDisplay

          class="offer"

          :pOffer="offer"

          :pCategories="categories"

          :pIsTiled="true"

          :pIsAdmin="true"

          @reloadOffers="fetchAll"

        />

      </v-row>

    </v-container>

  </v-card>

</template>

<script>

import axios from 'axios'

export default {

  components: {

    OffersDisplay: () => import('../OffersDisplay')

  },

  data () {

    return {

      activeOffers: [],

      archivedOffers: [],

      categories: []

    }

  },

  created () {

    this.fetchAll()

  },

  methods: {

    fetchAll () {

      this.fetchCategories()

      this.fetchAllOffers()

    },

    fetchCategories () {

      const path = 'http://127.0.0.1:8000/api/offer\_categories'

      axios

        .get(path)

        .then(res => {

          const {data:{data}} = res

          if (data && !data.length){

            this.categories = [data]

          } else {

            this.categories = data

          }

        })

        .catch(err => {

          this.errorCode = err.response.status

          this.responseError()

        })

    },

    fetchAllOffers () {

      this.activeOffers = []

      this.archivedOffers = []

      const config = {

        headers: {

          'Authorization': 'Bearer '+this.$auth.token(),

        }

      }

      axios

        .get('http://127.0.0.1:8000/api/auth/offers')

        .then(res => {

          const {data:{data}} = res

          data.map(offer => {

            switch (offer.status) {

              case 'active':

                this.activeOffers.push(offer)

                break

              case 'archived':

                this.archivedOffers.push(offer)

                break

            }

          })

        })

        .catch(err => {

          console.log(err);

          this.errorCode = err

          this.responseError()

        })

    },

  }

}

</script>

<style scoped>

  .offers-container {

    max-height: 500px;

    overflow: auto;

  }

  .offer {

    width: 100%;

  }

</style>

**frontend/components/admin/UserList.vue**

<template>

  <v-container

    class="users-container"

  >

    <v-row

      v-for="user in pUsersArray"

      :key="user.id"

    >

      <v-col>

        <span class="font-weight-bold">{{ user.email }}</span> ({{ user.username }})

      </v-col>

      <v-col>

        <v-dialog

          v-model="user.suspendDialog"

        >

          <template

            #activator="{ on }"

          >

            <v-btn

              text

              color="error"

              v-on="on"

            >

              <span

                v-if="pIsSuperAdmin"

              >

                {{ user.role === 'user' ? 'Promote' : 'Remove' }}

              </span>

              <span

                v-else

              >{{ isUserSuspended(user.status) ? 'Activate' : 'Suspend' }}</span>

            </v-btn>

          </template>

          <v-card>

            <v-card-title>

              Attention!

            </v-card-title>

            <v-card-text>

              Would you really like to

              <span

                v-if="pIsSuperAdmin"

              >

                {{ user.role === 'user' ? 'promote to admins' : 'remove from admins' }}

              </span>

              <span

                v-else

              >

                {{ isUserSuspended(user.status) ? 'activate' : 'suspend' }}

              </span>

              user

              <span class="font-weight-bold">{{ user.email }}</span> ({{ user.username }}) ?

            </v-card-text>

            <v-card-actions>

              <v-btn

                text

                width="50%"

                color="success"

                @click="pIsSuperAdmin ? changeUserPrivilege(user, (user.role === 'user' ? 'admin' : 'user')) : changeUserStatus(user, (isUserSuspended(user.status) ? 'active' : 'suspended'))"

              >

                Confirm

              </v-btn>

              <v-btn

                text

                width="50%"

                color="error"

                @click="user.suspendDialog = false"

              >

                Cancel

              </v-btn>

            </v-card-actions>

          </v-card>

        </v-dialog>

      </v-col>

    </v-row>

  </v-container>

</template>

<script>

import axios from 'axios'

export default {

  props: {

    pUsersArray: {type: Array, default: null},

    pIsSuperAdmin: {type: Boolean, default: false}

  },

  methods: {

    changeUserPrivilege (user, role) {

      const config = {

        headers: {

          'Authorization': 'Bearer '+this.$auth.token(),

        }

      }

      axios

        .post('http://127.0.0.1:8000/api/auth/user/setRole', { user\_id: user.id, role: role }, config)

        .then(res => {

          this.$emit('update')

        })

        .catch(err => {

          console.log(err)

        })

    },

    changeUserStatus (user, status) {

      const config = {

        headers: {

          'Authorization': 'Bearer '+this.$auth.token(),

        }

      }

      axios

        .post('http://127.0.0.1:8000/api/auth/user/setStatus', { user\_id: user.id, status: status }, config)

        .then(res => {

          this.$emit('update')

        })

        .catch(err => {

          console.log(err)

        })

    },

    isUserSuspended: function (status) {

      if (status === 'active') {

        return false

      }

      return true

    }

  }

}

</script>

**frontend/components/admin/Users.vue**

<template>

  <v-card>

    <v-card-title>

      Users

    </v-card-title>

    <v-card-subtitle>

      Active

    </v-card-subtitle>

    <v-card-text>

      <UserList

        :pUsersArray="users"

        @update="fetchUsers()"

      />

    </v-card-text>

    <v-card-subtitle>

      Suspended

    </v-card-subtitle>

    <v-card-text>

      <UserList

        :pUsersArray="suspendedUsers"

        @update="fetchUsers()"

      />

    </v-card-text>

  </v-card>

</template>

<script>

import axios from 'axios'

export default {

  components: {

    UserList: () => import('./UserList')

  },

  data () {

    return {

      users: [],

      suspendedUsers: [],

      suspendDialog: false

    }

  },

  created () {

    this.fetchUsers()

  },

  methods: {

    fetchUsers () {

      axios

        .get('http://127.0.0.1:8000/api/auth/users')

        .then(res => {

          let incomingUsers

          if (res.data.users.length) {

            incomingUsers = res.data.users

          } else {

            incomingUsers = [res.data.users]

          }

          this.users = []

          this.suspendedUsers= []

          for (const user of incomingUsers) {

            user.suspendDialog = false

            switch (user.status) {

              case 'active':

                this.users.push(user)

                break

              case 'suspended':

                this.suspendedUsers.push(user)

                break

            }

          }

        })

        .catch(err => {

          console.log(err)

        })

    }

  }

}

</script>

<style scoped>

  .users-container {

    max-height: 70vh;

    overflow: auto;

  }

</style>

**frontend/components/essentials/DeleteOfferDialog.vue**

<template>

  <v-dialog

    v-model="dialog"

  >

    <template

      #activator="{ on }"

    >

      <v-btn

        v-if="pOffer.author\_id === $auth.user().id || $auth.user().role !== 'user'"

        text

        color="error"

        v-on="on"

      >

        {{ $auth.user().role === "user" ? 'Delete' : 'Archive' }}

      </v-btn>

    </template>

    <v-card>

      <v-card-title>

        Attention!

      </v-card-title>

      <v-card-text>

        Would You really like to {{ $auth.user().role === 'user' ? 'delete' : 'archive' }} this offer? <br>

        {{ pOffer.title }}

      </v-card-text>

      <v-card-actions>

        <v-btn

          text

          width="50%"

          color="error"

          @click="deleteOffer()"

        >

          {{ $auth.user().role === "user" ? 'Delete' : 'Archive' }}

        </v-btn>

        <v-btn

          text

          width="50%"

          @click="dialog = false"

        >

          Cancel

        </v-btn>

      </v-card-actions>

    </v-card>

  </v-dialog>

</template>

<script>

export default {

  props: {

    pOffer: {type: Object, default: null}

  },

  data () {

    return {

      dialog: false

    }

  },

  methods: {

    deleteOffer () {

      this.dialog = false

      this.$emit('delete')

    }

  }

}

</script>

<style scoped>

  .dialog {

    position: relative;

  }

  .modal {

    position: absolute;

    background-color: rgba(red, green, blue, 1);

    width: 100%;

    height: 100%;

  }

</style>

**frontend/components/helpers/ConnectionError.vue**

<template>

  <div>

    <p>

      Server is not responding at the moment. <br>

      Try again later. <br>

      Error code: {{ errorCode }}

    </p>

  </div>

</template>

<script>

export default {

  props: {

    errorCode: {type: Number, default: null}

  }

}

</script>

**frontend/components/helpers/ImageLightbox.vue**

<template>

  <div

    :style="`max-height: ${pMaxHeight}; overflow: hidden;`"

  >

    <v-dialog v-model="lightboxDialog">

      <template #activator="{ on }">

        <v-img

          v-on="on"

          :src="src"

        />

      </template>

      <v-container>

        <v-img

          :src="src"

        />

      </v-container>

    </v-dialog>

  </div>

</template>

<script>

export default {

  name: 'ImageLightbox',

  props: {

    src: {type: String, default: null},

    pMaxHeight: {type: String, default: ''}

  },

  data () {

    return {

      lightboxDialog: false

    }

  }

}

</script>

**frontend/components/helpers/Loading.vue**

<template>

  <v-app>

    <v-container class="loading" />

  </v-app>

</template>

<style scoped>

  .loading {

    background-image: url("../../assets/logo.png");

    position: absolute;

    width: 30px;

    height: 30px;

    padding: 0;

    margin: 0;

    -webkit-animation:spin 4s linear infinite;

    -moz-animation:spin 4s linear infinite;

    animation:spin 4s linear infinite;

  }

  @-moz-keyframes spin { 100% { -moz-transform: rotate(360deg); } }

  @-webkit-keyframes spin { 100% { -webkit-transform: rotate(360deg); } }

  @keyframes spin { 100% { -webkit-transform: rotate(360deg); transform:rotate(360deg); } }

</style>

**frontend/components/helpers/ModalDialog.vue**

<template>

  <div

    class="modal"

    @click="closeModal()"

  >

    <div

      class="dialog"

    >

      <slot

        @closeModal="closeModal()"

      />

    </div>

  </div>

</template>

<script>

export default {

  methods: {

    closeModal () {

      this.$emit('closeModal')

    }

  }

}

</script>

<style scoped>

  .dialog {

    position: relative;

    max-width: 500px;

    left: 0;

    margin-left: auto;

    margin-right: auto;

    right: 0;

    overflow: auto;

  }

  .modal {

    position: fixed;

    top: 0;

    left: 0;

    height: 100vh;

    margin: 0;

    min-width: 100vw;

    background-color:rgba(0, 0, 0, 0.561);

    z-index: 9;

  }

</style>

**frontend/components/AddOffer.vue**

<template>

  <div>

    <v-form

      ref="ofForm"

      v-if="!response.status"

    >

      <v-card

        style="overflow: hidden;"

        max-width="100%"

      >

        <v-card-title>

          <h1

            class="headline font-weight-regular"

          >

            Add Offer

          </h1>

          <v-spacer />

          <v-btn

            icon

            @click="closeDialog()"

          >

            <v-icon>mdi-close</v-icon>

          </v-btn>

        </v-card-title>

        <v-card-text>

          <v-text-field

            v-model="title"

            :rules="[rules.required]"

            :counter="32"

            placeholder="Offer Title ..."

            label="Title"

            outlined

          />

          <v-row>

            <v-col>

              <v-textarea

                v-model="body"

                :rules="[rules.required]"

                :counter="600"

                rows="9"

                placeholder="Offer Description ..."

                label="Description"

                outlined

              />

            </v-col>

            <v-col>

              <v-card

                outlined

              >

                <v-card-title

                  class="title font-weight-light"

                >

                  Parts

                </v-card-title>

                <v-card-text>

                  <v-row

                    v-for="part in partObjects"

                    :key="part.id"

                  >

                    <v-col>

                      <v-text-field

                        v-model="part.label"

                        :rules="[rules.required]"

                        :counter="32"

                        placeholder="Part Label ..."

                        label="Label"

                        outlined

                      />

                    </v-col>

                    <v-col

                      v-if="categories.length"

                    >

                      <v-select

                        v-model="part.category"

                        required

                        label="Category"

                        solo

                        :items="categories"

                        item-text="label"

                        item-value="id"

                      />

                    </v-col>

                    <v-btn

                      v-if="part.id !== 0"

                      icon

                      @click="removeNewPart(part.id)"

                    >

                      <v-icon>mdi-close</v-icon>

                    </v-btn>

                  </v-row>

                  <v-row

                    v-if="!areCategoriesFilled"

                  >

                    <div

                      class="caption-category"

                    >

                      Provide a category for each part.

                    </div>

                  </v-row>

                  <v-btn

                    class="mx-2"

                    fab

                    dark

                    color="#616161"

                    @click="addNewPart()"

                  >

                    <v-icon dark>

                      mdi-plus

                    </v-icon>

                  </v-btn>

                </v-card-text>

              </v-card>

            </v-col>

          </v-row>

          <v-row>

            <v-col>

              <v-text-field

                v-model="contactPhone"

                :rules="[scoped\_rules.contactPhone]"

                :counter="12"

                label="Contact Phone"

                outlined

              />

            </v-col>

            <v-col>

              <v-text-field

                :value="contactEmail"

                disabled

                label="Contact Email"

                outlined

              />

            </v-col>

          </v-row>

        </v-card-text>

        <v-card-actions class="pb-0">

          <v-container class="pa-0">

            <v-row class="pl-3 pr-5">

              <v-card

                class="file-input-card"

                outlined

              >

                <v-card-text>

                  <v-file-input

                    v-model="files"

                    :rules="[scoped\_rules.filesRequired]"

                    multiple

                    width="50%"

                  />

                  <v-container>

                    <v-row class="align-center justify-center">

                      <v-card

                        v-if="files"

                        width="100%"

                        class="block"

                        color="blue-grey lighten-5"

                      >

                        <v-img

                          v-for="(image, index) in files"

                          :key="index"

                          class="d-inline-block ma-5"

                          max-width="200px"

                          max-height="200px"

                          :src="makeUrl(image)"

                        />

                      </v-card>

                    </v-row>

                  </v-container>

                </v-card-text>

              </v-card>

            </v-row>

            <div>

              <v-row

                class="caption caption-alert-wrapper"

              >

                <div

                  class="caption-alert-container"

                >

                  <v-icon

                    small

                  >

                    mdi-alert-outline

                  </v-icon>

                  Check the information validity before submitting.

