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

Izglītības programma: Programmēšana

**KVALIFIKĀCIJAS DARBS**

**Nomāt Prieks mājaslapas datu uzskaites**

**automatizētā sistēma**

Paskaidrojošais raksts 108 lpp.

Audzēknis: R. Kiršholcs

Vadītājs: M. Danne

Normu kontrole: O. Sabanska

2019/2020.m.g.

**Rīga**

**ANOTĀCIJA**

Kvalifikācijas darbā ir aprakstīts Nomāt prieks mājaslapas datu uzskaites automatizētās sistēmas izstrādāšanas process. Sistēma dod klientiem iespēju pasūtīt sev vajadzīgās preces kādam pasākumam, vai kādam citam iemeslam uz sevis izvēlēto vietu. Sistēma tika izstrādāta PHP programmēšanas valodā, izmantojot Boostrap 4.5.0 ietvaru un tiek izmantota arī MySQL datubāzes vadības sistēma.

            Kvalifikācijas darbs ietver ievadu, uzdevumu nostādni, prasību specifikāciju, uzdevuma risināšanas līdzekļu izvēles pamatojumu, programmatūras produkta modelēšanas un projektēšanas aprakstu, datu struktūru aprakstu, lietotāju ceļvedi, nobeigumu un pielikumu. Kvalifikācijas darba ievadā ir aprakstīta produktu nomāšanu. Uzdevuma nostādnē ir norādīti uzdevumi, kurus sistēmai būs nepieciešams veikt. Prasību specifikācija sastāv no ieejas un izejas informācijas, kā arī no sistēmas funkcionālajām un nefunkcionālajām prasībām. Uzdevuma risināšanas līdzekļu izvēles pamatojumā ir norādīti, kādi līdzekļi tiks izmantoti izstrādāšanai un kādiem nolūkiem tie tiek izmantoti. Programmatūras produkta modelēšanas un projektēšanas apraksts sastāv no sistēmas modeļa, kas ietver sistēmas arhitektūru un ER modeli, un funkcionālās sistēmas modeļi, kas satur datu plūsmu modeli. Datu struktūru aprakstā tiek parādīta datu bāzes relāciju shēma, kā arī tabulu struktūra ar aprakstu, kas ietver datu tipu, datu garumu un piezīmju norādīšanu. Lietotāja ceļvedī ir norādītas nepieciešamās sistēmas prasības aparatūrai un programmatūrai, sistēmas instalācija un palaišana, kā arī programmas apraksts, kas paskaidro, kā pareizi lietot sistēmu. Testa piemērā ir parādīts, kā administrators pievieno jaunu produktu un to rediģē.

            Kvalifikācijas darbs sastāvs no 108 lappusēm, kurā ietilpst 35 attēli un 4 tabulas.

**ANNOTATION**

The qualification work describes the process of developing the Automated System for “Nomat prieks” Website Data Accounting. The system gives customers the opportunity to order the goods they need for an event or for some other reason to a place of their choice. The system was developed in PHP programming language using Boostrap 4.5.0 framework and also uses MySQL database management system.

            The qualification work includes an introduction, task statement, requirements specification, justification of the choice of task solving means, description of software product modeling and design, description of data structures, user guide, conclusion and appendix. The introduction to the qualification paper describes product rental. The task statement specifies the tasks that the system will need to perform. The requirements specification consists of input and output information as well as system functional and non-functional requirements. The rationale for choosing the means of solving the task indicates which tools will be used for the development and for what purposes they are used. The software product modeling and design description consists of a system model that includes a system architecture and an ER model, and functional system models that contain a data flow model. The description of the data structures shows the database relational scheme, as well as the table structure with the description, which includes the data type, data length and notes. The user guide provides the necessary system hardware and software requirements, system installation and startup, and a program description that explains how to use the system properly. The test example shows how an administrator adds and edits a new product.

            The qualification paper consists of 108 pages, which includes 35 images and 4 tables.

**SATURS**

[**IEVADS** 5](#_Toc41999819)

[**1.** **Uzdevuma nostādne** 6](#_Toc41999820)

[**2.** **PRASĪBU SPECIFIKĀCIJA** 7](#_Toc41999821)

[**2.1.** **Ieejas un izejas informācijas apraksts** 7](#_Toc41999822)

[**2.2.** **Funkcionālās