                </div>

              </v-row>

            </div>

            <v-divider />

            <v-row class="pa-0">

              <!-- escape buttons -->

              <v-btn

                style="border-radius: 0 0 0 5px;"

                text

                color="primary"

                d-inline-block

                width="50%"

                height="50px"

                @click="submit()"

              >

                Submit

              </v-btn>

              <v-spacer />

              <v-btn

                style="border-radius: 0 0 5px 0;"

                text

                d-inline-block

                width="50%"

                height="50px"

                @click="closeDialog()"

              >

                Cancel

              </v-btn>

            </v-row>

          </v-container>

        </v-card-actions>

      </v-card>

    </v-form>

    <v-card

      v-if="response.status"

    >

      <v-card-title>

        {{ response.status }}

      </v-card-title>

      <v-card-text>

        {{ response.message }}

      </v-card-text>

      <v-btn

        class="dismiss-btn"

        depressed

        block

        @click="closeDialog(true)"

      >

        Ok

      </v-btn>

    </v-card>

  </div>

</template>

<script>

import external\_rules from '@/plugins/rules/rules.js'

import axios from 'axios'

export default {

  props: {

    pCategories: { type: Array, default: null }

  },

  data() {

    return {

      title: null,

      body: null,

      files: [],

      path: null, // debug var

      contactEmail: null,

      contactPhone: null,

      rules: external\_rules,

      scoped\_rules: {

        contactPhone: v => (!!v && v.length > 6 && v.length < 40 && /^[0-9]+$/.test(v)) || 'Provide a valid phone number without any symbols',

        filesRequired: v => (!!v && this.files.length >=1) || 'You need to attach at least 1 image'

      },

      partObjects: [

        {

          id: 0,

          label: null,

          category: null

        }

      ],

      categories: [],

      areCategoriesFilled: false,

      response: {

        status: null,

        message: null

      }

    }

  },

  watch: {

    partObjects: {

      deep: true,

      handler: function () {

        this.areCategoriesFilled = true

        this.partObjects.map(part => {

          if (!part.category) {

            this.areCategoriesFilled = false

          }

        })

      }

    }

  },

  created () {

    this.contactEmail = this.$auth.user().email

  },

  mounted () {

    if (this.pCategories) {

      this.categories = this.pCategories

    }

  },

  beforeDestroy () {

    this.closeDialog()

  },

  methods: {

    addNewPart () {

      this.partObjects.push({

        id: this.partObjects.length,

        label: null,

        category: null

      })

    },

    closeDialog (afterSubmit = false) {

      this.title = null

      this.body = null

      this.contactPhone = null

      this.files = []

      this.path = null

      this.formattedFiles = []

      this.areCategoriesFilled = false

      this.partObjects = [

        {

          id: 0,

          label: null,

          category: null

        }

      ]

      this.response = {

        status: null,

        message: null

      }

      if (!afterSubmit) {

        this.$refs.ofForm.reset()

      }

      this.$emit('closeAddOfferDialog')

    },

    makeUrl (file) {

      return URL.createObjectURL(file)

    },

    removeNewPart (partId) {

      let filtered = this.partObjects.filter(part => (part.id !== partId))

      if (filtered.length) {

        let prevPartId = 0

        filtered.map(part => {

          if (part.id > 0) {

            if (part.id !== prevPartId + 1) {

              part.id = prevPartId + 1

            }

            prevPartId = part.id

          }

        })

      }

      this.partObjects = filtered.length ? filtered : [filtered]

    },

    submit () {

      if (this.$refs.ofForm.validate() && this.areCategoriesFilled) {

        const offerData = new FormData()

        offerData.append('user\_id', this.$auth.user().id)

        offerData.append('title', this.title)

        offerData.append('body', this.body)

        offerData.append('contact\_phone', this.contactPhone)

        offerData.append('contact\_email', this.contactEmail)

        offerData.append('preview\_image\_id', 0)

        offerData.append('parts', JSON.stringify(this.partObjects))

        // offerData

        for (let i = 0; i < this.files.length; i++) {

          offerData.append('images[]', this.files[i], this.files[i].name)

        }

        const config = {

          headers: {

            'Authorization': 'Bearer '+this.$auth.token(),

            'Content-Type': 'multipart/form-data'

          }

        }

        axios

          .post('http://127.0.0.1:8000/api/auth/offer/add', offerData, config)

          .then (res => {

            switch (res.status) {

              case 200:

                this.response.status = 'Success'

                this.response.message = 'Offer has been successfully submitted!'

                break

              case 500:

                this.response.status = 'Error'

                this.response.message = 'Server is not responging. Error code: 500'

                break

              default:

                this.response.status = 'Hmm...'

                this.response.message = 'Something went wrong...'

                break

            }

          })

          .catch ((err) => {

            console.log(err)

          })

      }

    }

  }

}

</script>

<style scoped>

  .caption-alert-container {

    border-radius: 10px;

    background-color: sandybrown;

    text-align: center;

  }

  .caption-alert-wrapper {

    display: inline;

  }

  .caption-category {

    margin: 0 auto;

    border-radius: 5px;

    border: 1px solid lightgrey;

    width: 100%;

    padding: 5px;

    margin-bottom: 15px;

    text-align: center;

    background-color: rgb(241, 241, 241);

  }

  .dismiss-btn {

    margin: 0 auto;

  }

  .file-input-card {

    width: 100%;

  }

</style>

**frontend/components/AuctionDisplay.vue**

<template>

  <v-container

    class="contains"

  >

    <ImageLightbox

      :src="pAuction.auction\_object.preview\_image"

      class="img"

    />

    <v-divider

      vertical

    />

    <h2

      class="title"

    >

      {{ pAuction.auction\_object.name }}

    </h2>

    <v-divider

      vertical

    />

    <v-container>

      <v-row

        justify="center"

      >

        Started At: {{ new Date(pAuction.started\_at).toLocaleDateString() }}

      </v-row>

      <v-row

        v-if="pAuction.finished\_at"

        justify="center"

      >

        Finished At: {{ new Date(pAuction.finished\_at).toLocaleDateString() }}

      </v-row>

    </v-container>

    <v-divider

      vertical

    />

    <!-- charity block -->

    <v-container

      v-if="isCharity"

    >

      <v-row

        class="charity-row"

      >

        <v-col

          class="participant-col"

        >

          Goal

          <v-chip>

            {{ pAuction.auction\_data[0].goal }} EUR

          </v-chip>

        </v-col>

      </v-row>

    </v-container>

    <!-- /charity block -->

    <!-- commercial block -->

    <v-container

      class="commercial-container"

      v-if="isCommercial"

    >

      <v-row>

        <v-col

          v-if="winner"

          cols="3"

          class="participant-col"

        >

          <div>

            <v-img

              class="profile-pic"

              :src="winner.avatar\_path"

              width="48"

              height="48"

            />

          </div>

        </v-col>

        <v-col

          v-if="winner"

          cols="4"

          class="participant-col"

        >

          {{ winner.username }}

        </v-col>

        <v-col

          class="participant-col"

        >

          <span>Starting bid</span>

          <v-row>

            <v-chip>

              {{ pAuction.auction\_data[0].start\_bid }} EUR

            </v-chip>

          </v-row>

          <span

            v-if="winner"

          >Leading bid</span>

          <v-row

            v-if="winner"

          >

            <v-chip

              color="#9CCC65"

            >

              {{ winner.amount }} EUR

            </v-chip>

          </v-row>

        </v-col>

      </v-row>

    </v-container>

    <!-- /commercial block -->

  </v-container>

</template>

<script>

export default {

  components: {

    ImageLightbox: () => import('./helpers/ImageLightbox')

  },

  props: {

    pAuction: {type: Object, default: null},

    pType: {type: String, default: null}

  },

  data () {

    return {

      isCharity: false,

      isCommercial: false,

      // commerical

      winner: null

    }

  },

  created () {

    if (this.pType === 'charity') {

      this.isCharity = true

    } else if (this.pType === 'commercial') {

      this.isCommercial = true

    }

    if (this.isCommercial) {

      this.getCommercialWinner()

    }

  },

  methods: {

    // commerical

    getCommercialWinner () {

      const auctionData = this.pAuction.auction\_data

      this.winner = this.pAuction.participants.find(participant => participant.id === auctionData[0].highest\_bid\_user\_id)

    }

  }

}

</script>

<style scoped>

  .charity-row {

    align-items: center;

    text-align: center;

  }

  .commercial-container {

    margin-left: 20px;

  }

  .contains {

    max-height: 130px;

    align-items: center;

    display: inline-flex;

  }

  .img {

    max-width: 200px;

    max-height: 200px;

    margin-right: 20px;

  }

  .participant-col {

    align-self: center;

    justify-self: center;

  }

  .profile-pic {

    border-radius: 50%;

  }

  .title {

    margin-left: 20px;

    margin-right: 20px;

    width: 100%;

  }

</style>

**frontend/components/AuctionHistory.vue**

<template>

  <v-container

    class="history-container"

  >

    <v-card>

      <v-card-title

        @click="isHistoryExpanded = !isHistoryExpanded"

      >

        <h2>{{ auctionType }} Auctions</h2>

        <v-spacer />

        <v-icon

          v-if="!isHistoryExpanded"

        >

          mdi-chevron-down

        </v-icon>

        <v-icon

          v-else

        >

          mdi-chevron-up

        </v-icon>

      </v-card-title>

      <v-card-text

        v-if="isHistoryExpanded"

      >

        <v-card

          v-for="auction in pAuctionArray"

          :key="auction.id"

          outlined

          class="object-card"

        >

          <AuctionDisplay

            :pAuction="auction"

            :pType="pType"

          />

        </v-card>

      </v-card-text>

    </v-card>

  </v-container>

</template>

<script>

export default {

  components: {

    AuctionDisplay: () => import('./AuctionDisplay')

  },

  props: {

    pAuctionArray: {type: Array, default: null},

    pType: {type: String, default: null},

  },

  data () {

    return {

      isHistoryExpanded: false

    }

  },

  computed: {

    auctionType: function () {

      if (this.pType) {

        switch (this.pType) {

          case 'charity':

            return 'Charity'

            break

          case 'commercial':

            return 'Commercial'

            break

        }

      }

      return null

    }

  }

}

</script>

<style scoped>

  .history-container {

    background-color: white;

  }

  .object-card {

    width: 100%;

  }

</style>

**frontend/components/AuctionTemplate.vue**

<template>

  <v-card

    outlined

  >

    <v-card-title>

      <ImageLightbox

        class="lightbox"

        :src="imgSrc()"

      />

      <h2

        class="auction-object-title font-weight-light"

      >

        {{ pAuction.auction\_object.name }}

      </h2>

    </v-card-title>

    <v-card-text>

      <slot />

      <v-container

        v-if="!isUserIsLogged"

      >

        <h3>You need to Log In to be able to participate</h3>

      </v-container>

      <v-container>

        <div

          v-for="participant in sortedParticipants"

          :key="participant.id"

        >

          <v-divider />

          <v-container

            class="participant-row"

          >

            <v-row

              :class="{ 'hightlight-row' : participant.id === $auth.user().id }"

            >

              <v-col

                cols="2"

              >

                <div>

                  <v-img

                    class="profile-pic"

                    :src="participant.avatar\_path"

                    width="32"

                    height="32"

                  />

                </div>

              </v-col>

              <v-col

                cols="5"

                class="participant-col"

              >

                {{ participant.username }}

              </v-col>

              <v-col

                class="participant-col"

              >

                <v-chip>

                  {{ participant.amount }} EUR

                </v-chip>

              </v-col>

            </v-row>

          </v-container>

        </div>

      </v-container>

    </v-card-text>

    <v-card-actions

      v-if="checkPayPal"

    >

      <v-container

        v-if="canUserParticipate"

      >

        <v-text-field

          v-model="amount"

          label="Amount (EUR)"

          outlined

          clearable

          :rules="[scoped\_rules.charityAmount]"

        />

        <v-btn

          @click="showPayPalDialog = true"

          block

          depressed

          color="#ffc439"

          :disabled="!correctAmount"

        >

          Participate

        </v-btn>

        <modal-dialog

          v-if="showPayPalDialog"

          class="modal"

        >

          <v-card>

            <v-container>

              <PayPal

                class="dialog"

                :pAmount="Number(amount)"

                :pProduct="auctionProduct"

                @closeModal="closePayPalDialog()"

              />

            </v-container>

          </v-card>

        </modal-dialog>

      </v-container>

    </v-card-actions>

  </v-card>

</template>

<script>

export default {

  components: {

    ImageLightbox: () => import('./helpers/ImageLightbox'),

    ModalDialog: () => import('./helpers/ModalDialog'),

    PayPal: () => import('./PayPal')

  },

  props: {

    pAuction: {type: Object, default: null},

    pUsePayPal: {type: Boolean, default: true}

  },

  data () {

    return {

      amount: 100.00,

      total\_amount: 0,

      showPayPalDialog: false,

      auctionProduct: {},

      scoped\_rules: {

        charityAmount: v => (!!v && v > 0 && v <= (this.pAuction.auction\_data[0].goal - this.total\_amount)) || 'Invalid amount'

      },

      sortedParticipants: null

    }

  },

  computed: {

    auctionParticipants: function () {

      return this.pAuction.participants

    },

    canUserParticipate: function () {

      if (this.auctionParticipants.find(participant => participant.id === this.$auth.user().id)) {

        return false

      }

      return true

    },

    checkPayPal: function () {

      if (Object.keys(this.$auth.user()).length && this.pUsePayPal) {

        return true

      }

      return false

    },

    correctAmount: function () {

      if (this.total\_amount >= this.pAuction.auction\_data[0].goal) {

        return false

      }

      if (this.amount > 0 && this.amount <= (this.pAuction.auction\_data[0].goal - this.total\_amount)) {

        return true

      }

      return false

    },

    isUserIsLogged: function () {

      if (Object.keys(this.$auth.user()).length !== 0) {

        return true

      }

      return false

    }

  },

  watch: {

    pAuction: {

      deep: true,

      handler: function () {

        this.getTotalAmount()

        this.setTotalAmount()

      }

    }

  },

  created () {

    this.getTotalAmount()

    this.setTotalAmount()

    this.auctionProduct = {

      auctionId: this.pAuction.id,

      description: this.pAuction.auction\_object.name

    }

    this.sortParticipants()

  },

  methods: {

    closePayPalDialog () {

      this.showPayPalDialog = false

      this.$emit('updateAuction')

    },

    getTotalAmount () {

      this.total\_amount = 0

      this.auctionParticipants.map(participant => {

        this.total\_amount += participant.amount

      })

    },

    imgSrc () {

      if (this.pAuction) {

        return this.pAuction.auction\_object.preview\_image

      }

      return ''

    },

    setTotalAmount () {

      let operation

      switch (this.pAuction.type) {

        case 'charity':

          operation = 'setCharityTotal'

          break

        case 'commercial':

          operation = 'setCommercialTotal'

          break

      }

      this.$store.commit(operation, this.total\_amount)

    },

    sortParticipants () {

      this.sortedParticipants = this.auctionParticipants.sort((a, b) => (b.amount - a.amount))

    }

  }

}

</script>

<style scoped>

  .auction-object-title {

    margin-top: 10px;

  }

  .hightlight-row {

    background-color: lightgrey;

  }

  .lightbox {

    width: 100%;

  }

  .participant-col {

    align-self: center;

  }

  .participant-row {

    padding: 0;

  }

  .profile-pic {

    border-radius: 50%;

  }

</style>

**frontend/components/CharityAuction.vue**

<template>

  <AuctionTemplate

    :pAuction="pAuction"

    @updateAuction="fetchUpdatedAuctions()"

  >

    <v-container>

      <v-row>

        <v-progress-linear

          height="25"

          color="amber"

          :value="totalProgress"

        >

          <span

            class="font-weight-light progress-caption"

          >Progress</span>

          <v-spacer />

          <span

            class="font-weight-light progress-caption"

          >{{ totalAmount }} / {{ pAuction.auction\_data[0].goal || 0 }} EUR</span>

        </v-progress-linear>

      </v-row>

    </v-container>

  </AuctionTemplate>

</template>

<script>

export default {

  components: {

    AuctionTemplate: () => import('./AuctionTemplate')

  },

  props: {

    pAuction: {type: Object, default: null}

  },

  computed: {

    totalAmount: function () {

      return this.$store.getters['charityTotal']

    },

    totalProgress: function () {

      return (this.totalAmount / this.pAuction.auction\_data[0].goal) \* 100

    }

  },

  methods: {

    fetchUpdatedAuctions () {

      this.$emit('updateAuction')

    }

  }

}

</script>

<style scoped>

  .progress-caption {

    margin-left: 10px;

    margin-right: 10px;

  }

</style>

**frontend/components/CommercialAuction.vue**

<template>

  <AuctionTemplate

    :pAuction="pAuction"

    :pUsePayPal="false"

  >

    <v-container

      class="bid-container"

    >

      <div

        class="crucial-info-container"

      >

        <h2

          class="font-weight-light"

        >

          Starting bid: {{ pAuction.auction\_data[0].start\_bid }} EUR

        </h2>

        <v-spacer />

        <h3

          class="font-weight-light"

        >

          Ends on: <span class="font-weight-bold">{{ new Date(pAuction.auction\_data[0].end\_date).toLocaleDateString() }}</span>

        </h3>

      </div>

      <v-row

        v-if="checkUserParticipation"

      >

        <v-col

          class="bid-col"

        >

          <v-text-field

            v-model="bid"

            class="bid-field"

            :value="highestBid"

            :rules="[scoped\_rules.commericalAmount]"

            label="Amount (EUR)"

            outlined

          />

        </v-col>

        <v-col

          class="bid-col"

        >

          <v-dialog

            v-model="showConfirmationDialog"

          >

            <template #activator="{ on }">

              <v-btn

                depressed

                color="primary"

                v-on="on"

                @click="showConfirmationDialog = true"

              >

                Place bid

              </v-btn>

            </template>

            <v-card

              v-if="!isPaid"

            >

              <v-card-title>

                Attention!

              </v-card-title>

              <v-card-text>

                By placing a bid of {{ bid }} EUR

                You agree to pay the delivered amount within

                24 hours after the end of the auction.

                Continue?

              </v-card-text>

              <v-card-actions>

                <v-btn

                  text

                  color="primary"

                  @click="submitBid()"

                >

                  Ok

                </v-btn>

                <v-btn

                  text

                  color="error"

                  @click="showConfirmationDialog = false"

                >

                  Cancel

                </v-btn>

              </v-card-actions>

            </v-card>

            <v-card

              v-if="isPaid"

            >

              <v-card-title>

                {{ response.status }}

              </v-card-title>

              <v-card-text>

                {{ response.message }}

              </v-card-text>

              <v-card-actions>

                <v-btn

                  block

                  depressed

                  @click="closeDialog()"

                >

                  Ok

                </v-btn>

              </v-card-actions>

            </v-card>

          </v-dialog>

        </v-col>

      </v-row>

      <v-row

        class="bid-row"

        v-if="pAuction.auction\_data[0].highest\_bid\_user\_id"

      >

        <v-col>

          Highest bid:

        </v-col>

        <v-col>

          <v-chip

            v-if="bidLeader"

            large

          >

            <v-img

              class="profile-pic"

              :src="bidLeader.avatar\_path"

              width="32"

              height="32"

            />

            {{ bidLeader.username }}

            {{ getHighestBidAmount() }} EUR

          </v-chip>

        </v-col>

      </v-row>

    </v-container>

  </AuctionTemplate>

</template>

<script>

import axios from 'axios'

import insertAuctionParticipant from '../plugins/insertAuctionParticipant'

export default {

  components: {

    AuctionTemplate: () => import('./AuctionTemplate')

  },

  props: {

    pAuction: {type: Object, default: null}

  },

  data () {

    return {

      bid: 0,

      scoped\_rules: {

        commericalAmount: v => (!!v && v > (this.pAuction.auction\_data[0].highest\_bid\_user\_id ? this.highestBid : this.highestBid - 1 )) || 'Invalid amount'

      },

      showConfirmationDialog: false,

      response: {

        status: null,

        message: ''

      },

      isPaid: false,

      bidLeader: {},

      highestBid: 0

    }

  },

  computed: {

    checkUserParticipation: function (params) {

      const now = new Date().getTime()

      const endDate = new Date(this.pAuction.auction\_data[0].end\_date).getTime()

      if (now > endDate) {

        return false

      }

      if (this.isUserIsLogged && !this.pAuction.participants.find(participant =>

        participant.id === this.$auth.user().id

      )) {

        return true

      }

      return false

    },

    isUserIsLogged: function () {

      if (Object.keys(this.$auth.user()).length !== 0) {

        return true

      }

      return false

    }

  },

  watch: {

    pAuction: function () {

      this.getHighestBidUser(this.pAuction.auction\_data[0].highest\_bid\_user\_id)

    }

  },

  created () {

    this.highestBid = this.pAuction.auction\_data[0].start\_bid

    this.bid = this.highestBid

    this.getHighestBidUser(this.pAuction.auction\_data[0].highest\_bid\_user\_id)

  },

  methods: {

    closeDialog () {

      this.showConfirmationDialog = false

      this.$emit('updateAuction')

    },

    getHighestBidAmount () {

      return this.bidLeader.amount

    },

    getHighestBidUser (userId) {

      this.bidLeader = this.pAuction.participants.find(participant =>

        participant.id === userId

      )

      if (this.bidLeader) {

        this.highestBid = this.bidLeader.amount

        this.bid = this.highestBid + 1

      }

    },

    async submitBid () {

      const insertStatus = await insertAuctionParticipant(

        this.$auth.token(),

        this.$auth.user().id,

        this.pAuction.id,

        this.bid,

        this.response,

        this.isPaid

      )

      const config = {

        headers: {

          'Authorization': 'Bearer '+this.$auth.token(),

          'Content-Type': 'multipart/form-data'

        }

      }

      const auctionBidData = new FormData()

      auctionBidData.append('auction\_id', this.pAuction.id)

      auctionBidData.append('user\_id', this.$auth.user().id)

      auctionBidData.append('amount', this.bid)

      axios

        .post('http://127.0.0.1:8000/api/auth/auction/checkBid', auctionBidData, config)

        .then (res => {

          console.log(res)

        })

        .catch ((err) => {

          console.log(err)

        })

      this.response.status = insertStatus.response.status

      this.response.message = insertStatus.response.message

      this.isPaid = insertStatus.isPaid

    }

  }

}

</script>

<style scoped>

  .bid-col {

    align-self: center;

  }

  .bid-container {

    padding: 0;

  }

  .bid-field {

    margin-top: 30px;

  }

  .bid-row {

    align-items: center;

    text-align: center;

  }

  .crucial-info-container {

    display: flex;

  }

  .profile-pic {

    border-radius: 50%;

    margin-right: 10px;

  }

</style>

**frontend/components/Login.vue**

<template>

  <v-container fill-height>

    <v-form ref="lForm">

      <v-text-field

        v-model="loginData"

        label="Username or Email"

        prepend-icon="mdi-account"

        :rules="[rules.required]"

        @keypress.enter.native="login"

      />

      <v-text-field

        v-model="password"

        label="Password"

        prepend-icon="mdi-lock"

        type="password"

        :rules="[rules.required]"

        @keypress.enter.native="login"

      />

      <v-expand-transition>

        <p

          v-show="isError"

          style="color: red;"

          class="subtitle-2 pl-4"

        >

          <v-icon

            color="red"

            size="17px"

            class="pr-0 pb-1"

          >

            mdi-alert

          </v-icon> <span v-if="!isUserSuspended">Wrong username/email or password</span>

          <span v-else>Your account has been suspended</span>

        </p>

      </v-expand-transition>

      <v-card-actions>

        <v-btn

          class="mx-auto"

          text

          :disabled="isLoading"

          :loading="isLoading"

          @click="login"

        >

          Sign In

        </v-btn>

      </v-card-actions>

    </v-form>

  </v-container>

</template>

<script>

import { mapMutations } from 'vuex'

import external\_rules from  '@/plugins/rules/rules.js'

import fetchFavoriteOffers from '@/plugins/fetchFavoriteOffers.js'

export default {

  data() {

    return {

      loginData: null,

      password: null,

      info: 'null',

      isLoading: false,

      rules: external\_rules,

      isUserSuspended: true,

      isError: false

    }

  },

  methods: {

    ...mapMutations([

      'fadeLogin',

      'changeAuthState'

    ]),

    login() {

      this.isUserSuspended = false

      if (this.$refs.lForm.validate()){

        this.isError = false

        this.isLoading = true

        let that = this

        this.$auth.login({

          data: {

            email: that.loginData,

            username: that.loginData,

            password: that.password

          },

          success: function() {

            this.isLoading = false

            this.fadeLogin()

            this.changeAuthState()

            const redirectTo = this.redirect ? this.redirect.from.name : this.$auth.user().role === 'super\_admin' ? 'admin.dashboard' : this.$auth.user().role === 'admin' ? 'admin.dashboard' : 'dashboard'

            this.$router.push({name: redirectTo})

            this.fetchFavoriteOffers()

          },

          error : function(err) {

            if (err.response.data.error === 'account\_suspended') {

              this.isUserSuspended = true

            }

            this.isLoading = false

            this.isError = true

          },

          rememberMe: true,

          fetchUser: true,

        })

      }

    },

    async fetchFavoriteOffers () {

      const result = await fetchFavoriteOffers(this.$auth.user().id)

      if (result) {

        this.$store.commit('setFavoriteOffers', result)

      }

    }

  }

}

</script>

<style scoped>

  .form {

    z-index: 1;

  }

</style>

**frontend/components/OffersDisplay.vue**

<template>

  <div>

    <div

      v-if="pIsTiled"

    >

      <v-card>

        <v-card-title>

          <div

            class="profile-pic"

          >

            <v-img

              width="32"

              height="32"

              :src="pOffer.preview\_image"

            />

          </div>

          {{ pOffer.title }}

          <v-spacer />

          <v-dialog

            v-model="showOfferDialog"

            max-width="1500px"

          >

            <template #activator="{on}">

              <v-btn

                text

                v-on="on"

              >

                Expand

              </v-btn>

            </template>

            <ShowOffer

              :pCategories="pCategories"

              :pOffer="pOffer"

              @closeShowDialog="closeShowOfferDialog"

            />

          </v-dialog>

        </v-card-title>

      </v-card>

    </div>

    <div

      v-if="!pIsTiled"

    >

      <v-card style="border-radius: 5px 5px 0 5px;">

        <v-row

          ml-5

          class="title"

        >

          <v-card-title

            pl-5

            class="headline"

          >

            <div>

              {{ pOffer.title }}

            </div>

            <v-spacer />

            <v-btn

              v-if="pOffer.canBeFavorited"

              icon

              @click="favoriteOffer()"

            >

              <v-icon

                v-if="pOffer.isFavorite"

              >

                mdi-heart

              </v-icon>

              <v-icon

                v-else

              >

                mdi-heart-outline

              </v-icon>

            </v-btn>

          </v-card-title>

        </v-row>

        <v-row>

          <v-col lg="4">

            <image-lightbox

              :pMaxHeight="'250px'"

              :src="pOffer.preview\_image"

            />

          </v-col>

          <v-col>

            <v-card-text>{{ pOffer.body }}</v-card-text>

          </v-col>

        </v-row>

      </v-card>

      <v-row

        class="justify-end"

        mr-5

      >

        <v-dialog

          v-model="showOfferDialog"

          max-width="1500px"

        >

          <template #activator="{on}">

            <v-btn

              x-large

              style="border-radius: 0 0 5px 5px;margin-right: 11px"

              color="primary"

              v-on="on"

            >

              More

            </v-btn>

          </template>

          <ShowOffer

            :pCategories="pCategories"

            :pOffer="pOffer"

            @closeShowDialog="closeShowOfferDialog"

          />

        </v-dialog>

      </v-row>

    </div>

  </div>

</template>

<script>

import axios from 'axios'

import ImageLightbox from '@/components/helpers/ImageLightbox.vue'

import ShowOffer from '@/components/ShowOffer.vue'

import fetchFavoriteOffers from '@/plugins/fetchFavoriteOffers.js'

export default {

  components: {

    ImageLightbox,

    ShowOffer

  },

  props: {

    pCategories: { type: Array, default: null },

    pOffer: {type: Object, default: null},

    pIsTiled: {type: Boolean, default: false},

    pIsAdmin: {type: Boolean, default: false},

  },

  data () {

    return {

      showOfferDialog: false

    }

  },

  methods: {

    closeShowOfferDialog () {

      this.showOfferDialog = false

      if (this.pIsAdmin) {

        this.$emit('reloadOffers')

      }

    },

    favoriteOffer () {

      const config = {

        headers: {

          'Authorization': 'Bearer '+this.$auth.token(),

          'Content-Type': 'multipart/form-data'

        }

      }

      const favoritedOfferData = new FormData()

      favoritedOfferData.append('offer\_id', this.pOffer.id)

      favoritedOfferData.append('user\_id', this.$auth.user().id)

      if (!this.pOffer.isFavorite) {

        favoritedOfferData.append('action', 'set')

      } else {

        favoritedOfferData.append('action', 'delete')

      }

      axios

        .post('http://127.0.0.1:8000/api/auth/offers/setFavorite', favoritedOfferData, config)

        .then(async () => {

          const result = await fetchFavoriteOffers(this.$auth.user().id)

          if (result) {

            this.$store.commit('setFavoriteOffers', result)

          }

          this.$emit('updateOffers')

        })

        .catch(err => {

          console.log(err)

        })

    }

  }

}

</script>

<style scoped>

  .title {

    display: contents;

  }

  .profile-pic {

    margin-right: 30px;

  }

</style>

**frontend/components/OffersFilters.vue**

<template>

  <v-card

    outlined

  >

    <v-card-title

      @click="filtersExpanded = !filtersExpanded"

    >

      Filters

      <v-spacer />

      <div>

        <v-icon

          v-if="!filtersExpanded"

        >

          mdi-chevron-down

        </v-icon>

        <v-icon

          v-else

        >

          mdi-chevron-up

        </v-icon>

      </div>

    </v-card-title>

    <v-card-text

      class="filters-container"

      v-if="filtersExpanded"

    >

      <h3

        class="font-weight-light"

      >

        Title

      </h3>

      <v-text-field

        clearable

        v-model="title"

        outlined

        placeholder="Offer Title"

        dense

      />

      <h3

        class="font-weight-light"

      >

        Description

      </h3>

      <v-text-field

        clearable

        v-model="description"

        outlined

        placeholder="Offer Title"

        dense

      />

      <h3

        class="font-weight-light"

      >

        Parts

      </h3>

      <v-text-field

        clearable

        v-model="parts"

        outlined

        placeholder="Offer Title"

        dense

      />

      <h3

        class="font-weight-light"

      >

        Category

      </h3>

      <v-select

        clearable

        dense

        v-model="category"

        label="Category"

        solo

        :items="pCategories"

        item-text="label"

        item-value="id"

      />

      <div

        class="sorting-container"

      >

        <div

          class="sorting-title"

          @click="sortingExpanded = !sortingExpanded"

        >

          <h2>Sorting</h2>

          <v-spacer />

          <div>

            <v-icon

              v-if="!sortingExpanded"

            >

              mdi-chevron-down

            </v-icon>

            <v-icon

              v-else

            >

              mdi-chevron-up

            </v-icon>

          </div>

        </div>

        <div

          class="sorting-contains"

          v-if="sortingExpanded"

        >

          <h3

            class="font-weight-light"

          >

            By Date

          </h3>

          <v-select

            clearable

            dense

            v-model="date"

            solo

            :items="dateSorting"

          />

        </div>

      </div>

    </v-card-text>

    <v-divider />

    <v-card-actions

      class="actions-container"

      v-if="filtersExpanded"

    >

      <v-container

        class="btns-container"

      >

        <v-row>

          <v-col>

            <v-btn

              depressed

              block

              @click="fetchFilteredOffers()"

            >

              Search

            </v-btn>

          </v-col>

          <v-col>

            <v-btn

              depressed

              block

              @click="clearFilters()"

            >

              Clear

            </v-btn>

          </v-col>

        </v-row>

      </v-container>

    </v-card-actions>

  </v-card>

</template>

<script>

export default {

  props: {

    pCategories: { type: Array, default: null }

  },

  data () {

    return {

      title: null,

      description: null,

      parts: null,

      category: null,

      date: null,

      categories: [],

      filtersExpanded: false,

      sortingExpanded: false,

      dateSorting: [

        'accending',

        'descending'

      ]

    }

  },

  mounted () {

    if (this.pCategories) {

      this.categories = this.pCategories

    }

  },

  methods: {

    clearFilters () {

      this.title = null

      this.description = null

      this.parts = null

      this.category = null

      this.fetchFilteredOffers()

    },

    fetchFilteredOffers () {

      const offerFilters = {

        title: this.title,

        description: this.description,

        parts: this.parts,

        category: this.category,

        date: this.date

      }

      this.$store.commit('setOffersFilters', offerFilters)

      this.$emit('updateFilters')

    }

  }

}

</script>

<style scoped>

  .actions-container, .filters-container {

    padding-bottom: 0;

    margin-bottom: 20px;

  }

  .btns-container {

    padding-top: 0;

  }

  .sorting-container {

    width: 100%;

  }

  .sorting-contains {

    margin-top: 20px;

  }

  .sorting-title {

    display: flex;

  }

</style>

**frontend/components/PayPal.vue**

<template>

  <div>

    <div v-if="!isPaid">

      <div

        style="float: right;"

      >

        <v-btn

          icon

          @click="closePayPal()"

        >

          <v-icon>mdi-close</v-icon>

        </v-btn>

      </div>

      <div

        class="description"

      >

        <h1>Auction participation amount</h1>

        <h3>for</h3>

        <h2>{{ product.description }}</h2>

        <div

          class="price"

        >

          <h1>{{ product.price }} EUR</h1>

        </div>

      </div>

    </div>

    <div v-if="isPaid">

      <h1>{{ response.status }}!</h1>

      <h1>{{ response.message }}</h1>

      <v-btn

        block

        depressed

        @click="closePayPal()"

      >

        Ok

      </v-btn>

    </div>

    <div

      v-if="!isPaid"

    >

      <div

        class="loading-container"

        v-if="!isLoaded"

      >

        <v-progress-circular

          class="loading"

          indeterminate

          color="white"

        />

      </div>

      <div

        class="paypal-container"

        ref="paypal"

      />

      <div

        v-if="isError"

      >

        <span

          class="caption"

          style="color: red;"

        >{{ response.message }}!</span>

      </div>

    </div>

  </div>

</template>

<script>

import axios from 'axios'

import insertAuctionParticipant from '../plugins/insertAuctionParticipant'

export default {

  props: {

    pAmount: { type: Number, default: 0 },

    pProduct: { type: Object, default: null },

  },

  data () {

    return {

      isLoaded: false,

      isPaid: false,

      product: {

        price: 0,

        description: ""

      },

      response: {

        status: null,

        message: ''

      },

      isError: false

    };

  },

  created () {

    this.product.price = this.pAmount

    this.product.description = this.pProduct.description

  },

  mounted () {

    const script = document.createElement("script")

    script.src = "https://www.paypal.com/sdk/js?currency=EUR&client-id=AfmV7MYZEqRm0p\_NtJREd41YtDYyRm5LVkxeXnnQW1kRs9YK2hCG3ItNUzecp173afMRFOZoJWq9jhMu"

    script.addEventListener("load", this.loadPayPalComponent)

    document.body.appendChild(script);

  },

  methods: {

    closePayPal () {

      this.$emit('closeModal')

    },

    loadPayPalComponent: function() {

      window.paypal.Buttons({

        style: {

          size: 'small',

          color: 'gold',

          shape: 'pill',

          label: 'checkout',

        },

        createOrder: (data, actions) => {

          this.isLoaded = false

          return actions.order.create({

            purchase\_units: [

              {

                description: this.product.description,

                amount: {

                  currency\_code: "EUR",

                  value: this.product.price

                }

              }

            ]

          })

        },

        onApprove: async (data, actions) => {

          const order = await actions.order.capture();

          if (order.status === "COMPLETED") {

            this.insertAuctionParticipant(this.product.price)

          }

        },

        onError: err => {

          this.response.status = 'Error'

          this.response.message = 'Error occured on this transaction'

          this.isLoaded = true

          this.isError = true

        },

        onCancel: (data, actions) => {

          this.isLoaded = true

        }

      })

      .render(this.$refs.paypal)

        .then(() => {

          this.isLoaded = true

        })

    },

    async insertAuctionParticipant (price) {

      const insertStatus = await insertAuctionParticipant(

        this.$auth.token(),

        this.$auth.user().id,

        this.pProduct.auctionId,

        price,

        this.response,

        this.isPaid

      )

      this.response.status = insertStatus.response.status

      this.response.message = insertStatus.response.message

      this.isPaid = insertStatus.isPaid

      this.isLoaded = true

    }

  }

}

</script>

<style scoped>

  .description {

    text-align: center;

  }

  .loading {

    top: 50%;

  }

  .loading-container {

    position: absolute;

    background-color:rgba(0, 0, 0, 0.561);

    top: 0;

    left: 0;

    width: 100%;

    text-align-last: center;

    height: 100%;

    z-index: 90;

  }

  .paypal-container {

    position: relative;

    z-index: 0;

  }

  .price {

    border: 1px solid lightgrey;

    border-radius: 50px;

  }

</style>

**frontend/components/ShowOffer.vue**

<template>

  <v-form>

    <v-card>

      <v-card-title>

        <div>

          <h3>{{ pOffer.title }}</h3>

          <h4 class="subtitle-1">

            added on {{ new Date(pOffer.created\_at).toLocaleDateString() }}

          </h4>

        </div>

        <v-spacer />

        <DeleteOfferDialog

          v-if="pOffer.status === 'active'"

          :pOffer="pOffer"

          @delete="deleteOffer()"

        />

        <v-btn

          icon

          @click="closeDialog()"

        >

          <v-icon>mdi-close</v-icon>

        </v-btn>

      </v-card-title>

      <v-card-text>

        <v-card

          style="margin-bottom: 10px"

          outlined

        >

          <v-card-title

            class="font-weight-light"

          >

            Description

          </v-card-title>

          <v-card-text>

            {{ pOffer.body }}

          </v-card-text>

        </v-card>

        <v-card

          outlined

          v-if="parts.length"

        >

          <v-card-title

            class="font-weight-light"

          >

            Parts

          </v-card-title>

          <v-card-text>

            <v-data-table

              :headers="headers"

              :items="partsItems"

              :items-per-page="5"

              item-key="label"

              group-by="category"

              show-group-by

              dense

            />

          </v-card-text>

        </v-card>

        <v-container>

          <v-card>

            <image-lightbox

              v-for="image in images"

              :key="image"

              :src="image"

              class="image d-inline-block ma-5"

            />

          </v-card>

        </v-container>

        <v-container

          class="contacts"

        >

          <h3>

            Contacts

          </h3>

          <v-row>

            <v-col>

              <h3 class="subtitle-1">

                Added By:

              </h3>

              <v-chip

                large

              >

                <v-row

                  class="author-row"

                >

                  <v-col

                    class="author-col"

                  >

                    <v-img

                      class="profile-pic"

                      :src="pOffer.author\_info.avatar\_path"

                      width="32"

                      height="32"

                    />

                  </v-col>

                  <v-col

                    class="author-col"

                  >

                    {{ pOffer.author\_info.username }}

                  </v-col>

                </v-row>

              </v-chip>

            </v-col>

            <v-col>

              <h3 class="subtitle-1">

                Phone:

              </h3>

              <v-chip>

                <span class="constacts-data">{{ pOffer.contact\_phone }}</span>

              </v-chip>

            </v-col>

            <v-col>

              <h3 class="subtitle-1">

                Email:

              </h3>

              <v-chip>

                <span class="constacts-data">{{ pOffer.author\_info.email }}</span>

              </v-chip>

            </v-col>

          </v-row>

        </v-container>

      </v-card-text>

    </v-card>

  </v-form>

</template>

<script>

import axios from 'axios'

export default {

  components: {

    DeleteOfferDialog: () => import('./essentials/DeleteOfferDialog'),

    ImageLightbox: () => import('./helpers/ImageLightbox')

  },

  props: {

    pCategories: {type: Array, default: null},

    pOffer: {type: Object, default: null}

  },

  data() {

    return {

      images: [],

      parts: [],

      // table data

      headers: [

        {

          text: 'Label',

          align: 'start',

          value: 'label'

        },

        {

          text: 'Category',

          value: 'category'

        }

      ],

      partsItems: []

    }

  },

  created() {

    this.getImages()

    this.getParts()

  },

  methods: {

    closeDialog () {

      this.$emit('closeShowDialog')

    },

    deleteOffer () {

      const config = {

        headers: {

          'Authorization': 'Bearer '+this.$auth.token()

        }

      }

      axios

        .post('http://127.0.0.1:8000/api/auth/offer/delete\_soft', { id : this.pOffer.id }, config)

        .then(res => {

          this.$emit('closeShowDialog')

        })

        .catch(err => {

        })

    },

    getCategoryName (categoryId) {

      const category = this.pCategories.find(category => { return category.id === categoryId })

      return category.label

    },

    getImages () {

      const req = 'http://127.0.0.1:8000/api/offer\_media/'+this.pOffer.id

      axios

        .get (req)

        .then ((result) => {

          result.data.photo\_path.forEach(element => {

            this.images.push(element)

          });

        })

        .catch((err) => {

          console.log(err)

        })

    },

    async getParts () {

      const req = 'http://127.0.0.1:8000/api/offer/'+this.pOffer.id

      axios

        .get (req)

        .then ((result) => {

          const {data:{data}} = result

          this.parts = data.parts

          this.getTableContents()

        })

        .catch((err) => {

          console.log(err)

        })

    },

    getTableContents () {

      for (const part of this.parts) {

        this.partsItems.push({

          'label': part.part,

          'category': this.getCategoryName(part.category)

        })

      }

    }

  },

}

</script>

<style scoped>

  .author-col {

    align-self: center;

  }

  .contacts {

    border: 1px solid lightgrey;

    border-radius: 5px;

    padding-bottom: 0;

  }

  .constacts-data {

    font-weight: bold;

  }

  .image {

    max-width: 200px;

    max-height: 200px;

  }

  .profile-pic {

    border-radius: 50%;

  }

</style>

**frontend/components/Topic.vue**

<template>

  <v-card

    outlined

  >

    <v-card-title>

      {{ pTopic.title }}

      <v-spacer />

      <v-dialog

        v-if="$auth.user().id && ($auth.user().id === pTopic.author\_data.id || $auth.user().role !== 'user')"

        v-model="showDeleteDialog"

      >

        <template

          #activator="{ on }"

        >

          <v-btn

            icon

            v-on="on"

            color="error"

            @click="showDeleteDialog = true"

          >

            <v-icon>mdi-delete-forever-outline</v-icon>

          </v-btn>

        </template>

        <v-card>

          <v-card-title>

            Attention!

          </v-card-title>

          <v-card-text>

            Would You really like to delete this topic with all its comments?

          </v-card-text>

          <v-card-actions>

            <v-btn

              width="50%"

              text

              color="primary"

              @click="deleteTopic()"

            >

              Confirm

            </v-btn>

            <v-btn

              width="50%"

              text

              color="error"

              @click="showDeleteDialog = false"

            >

              Cancel

            </v-btn>

          </v-card-actions>

        </v-card>

      </v-dialog>

      <v-btn

        icon

        @click="fetchComments()"

      >

        <v-icon>mdi-autorenew</v-icon>

      </v-btn>

      <v-btn

        icon

        @click="closeTopic()"

      >

        <v-icon>mdi-close</v-icon>

      </v-btn>

    </v-card-title>

    <v-card-subtitle>

      Added by: {{ pTopic.author\_data.username }}

      <v-spacer />

      Added on: {{ pTopic.created\_at }}

    </v-card-subtitle>

    <v-card-text

      class="font-weight-regular"

    >

      {{ pTopic.body }}

    </v-card-text>

    <v-card-text>

      <v-container

        v-if="$auth.user().id"

      >

        <v-form

          ref="comForm"

        >

          <v-textarea

            v-model="body"

            :rules="[rules.required]"

            :counter="600"

            rows="4"

            placeholder="Comment ..."

            label="Comment"

            outlined

          />

          <v-btn

            text

            @click="submitComment()"

          >

            Submit

          </v-btn>

        </v-form>

      </v-container>

      <v-container

        style="max-height: 700px; overflow: auto;"

        v-if="comments.length"

      >

        <v-card

          outlined

        >

          <v-card-text>

            <v-container>

              <v-row

                v-for="comment in comments"

                :key="comment.id"

              >

                <v-row>

                  <v-col

                    cols="3"

                  >

                    <v-row

                      class="profile-col"

                    >

                      <v-img

                        class="profile-pic"

                        width="32"

                        height="32"

                        :src="comment.author\_data.avatar\_path"

                      />

                    </v-row>

                    <v-row

                      class="profile-col"

                    >

                      {{ comment.author\_data.username }}

                    </v-row>

                  </v-col>

                  <v-col>

                    {{ comment.body }}

                  </v-col>

                </v-row>

                <v-row>

                  Added on: {{ comment.created\_at }}

                </v-row>

                <v-dialog

                  v-if="$auth.user().id && ($auth.user().id === comment.author\_id || $auth.user().role !== 'user')"

                  v-model="comment.dialog"

                >

                  <template

                    #activator="{ on }"

                  >

                    <v-btn

                      icon

                      v-on="on"

                      color="error"

                      @click="comment.dialog = true"

                    >

                      <v-icon>mdi-delete-forever-outline</v-icon>

                    </v-btn>

                  </template>

                  <v-card>

                    <v-card-title>

                      Attention!

                    </v-card-title>

                    <v-card-text>

                      Would You really like to delete this comment?

                    </v-card-text>

                    <v-card-actions>

                      <v-btn

                        width="50%"

                        text

                        color="primary"

                        @click="deleteComment(comment.id)"

                      >

                        Confirm

                      </v-btn>

                      <v-btn

                        width="50%"

                        text

                        color="error"

                        @click="comment.dialog = false"

                      >

                        Cancel

                      </v-btn>

                    </v-card-actions>

                  </v-card>

                </v-dialog>

              </v-row>

            </v-container>

          </v-card-text>

        </v-card>

      </v-container>

    </v-card-text>

  </v-card>

</template>

<script>

import axios from 'axios'

import external\_rules from '../plugins/rules/rules'

export default {

  props: {

    pTopic: {type: Object, default: null}

  },

  data () {

    return {

      comments: [],

      rules: external\_rules,

      body: null,

      showDeleteDialog: false

    }

  },

  created () {

    this.fetchComments()

  },

  methods: {

    closeTopic () {

      this.$emit('closeDialog')

    },

    deleteComment (commentId) {

      const config = {

        headers: {

          'Authorization': 'Bearer '+this.$auth.token(),

        }

      }

      axios

        .post('http://127.0.0.1:8000/api/auth/forum/comment/delete', { comment: commentId}, config)

        .then (res => {

          this.showDeleteCommentDialog = false

          this.fetchComments()

        })

        .catch ((err) => {

          console.log(err)

        })

    },

    deleteTopic () {

      const config = {

        headers: {

          'Authorization': 'Bearer '+this.$auth.token(),

        }

      }

      axios

        .post('http://127.0.0.1:8000/api/auth/forum/topic/delete', { topic: this.pTopic.id}, config)

        .then (res => {

          this.closeTopic()

        })

        .catch ((err) => {

          console.log(err)

        })

    },

    fetchComments () {

      this.comments = []

      const path = `http://127.0.0.1:8000/api/forum/comments?topic=${this.pTopic.id}`

      axios

        .get(path)

        .then(res => {

          const {data:{data}} = res

          if (data){

            this.comments = data

          }

          for (const comment of this.comments) {

            comment.dialog = false

          }

        })

        .catch(err => {

          console.log(err)

        })

    },

    submitComment () {

      if (this.$refs.comForm.validate()) {

        const commentData = new FormData()

        commentData.append('user\_id', this.$auth.user().id)

        commentData.append('body', this.body)

        commentData.append('topic\_id', this.pTopic.id)

        const config = {

          headers: {

            'Authorization': 'Bearer '+this.$auth.token(),

            'Content-Type': 'multipart/form-data'

          }

        }

        axios

          .post('http://127.0.0.1:8000/api/auth/forum/comment/add', commentData, config)

          .then (res => {

            this.$refs.comForm.reset()

            this.fetchComments()

          })

          .catch ((err) => {

            console.log(err)

          })

      }

    }

  }

}

</script>

<style scoped>

  .profile-col {

    place-content: center;

  }

  .profile-pic {

    border-radius: 50%;

    max-width: 32px;

  }

</style>

**frontend/components/TopicsDisplay.vue**

<template>

  <v-card

    v-if="topics.length && pCategory.id"

    outlined

  >

    <v-card-title

      v-for="topic in topics"

      :key="topic.id"

    >

      {{ topic.title }}

      <v-spacer />

      <span

        v-if="topic.author\_data.username"

        class="subtitle-1"

      >

        Author: {{ topic.author\_data.username }}

      </span>

      <v-dialog

        v-model="topic.dialog"

        persistent

        fullscreen

      >

        <template

          #activator="{ on }"

        >

          <v-btn

            v-on="on"

            @click="topic.dialog = true"

            text

          >

            Expand

          </v-btn>

        </template>

        <div

          style="background-color: white; height: 100%;"

        >

          <Topic

            :pTopic="topic"

            @closeDialog="closeDialog()"

          />

        </div>

      </v-dialog>

    </v-card-title>

  </v-card>

</template>

<script>

import axios from 'axios'

export default {

  components: {

    Topic: () => import('./Topic.vue')

  },

  props: {

    pCategory: {type: Object, default: null}

  },

  data () {

    return {

      topics: []

    }

  },

  watch: {

    pCategory: function () {

      this.fetchTopics()

    }

  },

  methods: {

    closeDialog () {

      this.topics.map(topic => {

        topic.dialog = false

      })

      this.fetchTopics()

    },

    fetchTopics () {

      this.topics = []

      const path = `http://127.0.0.1:8000/api/forum/topics?category=${this.pCategory.id}`

      axios

        .get(path)

        .then(res => {

          const {data:{data}} = res

          if (data) {

            this.topics = data

          }

          for (const topic of this.topics) {

            topic.dialog = false

          }

        })

        .catch(err => {

          console.log(err)

        })

    },

  }

}

</script>

**frontend/plugins/rules/rules.js**

const external\_rules = {

    required: v => !!v || 'This field is required',

    requiredItem: v => ((v && v=='male') || (v && v=='female')) || 'This item is required',

    length: {

        name: v => ((!v) || (v && v.length < 24)) || 'Name must be less than 24 characters',

        surname: v => ((!v) || (v && v.length < 24)) || 'Surname must be less than 24 characters',

        username: v => ((v && v.length >= 5) && (v && v.length < 17))  || 'Username must be more than 5 and less than 16 characters',

        newUsername: v => (!v || ((v && v.length >= 5) && (v && v.length < 17)) ) || 'Username must be more than 5 and less than 16 characters',

        password: v => ((v && v.length >= 8) && (v && v.length < 17))  || 'Password must be more than 8 and less than 16 characters',

        newPassword: v => (!v || ((v && v.length >= 8) && (v && v.length < 17)) ) || 'Password must be more than 8 and less than 16 characters'

    },

    textOnly: v => (!v || /^[a-zA-Z\_-]+$/.test(v)) || 'Field can contain letters only',

    email: v => /.+@.+\..+/.test(v) || 'Email must be valid',

    newEmail: v => (!v ||  /.+@.+\..+/.test(v) ) || 'Email must be valid'

};

export default external\_rules;

**frontend/plugins/fetchAuctions.js**

import axios from 'axios'

const fetchAuctions = async (path, type = 'active') => {

    return new Promise((resolve, reject) => {

        axios

        .get(path)

        .then(res => {

            const {data:{data}} = res;

            let auctions;

            if (!data) {

                return

            } else if (data.length) {

                auctions = data

            } else {

                auctions = [data]

            }

            let charityAuctions;

            let commercialAuctions;

            if (type === 'active') {

                charityAuctions = auctions.find(auction => {

                    return auction.type === 'charity'

                });

                commercialAuctions = auctions.find(auction => {

                    return auction.type === 'commercial'

                });

            } else {

                charityAuctions = auctions.filter(auction => {

                    return auction.type === 'charity'

                });

                commercialAuctions = auctions.filter(auction => {

                    return auction.type === 'commercial'

                });

            }

            resolve({

                charityAuctions,

                commercialAuctions

            });

        })

        .catch(err => {

            console.log(`Error fetching active auctions: ${err}`)

            reject({});

        });

    });

}

export default fetchAuctions;

**frontend/plugins/fetchFavoriteOffers.js**

import axios from 'axios'

const fetchFavoriteOffers = async (userId) => {

    return new Promise((resolve, reject) => {

        axios

            .get(`http://127.0.0.1:8000/api/offers/favorites/${userId}`)

            .then(res => {

                resolve(res.data);

            })

            .catch(err => {

                console.log(err);

                reject([]);

            })

    })

  }

export default fetchFavoriteOffers;

**frontend/plugins/insertAuctionParticipant.js**

import axios from 'axios'

const insertAuctionParticipant = async (token, userId, auctionId, price, response, isPaid) => {

    return new Promise((resolve, reject) => {

        const config = {

            headers: {

            'Authorization': 'Bearer '+token,

            'Content-Type': 'multipart/form-data'

            }

        };

        const participantData = new FormData();

        participantData.append('auction\_id', auctionId);

        participantData.append('user\_id', userId);

        participantData.append('amount', price);

        axios

            .post(

            'http://127.0.0.1:8000/api/auth/auction/addParticipant',

            participantData,

            config

            )

            .then(res => {

                switch (res.status) {

                    case 200:

                        response.status = 'Success'

                        response.message = `Successfull participation in the auction!`

                    break

                    case 500:

                        response.status = 'Error'

                        response.message = 'Server is not responging. Error code: 500'

                    break

                    default:

                        response.status = 'Hmm...'

                        response.message = 'Something went wrong...'

                    break

                }

                isPaid = true;

                resolve({

                    response,

                    isPaid

                });

            })

            .catch(err => {

                console.log(err);

                reject({});

            })

    });

}

export default insertAuctionParticipant;

**frontend/views/admin/ControlPanel.vue**

<template>

  <v-container>

    <v-row>

      <h2 class="headline">

        Logged as: {{ $auth.user().username }}, {{ $auth.user().email }}

      </h2>

      <v-spacer />

      <h2 class="headline">

        Role: {{ getTranslatedRole() }}

      </h2>

    </v-row>

    <v-row>

      <v-col

        cols="3"

      >

        <v-card

          outlined

        >

          <v-card-title>

            Menu

          </v-card-title>

          <v-list>

            <v-list-item-group

              mandatory

              v-model="menuSection"

            >

              <v-list-item

                v-for="(item, index) in menuItems"

                :key="index"

              >

                <v-list-item-icon>

                  <v-icon>

                    {{ item.icon }}

                  </v-icon>

                </v-list-item-icon>

                <v-list-item-content>

                  <v-list-item-title>

                    {{ item.label }}

                  </v-list-item-title>

                </v-list-item-content>

              </v-list-item>

            </v-list-item-group>

          </v-list>

        </v-card>

      </v-col>

      <v-col

        v-if="menuSection === 0"

      >

        <Auctions />

      </v-col>

      <v-col

        v-if="menuSection === 1"

      >

        <Offers />

      </v-col>

      <v-col

        v-if="menuSection === 2"

      >

        <Users />

      </v-col>

      <v-col

        v-if="menuSection === 3"

      >

        <Admins />

      </v-col>

    </v-row>

  </v-container>

</template>

<script>

export default {

  components: {

    Admins: () => import('../../components/admin/Admins'),

    Auctions: () => import('../../components/admin/Auctions'),

    Offers: () => import('../../components/admin/Offers'),

    Users: () => import('../../components/admin/Users')

  },

  data () {

    return {

      menuItems: [

        {

          label: 'Auctions',

          icon: 'mdi-currency-usd'

        },

        {

          label: 'Offers',

          icon: 'mdi-view-list'

        },

        {

          label: 'Users',

          icon: 'mdi-account-supervisor'

        }

      ],

      menuSection: null

    }

  },

  created () {

    this.checkIfSA()

  },

  methods: {

    getTranslatedRole () {

      switch (this.$auth.user().role) {

        case 'admin':

          return 'Administrator'

        case 'super\_admin':

          return 'Super User'

      }

    },

    checkIfSA () {

      if (this.$auth.user().role === 'super\_admin') {

        this.menuItems.push(

          {

            label: 'Admins',

            icon: 'mdi-account-box-multiple'

          }

        )

      }

    }

  }

}

</script>

**frontend/views/Auction.vue**

<template>

  <v-container

    class="wrapper"

  >

    <v-row>

      <v-col

        md="6"

      >

        <CharityAuction

          v-if="activeCharityAuction"

          :pAuction="activeCharityAuction"

          @updateAuction="fetchActiveAuctions()"

        />

        <v-container

          v-else

          class="no-active-auction"

        >

          <v-divider />

          <h3

            class="title font-weight-light"

          >

            No active charity auction at the moment

          </h3>

          <v-divider />

        </v-container>

      </v-col>

      <v-col

        md="6"

      >

        <CommercialAuction

          v-if="activeCommercialAuction"

          :pAuction="activeCommercialAuction"

          @updateAuction="fetchActiveAuctions()"

        />

        <v-container

          v-else

          class="no-active-auction"

        >

          <v-divider />

          <h3

            class="title font-weight-light"

          >

            No active commercial auction at the moment

          </h3>

          <v-divider />

        </v-container>

      </v-col>

    </v-row>

    <v-container

      class="history-dialog"

    >

      <v-dialog

        v-model="showHistoryDialog"

        style="width: 100%;"

        fullscreen

      >

        <template

          #activator="{ on }"

        >

          <v-btn

            class="history-btn"

            v-on="on"

            large

          >

            History

          </v-btn>

        </template>

        <v-container

          class="history-container"

        >

          <v-row>

            <div

              class="close-btn-container"

            >

              <v-btn

                class="close-btn"

                @click="showHistoryDialog = false"

                icon

                outlined

              >

                <v-icon>mdi-close</v-icon>

              </v-btn>

            </div>

          </v-row>

          <v-row>

            <AuctionHistory

              :pAuctionArray="charityAuctions"

              pType="charity"

            />

          </v-row>

          <v-row>

            <AuctionHistory

              :pAuctionArray="commercialAuctions"

              pType="commercial"

            />

          </v-row>

        </v-container>

      </v-dialog>

    </v-container>

  </v-container>

</template>

<script>

import fetchAuctions from '../plugins/fetchAuctions'

export default {

  components: {

    AuctionHistory: () => import('../components/AuctionHistory'),

    CharityAuction: () => import('../components/CharityAuction'),

    CommercialAuction: () => import('../components/CommercialAuction')

  },

  data () {

    return {

      activeCharityAuction: null,

      activeCommercialAuction: null,

      charityAuctions: [],

      commercialAuctions: [],

      showHistoryDialog: false

    }

  },

  watch: {

    charityAuctions: {

      deep: true,

      handler: function () {

        this.fetchActiveAuctions()

      }

    }

  },

  created () {

    this.fetchAuctions()

  },

  methods: {

    async fetchActiveAuctions() {

      fetchAuctions('http://127.0.0.1:8000/api/auctions')

        .then(res => {

          this.activeCharityAuction = res.charityAuctions

          this.activeCommercialAuction = res.commercialAuctions

        })

    },

    async fetchDismissedAuctions() {

      fetchAuctions('http://127.0.0.1:8000/api/auctions/dismissed', 'dismissed')

        .then(res => {

          this.charityAuctions = res.charityAuctions

          this.commercialAuctions = res.commercialAuctions

        })

    },

    fetchAuctions () {

      this.fetchActiveAuctions()

      this.fetchDismissedAuctions()

    }

  }

}

</script>

<style scoped>

  .close-btn {

    float: right;

    margin-right: 20px;

  }

  .close-btn-container {

    width: 100%;

  }

  .history-btn {

    position: fixed;

    margin: 0;

    width: 100%;

    left: 0;

    bottom: 0;

    border: 1px solid;

    border-radius: 25px 25px 0 0;

  }

  .history-container {

    height: 100%;

    min-width: 100%;

    background-color: white;

  }

  .history-dialog {

    text-align-last: end;

    position: absolute;

    top: 0;

    right: 0;

  }

  .no-active-auction {

    height: 70vh;

    vertical-align: middle;

    text-align: center;

    display: table-cell;

    vertical-align: middle;

  }

  .wrapper {

    margin-bottom: 100px;

  }

</style>

**frontend/views/Dashboard.vue**

<template>

  <v-app>

    <v-container>

      <v-row>

        <v-col lg="2">

          <v-card>

            <v-img

              class="mx-auto"

              height="200px"

              width="200px"

              :src="$auth.user().avatar\_path"

              @click="$refs.avatarInput.click()"

            />

            <input

              @change="handleUpload()"

              type="file"

              ref="avatarInput"

              style="display: none;"

            >

            <v-btn

              v-if="showAvatarUpdateButton"

              @click="uploadAvatar()"

              block

              depressed

            >

              Upload

            </v-btn>

            <v-card-title primary-title>

              <div>

                <h3 class="headline mb-0">

                  {{ $auth.user().username }}

                </h3>

                <div>

                  {{ $auth.user().name }} {{ $auth.user().surname }}

                </div>

              </div>

            </v-card-title>

            <v-card-actions>

              <v-dialog

                v-model="editDialog"

                max-width="800px"

              >

                <template #activator="{on}">

                  <v-btn

                    text

                    color="primary"

                    v-on="on"

                  >

                    Edit

                  </v-btn>

                </template>

                <v-card>

                  <v-card-title>

                    Edit profile

                    <v-spacer />

                    <v-btn

                      icon

                      @click="editDialog = false"

                    >

                      <v-icon>mdi-close</v-icon>

                    </v-btn>

                  </v-card-title>

                  <v-card-text>

                    <v-form ref="form">

                      <v-container>

                        <v-row class="align-center">

                          <v-col lg="2">

                            <v-subheader

                              color="black"

                              class="align-center title"

                            >

                              Name

                            </v-subheader>

                          </v-col>

                          <v-col lg="4">

                            <v-text-field

                              v-model="name"

                              :counter="24"

                              :rules="[rules.textOnly, rules.length.name]"

                            />

                          </v-col>

                          <v-col lg="2">

                            <v-subheader

                              color="black"

                              class="title"

                            >

                              Surname

                            </v-subheader>

                          </v-col>

                          <v-col lg="4">

                            <v-text-field

                              v-model="surname"

                              :counter="24"

                              :rules="[rules.textOnly, rules.length.surname]"

                            />

                          </v-col>

                        </v-row>

                        <v-row class="align-center">

                          <v-col lg="2">

                            <v-subheader

                              color="black"

                              class="title"

                            >

                              Username

                            </v-subheader>

                          </v-col>

                          <v-col lg="4">

                            <v-text-field

                              v-model="username"

                              :counter="16"

                              :rules="[rules.length.newUsername]"

                            />

                          </v-col>

                        </v-row>

                        <v-row>

                          <v-expand-transition>

                            <p

                              v-show="errors.username"

                              style="color: red;"

                              class="subtitle-2 pl-5"

                            >

                              <v-icon

                                color="red"

                                size="17px"

                                class="pr-1"

                              >

                                mdi-alert

                              </v-icon>{{ errors.username ? errors.username[0] : '' }}

                            </p>

                          </v-expand-transition>

                        </v-row>

                        <v-row class="align-center">

                          <v-col lg="2">

                            <v-subheader

                              color="black"

                              class="title"

                            >

                              Email

                            </v-subheader>

                          </v-col>

                          <v-col lg="4">

                            <v-text-field

                              v-model="email"

                              :rules="[rules.newEmail]"

                            />

                          </v-col>

                          <v-col>

                            <v-radio-group

                              v-model="gender"

                              :label="'Gender'"

                              :rules="[rules.requiredItem]"

                              class="pl-4 pt-0 mt-0"

                            >

                              <v-radio

                                color="primary"

                                :label="'Male'"

                                :value="'male'"

                              />

                              <v-radio

                                color="primary"

                                :label="'Female'"

                                :value="'female'"

                              />

                            </v-radio-group>

                          </v-col>

                        </v-row>

                        <v-row>

                          <v-expand-transition>

                            <p

                              v-show="errors.email"

                              style="color: red;"

                              class="subtitle-2 pl-5"

                            >

                              <v-icon

                                color="red"

                                size="17px"

                                class="pr-1"

                              >

                                mdi-alert

                              </v-icon>{{ errors.email ? errors.email[0] : '' }}

                            </p>

                          </v-expand-transition>

                        </v-row>

                        <v-row class="justify-center">

                          <v-btn

                            text

                            color="primary"

                            @click="changePass = !changePass"

                          >

                            Change Password

                          </v-btn>

                        </v-row>

                        <v-expand-transition>

                          <v-container>

                            <v-text-field

                              v-show="changePass"

                              v-model="password"

                              prepend-icon="mdi-lock"

                              type="password"

                              label="New Password"

                              :counter="16"

                              :rules="[rules.length.newPassword]"

                            />

                            <v-text-field

                              v-show="changePass"

                              v-model="password\_confirmation"

                              prepend-icon="mdi-lock"

                              label="Confirm Password"

                              type="password"

                              :rules="[scoped\_rules.passwordConfirmation]"

                            />

                          </v-container>

                        </v-expand-transition>

                        <v-card-actions>

                          <v-btn

                            :disabled="isLoading"

                            :loading="isLoading"

                            @click="submit()"

                          >

                            Submit

                          </v-btn>

                        </v-card-actions>

                      </v-container>

                    </v-form>

                  </v-card-text>

                </v-card>

              </v-dialog>

              <v-dialog

                v-model="deleteDialog"

                max-width="800px"

              >

                <template #activator="{on}">

                  <v-btn

                    text

                    color="error"

                    v-on="on"

                  >

                    Delete

                  </v-btn>

                </template>

                <v-card>

                  <v-card-title>Delete Account</v-card-title>

                  <v-card-text>

                    <p>Attention: You are deleting Your account. This cannot be undone. Would You like to proceed?</p>

                  </v-card-text>

                  <v-card-actions>

                    <v-btn

                      color="error"

                      @click="deleteAcc()"

                    >

                      Delete

                    </v-btn>

                    <v-btn

                      color="primary"

                      @click="deleteDialog = !deleteDialog"

                    >

                      Cancel

                    </v-btn>

                  </v-card-actions>

                </v-card>

              </v-dialog>

            </v-card-actions>

          </v-card>

        </v-col>

        <v-col>

          <v-card>

            <v-card-title primary-title>

              <v-container

                class="offers-title"

              >

                <v-row>

                  <h3 class="headline mb-0">

                    Placed Offers

                  </h3>

                  <v-spacer />

                  <v-btn

                    icon

                    @click="fetchProfileOffers(userOffersPath)"

                  >

                    <v-icon>mdi-autorenew</v-icon>

                  </v-btn>

                </v-row>

              </v-container>

            </v-card-title>

            <v-divider />

            <v-card-text>

              <v-tabs

                fixed-tabs

                v-model="offerTabs"

                class="offers-tabs"

              >

                <v-tab>

                  Your offers

                </v-tab>

                <v-tab>

                  Bookmarked offers

                </v-tab>

                <v-tabs-items

                  class="user-offers"

                  v-model="offerTabs"

                >

                  <v-tab-item>

                    <v-card

                      v-if="offers"

                    >

                      <ConnectionError

                        v-if="errorCode"

                        :errorCode="errorCode"

                      />

                      <div

                        v-for="offer in offers"

                        :key="offer.id"

                      >

                        <OffersDisplay

                          :pOffer="offer"

                          :pCategories="categories"

                          :pIsTiled="true"

                        />

                      </div>

                    </v-card>

                  </v-tab-item>

                  <v-tab-item>

                    <v-card

                      v-if="bookmarkedOffers"

                    >

                      <ConnectionError

                        v-if="errorCode"

                        :errorCode="errorCode"

                      />

                      <div

                        v-for="offer in bookmarkedOffers"

                        :key="offer.id"

                      >

                        <OffersDisplay

                          :pOffer="offer"

                          :pCategories="categories"

                          :pIsTiled="true"

                        />

                      </div>

                    </v-card>

                  </v-tab-item>

                </v-tabs-items>

              </v-tabs>

            </v-card-text>

          </v-card>

          <v-card

            class="auction-card"

          >

            <v-card-title>

              <v-container

                class="offers-title"

              >

                <v-row>

                  <h3 class="headline mb-0">

                    Participated Auctions

                  </h3>

                </v-row>

              </v-container>

            </v-card-title>

            <v-divider />

            <v-card-subtitle>

              Active

            </v-card-subtitle>

            <div

              v-if="activeCharityAuction"

            >

              <AuctionHistory

                :pAuctionArray="[activeCharityAuction]"

                :pType="'charity'"

              />

            </div>

            <div

              v-if="activeCommercialAuction"

            >

              <AuctionHistory

                :pAuctionArray="[activeCommercialAuction]"

                :pType="'commercial'"

              />

            </div>

            <v-card-subtitle>

              Dismissed

            </v-card-subtitle>

            <AuctionHistory

              v-if="charityAuctions.length"

              :pAuctionArray="charityAuctions"

              :pType="'charity'"

            />

            <AuctionHistory

              v-if="commercialAuctions.length"

              :pAuctionArray="commercialAuctions"

              :pType="'commercial'"

            />

          </v-card>

        </v-col>

      </v-row>

    </v-container>

  </v-app>

</template>

<script>

import axios from 'axios'

import external\_rules from '@/plugins/rules/rules.js'

import fetchAuctions from '@/plugins/fetchAuctions.js'

export default {

  components: {

    AuctionHistory: () => import('../components/AuctionHistory'),

    ConnectionError: () => import('../components/helpers/ConnectionError'),

    OffersDisplay: () => import('../components/OffersDisplay')

  },

  data() {

    return {

      email: this.$auth.user().email,

      username: this.$auth.user().username,

      name: this.$auth.user().name,

      surname: this.$auth.user().surname,

      gender: this.$auth.user().gender,

      password: null,

      password\_confirmation: null,

      scoped\_rules: {

        passwordConfirmation: v => ((!v && !this.password) || (v && v == this.password) ) || 'Fields do not match',

      },

      rules: external\_rules,

      editDialog: false,

      deleteDialog: false,

      changePass: false,

      userObject: {},

      errors: {},

      errorCode: null,

      isLoading: false,

      isError: false,

      offerTabs: null,

      offers: [],

      categories: [],

      isEmptySet: false,

      userOffersPath: 'http://127.0.0.1:8000/api/auth/offer/get',

      avatarFile: null,

      showAvatarUpdateButton: false,

      bookmarkedOffers: null,

      activeCharityAuction: null,

      activeCommercialAuction: null,

      charityAuctions: [],

      commercialAuctions: []

    }

  },

  computed: {

    favoriteOffers: function () {

      return this.$store.getters['favoriteOffers']

    }

  },

  created() {

    this.fetchProfileOffers(this.userOffersPath)

    this.fetchOffers()

    this.fetchCategories()

    this.fetchUserAuctions()

  },

  methods: {

    async fetchUserAuctions() {

      const userData = new FormData()

      userData.append('user\_id', this.$auth.user().id)

      const config = {

        headers: {

          'Authorization': 'Bearer '+this.$auth.token(),

          'Content-Type': 'multipart/form-data'

        }

      }

      axios

        .post(`http://127.0.0.1:8000/api/auth/auctions/`, userData, config)

        .then(res => {

          const {data} = res

          const activeAutcions = data[0]

          const dismissedAuctions = data[1]

          if (activeAutcions.length) {

            for (const auction of activeAutcions) {

              switch (auction.type) {

                case 'charity':

                  this.activeCharityAuction = auction

                  break

                case 'commercial':

                  this.activeCommercialAuction = auction

                  break

              }

            }

          }

          if (dismissedAuctions.length) {

            for (const auction of dismissedAuctions) {

              switch (auction.type) {

                case 'charity':

                  this.charityAuctions.push(auction)

                  break

                case 'commercial':

                  this.commercialAuctions.push(auction)

                  break

              }

            }

          }

        })

        .catch(err => {

            console.log(`Error fetching user auctions: ${err}`)

        });

    },

    submit () {

      if (this.$refs.form.validate()) {

        this.update();

      }

    },

    update () {

      this.errors = {}

      this.isError = false

      this.isLoading = true

      this.userObject = {

        id: this.$auth.user().id,

        email: this.email,

        username: this.username,

        name: this.name,

        surname: this.surname,

        gender: this.gender,

        password: this.password

      }

      axios.post('http://127.0.0.1:8000/api/auth/userUpdate',

        {

          id: this.$auth.user().id,

          email: this.email,

          username: this.username,

          name: this.name,

          surname: this.surname,

          gender: this.gender,

          password: this.password

        },

          {

            headers: {

              'Authorization': 'Bearer '+this.$auth.token(),

            }

        }).then(() => {

          this.$auth.user(this.userObject)

          this.editDialog = false

          this.isLoading = false

        }).catch( err => {

          if (err.response.status != 422){

            this.isError = true

            this.errorCode = err.response.status

          }

          this.isLoading = false

          this.error = err.response.data.error

          this.errors = err.response.data.errors || {}

        });

    },

    deleteAcc () {

      this.$auth.logout({

        success: function () {

          axios

            .delete('http://127.0.0.1:8000/api/auth/userDelete', this.$auth.user().id)

            .catch( err => {

              console.log('error: ', err)

            })

          this.changeAuthState()

        },

        error: function () {

          console.log("Something went wrong with Your logout!")

        },

      })

    },

    fetchCategories () {

      const path = 'http://127.0.0.1:8000/api/offer\_categories'

      axios

        .get(path)

        .then(res => {

          const {data:{data}} = res

          if (data && !data.length){

            this.categories = [data]

          } else {

            this.categories = data

          }

        })

        .catch(err => {

          this.errorCode = err.response.status

        })

    },

    fetchOffers () {

      axios

        .get('http://127.0.0.1:8000/api/offers/getAll')

        .then(res => {

          const {data:{data}} = res

          let newBookmarks = []

          this.favoriteOffers.map(favoriteOffer => {

            data.map(offer => {

              if (favoriteOffer.offer\_id === offer.id && favoriteOffer.user\_id === this.$auth.user().id) {

                newBookmarks.push(offer)

              }

            })

          })

          this.bookmarkedOffers = newBookmarks

        })

        .catch(err => {

          this.errorCode = err.response

        })

    },

    fetchProfileOffers (path) {

      const offerData = new FormData()

      offerData.append('author\_id', this.$auth.user().id)

      const config = {

        headers: {

          'Authorization': 'Bearer '+this.$auth.token(),

          'Content-Type': 'multipart/form-data'

        }

      }

      axios

        .post(path, offerData, config)

        .then(res => {

          const {data:{data}} = res

          if (!data)

            this.isEmptySet = true

          else

            this.isEmptySet = false

          this.offers = data

        })

        .catch(err => {

          this.errorCode = err.response

        })

    },

    handleUpload () {

      if (this.$refs.avatarInput) {

        this.avatarFile = this.$refs.avatarInput.files[0]

        if (this.avatarFile) {

          this.showAvatarUpdateButton = true

        }

      }

    },

    uploadAvatar () {

      this.showAvatarUpdateButton = false

      const avatarInfo = new FormData

      avatarInfo.append('user\_id', this.$auth.user().id)

      avatarInfo.append('avatar', this.avatarFile)

      const config = {

        headers: {

          'Authorization': 'Bearer '+this.$auth.token(),

          'Content-Type': 'multipart/form-data'

        }

      }

      axios

        .post('http://127.0.0.1:8000/api/auth/userUpdateAvatar', avatarInfo, config)

        .then(res => {

          this.$auth.fetch()

        })

        .catch(err => {

          console.log(err)

        })

    },

  }

}

</script>

<style scoped>

  .auction-card {

    margin-top: 30px;

  }

  .offers-tabs {

    min-width: 100%;

  }

  .offers-title {

    padding-bottom: 0;

    padding-top: 0;

  }

  .user-offers {

    max-height: 250px;

    overflow: auto;

  }

</style>

**frontend/views/Forum.vue**

<template>

  <v-container>

    <v-row>

      <v-col

        cols="3"

      >

        <v-card

          outlined

        >

          <v-card-title>

            Categories

          </v-card-title>

          <v-list>

            <v-list-item-group

              v-model="categorySection"

            >

              <v-list-item

                v-for="category in categories"

                :key="category.id"

                @click="selectedCategory = category"

              >

                <v-list-item-content>

                  <v-list-item-title>

                    {{ category.label }}

                  </v-list-item-title>

                </v-list-item-content>

              </v-list-item>

            </v-list-item-group>

          </v-list>

        </v-card>

      </v-col>

      <v-col

        v-if="selectedCategory"

      >

        <v-row

          v-if="$auth.user().id"

        >

          <v-dialog

            v-model="addTopicDialog"

          >

            <template

              #activator="{ on }"

            >

              <v-btn

                text

                block

                v-on="on"

              >

                Create Topic

              </v-btn>

            </template>

            <v-form

              ref="topForm"

              v-if="!response"

            >

              <v-card>

                <v-card-title>

                  Add Topic

                </v-card-title>

                <v-card-text>

                  <v-text-field

                    v-model="title"

                    :rules="[rules.required]"

                    :counter="32"

                    placeholder="Topic Title ..."

                    label="Title"

                    outlined

                  />

                  <v-textarea

                    v-model="body"

                    :rules="[rules.required]"

                    :counter="600"

                    rows="3"

                    placeholder="Topic Description ..."

                    label="Description"

                    outlined

                  />

                </v-card-text>

                <v-card-actions>

                  <v-btn

                    color="primary"

                    text

                    width="50%"

                    @click="submit()"

                  >

                    Submit

                  </v-btn>

                  <v-btn

                    color="error"

                    text

                    width="50%"

                    @click="closeDialog()"

                  >

                    Cancel

                  </v-btn>

                </v-card-actions>

              </v-card>

            </v-form>

            <v-card

              v-if="response"

            >

              <v-card-title>

                {{ response }}

              </v-card-title>

              <v-card-actions>

                <v-btn

                  color="success"

                  text

                  block

                  @click="closeDialog(true)"

                >

                  Ok

                </v-btn>

              </v-card-actions>

            </v-card>

          </v-dialog>

        </v-row>

        <v-divider />

        <v-row

          style="width: 100%; background-color: white;"

        >

          <TopicsDisplay

            style="width: 100%;"

            :pCategory="selectedCategory"

          />

        </v-row>

      </v-col>

      <v-col

        v-if="!selectedCategory"

      >

        <h1

          class="display-3 font-weight-light"

        >

          Retro Forum

        </h1>

        <v-divider />

      </v-col>

    </v-row>

  </v-container>

</template>

<script>

import axios from 'axios'

import external\_rules from '../plugins/rules/rules'

export default {

  components: {

    TopicsDisplay: () => import('../components/TopicsDisplay')

  },

  data () {

    return {

      categories: [],

      categorySection: null,

      selectedCategory: null,

      addTopicDialog: false,

      rules: external\_rules,

      title: null,

      body: null,

      response: null

    }

  },

  created () {

    this.fetchCategories()

  },

  methods: {

    closeDialog (afterSubmit = false) {

      this.body = null

      this.title = null

      if (!afterSubmit) {

        this.$refs.topForm.reset()

      }

      this.addTopicDialog = false

      this.response = null

    },

    fetchCategories () {

      const path = 'http://127.0.0.1:8000/api/forum/categories'

      axios

        .get(path)

        .then(res => {

          const {data:{data}} = res

          if (data && !data.length){

            this.categories = [data]

          } else {

            this.categories = data

          }

        })

        .catch(err => {

          console.log(err)

        })

    },

    submit () {

      if (this.$refs.topForm.validate()) {

        const topicData = new FormData()

        topicData.append('user\_id', this.$auth.user().id)

        topicData.append('title', this.title)

        topicData.append('body', this.body)

        topicData.append('category\_id', this.selectedCategory.id)

        const config = {

          headers: {

            'Authorization': 'Bearer '+this.$auth.token(),

            'Content-Type': 'multipart/form-data'

          }

        }

        axios

          .post('http://127.0.0.1:8000/api/auth/forum/topic/add', topicData, config)

          .then (res => {

            this.response = 'Your topic will be created!'

            this.fetchCategories()

          })

          .catch ((err) => {

            console.log(err)

            this.response = 'Something went wrong!'

          })

      }

    }

  }

}

</script>

**frontend/views/Home.vue**

<template>

  <v-container>

    <h1

      class="display-1 font-weight-light"

    >

      Welcome to the \_

    </h1>

    <h1

      class="display-4 font-weight-light"

    >

      Retro Platfrom

    </h1>

    <v-divider />

    <h3

      class="subtitling caption font-weight-medium"

    >

      Place for the car enthusiasts

    </h3>

    <v-container

      class="slogan-container"

    >

      <div

        class="display-3"

        style="border-bottom: 1px solid;"

      >

        Fast and Reliable.

      </div>

    </v-container>

  </v-container>

</template>

<script>

export default {

}

</script>

<style scoped>

  .slogan-container {

    width: 100vw;

    height: 50vh;

    text-align: center;

    display: table-cell;

    vertical-align: middle;

  }

  .subtitling {

    text-align: center;

  }

</style>

**frontend/views/Offers.vue**

<template>

  <v-app>

    <v-container>

      <v-row>

        <v-col

          cols="3"

        >

          <OffersFilters

            :pCategories="categories"

            @updateFilters="offers\_update()"

          />

        </v-col>

        <v-col

          cols="9"

        >

          <v-card

            outlined

          >

            <v-row>

              <v-card-title class="offers-title display-1">

                Offer list

              </v-card-title>

              <v-spacer />

              <v-card-actions class="pr-6">

                <v-btn

                  icon

                  class="fetchOffersBtn"

                  @click="offers\_update()"

                >

                  <v-icon>mdi-autorenew</v-icon>

                </v-btn>

                <v-dialog

                  v-model="addOfferDialog"

                  max-width="1500px"

                >

                  <template #activator="{on}">

                    <v-btn

                      depressed

                      v-if="getAuthState()"

                      v-on="on"

                    >

                      Add Offer

                    </v-btn>

                  </template>

                  <AddOffer

                    :pCategories="categories"

                    @closeAddOfferDialog="closeAddOfferDialog()"

                  />

                </v-dialog>

              </v-card-actions>

            </v-row>

            <v-divider />

            <v-container

              id="offers"

              pa-0

            >

              <v-card v-if="isError">

                <ConnectionError :error-code="errorCode" />

              </v-card>

              <v-container

                v-if="isLoading"

                id="loading"

                class="loading"

              >

                <Loading />

              </v-container>

              <v-container

                v-if="!isLoading"

                class="offers-wrapper"

              >

                <v-row

                  v-if="isEmptySet"

                  style="margin-top: 25%;"

                  align="center"

                  justify="center"

                >

                  <v-col lg="3">

                    <v-divider />

                  </v-col>

                  <v-col lg="3">

                    <p

                      style="text-align: center;"

                      align-content-center

                    >

                      Nothing to show

                    </p>

                  </v-col>

                  <v-col lg="3">

                    <v-divider />

                  </v-col>

                </v-row>

                <v-container

                  v-for="item in offers"

                  :key="item.id"

                  justify-space-around

                >

                  <OffersDisplay

                    :pOffer="item"

                    :pCategories="categories"

                    @updateOffers="offers\_update()"

                  />

                </v-container>

              </v-container>

            </v-container>

            <v-divider />

            <v-pagination

              v-model="page\_number"

              style="margin: 15px;"

              :length="page\_count"

              :total-visible="7"

              :disabled="isPaginationDisabled"

              @input="offers\_update"

              @next="offers\_update"

              @previous="offers\_update"

            />

          </v-card>

        </v-col>

      </v-row>

    </v-container>

  </v-app>

</template>

<script>

import AddOffer from '@/components/AddOffer.vue'

import OffersDisplay from '@/components/OffersDisplay.vue'

import OffersFilters from '@/components/OffersFilters.vue'

import Loading from '@/components/helpers/Loading.vue'

import ConnectionError from '@/components/helpers/ConnectionError.vue'

import axios from 'axios'

import { mapGetters } from 'vuex'

export default {

  name: 'Offers',

  components: {

    AddOffer,

    OffersDisplay,

    OffersFilters,

    Loading,

    ConnectionError

  },

  data() {

    return {

      offers: [],

      categories: [],

      page\_count: null,

      page\_number: 1,

      offersFilters: {},

      isPaginationDisabled: false,

      isLoading: true,

      isError: false,

      isEmptySet: false,

      addOfferDialog: false,

      errorCode: null

    }

  },

  computed: {

    favoriteOffers: function () {

      return this.$store.getters['favoriteOffers']

    }

  },

  created() {

    this.fetchOffers('http://127.0.0.1:8000/api/offers')

    this.fetchCategories()

  },

  methods: {

    ...mapGetters ([

      'getAuthState',

    ]),

    closeShowDialog (offerId) {

      const buffer = this.offers

      buffer.map(offer => {

        if (offer.id === offerId) {

          offer.showOfferDialog = false

        }

      })

      this.offers = buffer

    },

    fetchCategories () {

      const path = 'http://127.0.0.1:8000/api/offer\_categories'

      axios

        .get(path)

        .then(res => {

          const {data:{data}} = res

          if (data && !data.length){

            this.categories = [data]

          } else {

            this.categories = data

          }

        })

        .catch(err => {

          this.errorCode = err.response.status

          this.responseError()

        })

    },

    fetchOffers (path) {

      axios

        .get(path)

        .then(res => {

          const {data:{data}} = res

          if (!data.length)

            this.isEmptySet = true

          else

            this.isEmptySet = false

          this.setOffers(data)

          this.isLoading = false

          const {data:{meta}} = res

          this.isPaginationDisabled = false

          this.setPageCount(meta.last\_page)

        })

        .catch(err => {

          console.log(err);

          this.errorCode = err

          this.responseError()

        })

    },

    responseError () {

      this.isLoading = false;

      this.isError = true;

      return true;

    },

    setOffers (data) {

      this.offers = data

      if (this.$auth.user().id) {

        this.offers.map(offer => {

          offer.showOfferDialog = false

          if (offer.author\_id !== this.$auth.user().id) {

            offer.canBeFavorited = true

          } else {

            offer.canBeFavorited = false

          }

          if (this.favoriteOffers.find(record => record.offer\_id === offer.id && record.user\_id === this.$auth.user().id)) {

            offer.isFavorite = true

          } else {

            offer.isFavorite = false

          }

        })

      }

    },

    setPageCount (data) {

      this.page\_count = data;

    },

    getPageNumber () {

      return this.page\_number;

    },

    getOffers () {

      return this.offers;

    },

    getPageCount () {

      return this.page\_count;

    },

    offers\_update () {

      this.offersFilters = this.$store.getters['offersFilters']

      this.isPaginationDisabled = true;

      this.isLoading = true;

      this.isError = false;

      const pagePath =

        `http://127.0.0.1:8000/api/offers?page=${this.page\_number}&title=${this.offersFilters.title || ''}&description=${this.offersFilters.description || ''}&parts=${this.offersFilters.parts || ''}&category=${this.offersFilters.category || ''}&date=${this.offersFilters.date || ''}`

      this.fetchOffers(pagePath);

    },

    closeAddOfferDialog () {

      this.offers\_update()

      this.addOfferDialog = false

    },

    showMore (itemID) {

      return this.offers[itemID];

    }

  },

}

</script>

<style scoped>

  #offers {

    margin: 0 auto;

    display: block;

    height: 690px;

    /\* max-width: 50%; \*/

    overflow: auto;

  }

  .loading {

    margin: 0 auto;

    padding-top: 25%;

    padding-left: 49%;

  }

  .err {

    padding-top: 12em;

    line-height: 2em;

  }

  #offers h2 {

    text-align: left;

  }

  img {

    padding-left: 12px;

    max-width: 300px;

  }

  .offers {

    overflow: hidden;

    /\* border: solid 4px; \*/

  }

  .offers h2 {

    position: relative;

    /\* border: solid 1px red; \*/

  }

  .offers p {

    position: relative;

    width: 50%;

    float: left;

    /\* border: solid 1px red; \*/

    margin-bottom: 20px;

    text-align: justify;

  }

  .offers button {

    margin-top: 20px;

    margin-right: 50px;

    padding: 20px;

    float: right;

  }

  .offers-title {

    margin-left: 10px;

  }

  .fetchOffersBtn {

    margin-right: 15px;

  }

</style>

**frontend/views/Registration.vue**

<template>

  <v-app>

    <v-container

      class="mx-auto"

      fill-height

    >

      <ConnectionError

        v-if="isError"

        width="30px"

        :error-code="errorCode"

      />

      <v-card

        v-if="!isError"

        width="500px"

        class="mx-auto"

      >

        <v-card-title>Register new account</v-card-title>

        <v-card-text>

          <v-form ref="form">

            <v-text-field

              v-model="name"

              prepend-icon="mdi-account-circle-outline"

              label="Name"

              :counter="24"

              :rules="[rules.textOnly, rules.length.name]"

            />

            <v-text-field

              v-model="surname"

              prepend-icon="mdi-account-circle-outline"

              label="Surname"

              :counter="24"

              :rules="[rules.textOnly, rules.length.surname]"

            />

            <v-text-field

              v-model="username"

              prepend-icon="mdi-account-circle"

              label="Username"

              :counter="16"

              :rules="[rules.required, rules.length.username]"

            />

            <v-expand-transition>

              <p

                v-show="errors.username"

                style="color: red;"

                class="subtitle-2 pl-5"

              >

                <v-icon

                  color="red"

                  size="17px"

                  class="pr-1"

                >

                  mdi-alert

                </v-icon>{{ errors.username ? errors.username[0] : '' }}

              </p>

            </v-expand-transition>

            <v-text-field

              v-model="email"

              prepend-icon="mdi-email"

              label="Email"

              :rules="[rules.required, rules.email]"

            />

            <v-expand-transition>

              <p

                v-show="errors.email"

                style="color: red;"

                class="subtitle-2 pl-5"

              >

                <v-icon

                  color="red"

                  size="17px"

                  class="pr-1"

                >

                  mdi-alert

                </v-icon>{{ errors.email ? errors.email[0] : '' }}

              </p>

            </v-expand-transition>

            <v-text-field

              v-model="password"

              prepend-icon="mdi-lock"

              type="password"

              label="Password"

              :counter="16"

              :rules="[rules.required, rules.length.password]"

            />

            <v-text-field

              v-model="password\_confirmation"

              prepend-icon="mdi-lock"

              label="Confirm Password"

              type="password"

              :rules="[rules.required, scoped\_rules.passwordConfirmation]"

            />

            <v-radio-group

              v-model="gender"

              :label="'Gender'"

              :rules="[rules.requiredItem]"

              class="pl-4 pt-0 mt-0"

            >

              <v-radio

                color="primary"

                :label="'Male'"

                :value="'male'"

              />

              <v-radio

                color="primary"

                :label="'Female'"

                :value="'female'"

              />

            </v-radio-group>

            <v-card-actions>

              <v-btn

                v-if="!isLoading"

                @click="submit()"

              >

                Submit

              </v-btn>

              <v-spacer />

            </v-card-actions>

            <v-card-text v-if="isLoading">

              <v-container class="mx-auto loading">

                <Loading />

              </v-container>

            </v-card-text>

          </v-form>

        </v-card-text>

      </v-card>

    </v-container>

  </v-app>

</template>

<script>

import Loading from '@/components/helpers/Loading.vue'

import ConnectionError from '@/components/helpers/ConnectionError.vue'

import external\_rules from '@/plugins/rules/rules.js'

export default {

  name: 'Signup',

  components: {

    ConnectionError,

    Loading

  },

  data() {

    return {

      name: null,

      surname: null,

      gender: null,

      username: null,

      email: null,

      password: null,

      password\_confirmation: null,

      rules: external\_rules,

      scoped\_rules: {

        passwordConfirmation: v => (v && v == this.password) || 'Fields do not match'

      },

      errors: {},

      errorCode: null,

      isLoading: false,

      isError: false

    }

  },

  methods: {

    autofill() {

      this.name = 'gdjhghjkgf',

      this.surname = 'gdfgdfg',

      this.username = 'pdgdsfgdsg',

      this.email = 'gfhgfh@fdggh.com',

      this.password = '123456789',

      this.password\_confirmation = '123456789'

    },

    submit() {

      if (this.$refs.form.validate()) {

        this.register();

      }

    },

    register() {

      this.errors = {}

      this.isError = false

      this.isLoading = true

      const that = this

      this.$auth.register({

        data: {

          name: that.name,

          surname: that.surname,

          username: that.username,

          gender: that.gender,

          email: that.email,

          password: that.password,

          password\_confirmation: that.password\_confirmation

        },

        success: function () {

          this.$router.push({name: 'offers', params: {successRegistrationRedirect: true}})

        },

        error: function (res) {

          console.log(res)

          if (res.response.status != 422){

            that.isError = true

            that.errorCode = res.response.status

          }

          that.isLoading = false

          that.error = res.response.data.error

          that.errors = res.response.data.errors || {}

        }

      })

    }

  }

}

</script>

<style scoped>

  .loading {

    margin: 0 auto;

    height: 30px;

    width: 30px;

  }

</style>

**frontend/App.vue**

<template>

  <v-app v-if="$auth.ready()">

    <v-tabs

      v-model="activeTab"

      fixed-tabs

    >

      <v-tab to="/">

        Home

      </v-tab>

      <v-tab to="/offers">

        Offers

      </v-tab>

      <v-tab to="/auction">

        Auction

      </v-tab>

      <v-tab

        v-if="!getAuthState()"

        to="/signup"

      >

        Sign Up

      </v-tab>

      <v-tab to="/forum">

        Forum

      </v-tab>

      <v-tab

        v-if="$auth.check()"

        to="/dashboard"

      >

        Dashboard

      </v-tab>

      <v-tab

        v-if="authAdmin"

        to="/admin"

      >

        Control Panel

      </v-tab>

      <v-divider :vertical="true" />

      <v-menu

        v-if="!getAuthState()"

        :close-on-click="true"

        :close-on-content-click="getLoginButton()"

        :offset-x="true"

        :offset-y="true"

      >

        <template #activator="{ on }">

          <v-btn

            style="height: 100%; width: 100%; border-radius: 0"

            max-width="10%"

            fill-height

            text

            v-on="on"

          >

            Log In

          </v-btn>

        </template>

        <v-list>

          <v-list-item>

            <Login />

          </v-list-item>

        </v-list>

      </v-menu>

      <v-tooltip

        bottom

        v-if="getAuthState()"

      >

        <template

          #activator="{ on }"

        >

          <v-btn

            style="height: 100%; width: 100%; border-radius: 0"

            max-width="10%"

            fill-height

            text

            :loading="isLoading"

            :disabled="isLoading"

            @click="logout"

            v-on="on"

          >

            Log Out

          </v-btn>

        </template>

        <span>{{ $auth.user().username }}</span>

      </v-tooltip>

      <v-tab-item id="/">

        <v-divider />

        <router-view v-if="activeTab === '/'" />

      </v-tab-item>

      <v-tab-item id="/offers">

        <v-divider />

        <router-view v-if="activeTab === '/offers'" />

      </v-tab-item>

      <v-tab-item id="/auction">

        <v-divider />

        <router-view v-if="activeTab === '/auction'" />

      </v-tab-item>

      <v-tab-item id="/signup">

        <v-divider />

        <router-view v-if="activeTab === '/signup'" />

      </v-tab-item>

      <v-tab-item id="/forum">

        <v-divider />

        <router-view v-if="activeTab === '/forum'" />

      </v-tab-item>

      <v-tab-item id="/dashboard">

        <v-divider />

        <router-view v-if="activeTab === '/dashboard'" />

      </v-tab-item>

      <v-tab-item id="/admin">

        <v-divider />

        <router-view v-if="activeTab === '/admin'" />

      </v-tab-item>

    </v-tabs>

  </v-app>

</template>

<script>

import axios from 'axios'

import Login from '@/components/Login'

import { mapGetters, mapMutations } from 'vuex'

import fetchFavoriteOffers from './plugins/fetchFavoriteOffers'

export default {

  name: 'App',

  components: {

    Login

  },

  data() {

    return {

      activeTab: null,

      isLoading: false

    }

  },

  computed: {

    authAdmin: function () {

      if (this.$auth.user().role === 'admin' || this.$auth.user().role === 'super\_admin') {

        return true

      }

      return false

    }

  },

  created() {

    this.$auth.ready(function () {

        if (this.$auth.token()) {

          this.fadeLogin()

          this.changeAuthState()

          this.fetchFavoriteOffers()

        }

    });

  },

  methods: {

    ...mapGetters([

      'getLoginButton',

      'getAuthState'

    ]),

    ...mapMutations([

      'changeAuthState',

      'fadeLogin'

    ]),

    logout() {

      this.isLoading = true

      this.$auth.logout({

        success: function () {

          this.isLoading = false

          this.fadeLogin()

          this.changeAuthState()

        },

        error: function () {

          console.log("Something went wrong with Your logout!")

        },

      })

    },

    async fetchFavoriteOffers () {

      const result = await fetchFavoriteOffers(this.$auth.user().id)

      if (result) {

        this.$store.commit('setFavoriteOffers', result)

      }

    },

  },

}

</script>

**frontend/auth.js**

import bearer from '@websanova/vue-auth/drivers/auth/bearer.js'

import authaxios from '@websanova/vue-auth/drivers/http/axios.1.x'

import authrouter from '@websanova/vue-auth/drivers/router/vue-router.2.x.js'

const config = {

  auth: bearer,

  http: authaxios,

  router: authrouter,

  tokenDefaultName: 'laravel-vue-spa',

  tokenStore: ['localStorage'],

  rolesVar: 'role',

  registerData: {url: 'http://127.0.0.1:8000/api/auth/register', method: 'POST', redirect: ''},

  loginData: {url: 'http://127.0.0.1:8000/api/auth/login', method: 'POST', redirect: '', fetchUser: true},

  logoutData: {url: 'http://127.0.0.1:8000/api/auth/logout', method: 'POST', redirect: '/', makeRequest: true},

  fetchData: {url: 'http://127.0.0.1:8000/api/auth/user', method: 'GET', enabled: true},

  refreshData: {url: 'http://127.0.0.1:8000/api/auth/refresh', method: 'GET', enabled: true, interval: 30}

}

export default config

**frontend/main.js**

import Vue from 'vue'

import VueRouter from 'vue-router'

import App from './App.vue'

import router from './router'

import auth from './auth'

import store from './store'

import vuetify from './plugins/vuetify';

import axios from 'axios'

import VueAxios from 'vue-axios'

import VueAuth from '@websanova/vue-auth'

Vue.config.productionTip = false

Vue.router = router

Vue.use(VueRouter)

// Set Vue authentication

Vue.use(VueAxios, axios)

axios.defaults.baseURL = `${process.env.MIX\_APP\_URL}/api`

Vue.use(VueAuth, auth)

new Vue({

  router,

  store,

  vuetify,

  render: h => h(App)

}).$mount('#app')

**frontend/router.js**

import VueRouter from 'vue-router'

import Home from './views/Home.vue'

import Offers from './views/Offers.vue'

import Auction from './views/Auction.vue'

import Registration from './views/Registration.vue'

import Dashboard from './views/Dashboard.vue'

import AdminDashboard from './views/admin/ControlPanel.vue'

import Forum from './views/Forum.vue'

const routes = [

  {

    path: '/',

    name: 'home',

    component: Home

  },

  {

    path: '/offers',

    name: 'offers',

    component: Offers

  },

  {

    path: '/auction',

    name: 'auction',

    component: Auction

  },

  {

    path: '/forum',

    name: 'forum',

    component: Forum

  },

  //auth forms

  {

    path: '/signup',

    name: 'signup',

    component: Registration,

    meta: {

      auth: false

    }

  },

  //user routes

  {

    path: '/dashboard',

    name: 'dashboard',

    component: Dashboard,

    meta: {

      auth: true

    },

  },

  //admin routes

  {

    path: '/admin',

    name: 'admin.dashboard',

    component: AdminDashboard,

    meta: {

      auth: {roles: ['super\_admin', 'admin'], redirect: {name: 'offers'}, forbiddenRedirect: '/403'}

    },

  },

  // everything else

  {

    path: '\*',

    name: 'any',

    component: Home

  },

]

const router = new VueRouter({

  history: true,

  mode: 'history',

  routes,

})

export default router

**frontend/store.js**

import Vue from 'vue'

import Vuex from 'vuex'

Vue.use(Vuex)

export default new Vuex.Store({

  state: {

    loginButtonClicked: false,

    isLogged: false,

    addOfferDialog: false,

    charityTotal: 0,

    commercialTotal: 0,

    offersFilters: {

      title: null,

      description: null,

      parts: null,

      category: null,

      date: null

    },

    favoriteOffers: null

  },

  mutations: {

    fadeLogin: state => {

      state.loginButtonClicked = !state.loginButtonClicked

    },

    changeAuthState: state => {

      state.isLogged = !state.isLogged

    },

    changeAppendOfferState: state => {

      state.addOfferDialog = !state.addOfferDialog

    },

    setCharityTotal (state, total) {

      state.charityTotal = total

    },

    setCommercialTotal (state, total) {

      state.commercialTotal = total

    },

    setOffersFilters (state, filters) {

      state.offersFilters = filters

    },

    setFavoriteOffers (state, offers) {

      state.favoriteOffers = offers

    }

  },

  getters: {

    getLoginButton: state => {

      return state.loginButtonClicked

    },

    getAuthState: state => {

      return state.isLogged

    },

    getAppendingOfferState: state => {

      return state.addOfferDialog

    },

    charityTotal: state => state.charityTotal,

    commercialTotal: state => state.commercialTotal,

    offersFilters: state => state.offersFilters,

    favoriteOffers: state => state.favoriteOffers

  },

  actions: {

  }

})

**api.php**

<?php

use Illuminate\Http\Request;

/\*

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

| API Routes

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

|

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

| routes are loaded by the RouteServiceProvider within a group which

| is assigned the "api" middleware group. Enjoy building your API!

|

\*/

Route::middleware('auth:api')->get('/user', function (Request $request) {

    return $request->user();

});

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

    Route::post('register', 'AuthController@register');

    Route::post('login', 'AuthController@login');

    Route::get('refresh', 'AuthController@refresh');

    Route::group(['middleware' => 'auth:api'], function(){

        // Auth

        Route::get('user', 'AuthController@user');

        Route::post('logout', 'AuthController@logout');

        Route::post('userUpdate', 'UserController@update');

        Route::post('userUpdateAvatar', 'UserController@updateAvatar');

        Route::delete('userDelete', 'UserController@delete');

        // Offers

        Route::prefix('offer')->group(function () {

            Route::post('get', 'OfferController@getUserOffers');

            Route::post('add', 'OfferController@store');

            Route::post('delete\_soft', 'OfferController@softDelete');

        });

        Route::prefix('offers')->group(function () {

            Route::get('/', 'OfferController@getServiceOffers');

            Route::post('setFavorite', 'OfferController@changeFavorite');

        });

        // Auction

        Route::prefix('auction')->group(function () {

            Route::post('addParticipant', 'AuctionParticipantsController@store');

            Route::post('checkBid', 'CommercialAuctionController@checkBid');

            Route::post('add', 'AuctionController@store');

            Route::post('finish', 'AuctionController@finishAuction');

        });

        Route::post('auctions', 'AuctionController@getUserAuctions');

        // Forum

        Route::prefix('forum')->group(function () {

            Route::prefix('topic')->group(function () {

                Route::post('add', 'TopicController@store');

                Route::post('delete', 'TopicController@delete');

            });

            Route::prefix('comment')->group(function () {

                Route::post('add', 'TopicCommentController@store');

                Route::post('delete', 'TopicCommentController@delete');

            });

        });

        // Users

        Route::prefix('users')->group(function () {

            Route::get('/', 'UserController@index')->middleware('isAdmin');

            Route::get('service', 'UserController@showAll')->middleware('isAdmin');

            Route::get('/{id}', 'UserController@show')->middleware('isAdminOrSelf');

        });

        Route::prefix('user')->group(function () {

            Route::post('setStatus', 'UserController@setStatus');

            Route::post('setRole', 'UserController@setRole');

        });

    });

});

Route::prefix('offers')->group(function () {

    Route::get('/getAll', 'OfferController@getAll');

    Route::get('/{filters?}', 'OfferController@index');

    Route::get('/favorites/{user\_id?}', 'OfferController@getFavoriteOffers');

});

Route::get('/offer/{id}', 'OfferController@show');

Route::get('/offer\_media/{id}', 'OfferController@media');

Route::get('/offer\_categories', 'OfferCategoryController@index');

Route::get('/auctions/{status?}', 'AuctionController@index');

Route::prefix('forum')->group(function () {

    Route::get('/categories', 'TopicCategoryController@index');

    Route::get('/comments/{topic?}', 'TopicCommentController@index');

    Route::get('/topics/{category?}', 'TopicController@index');

});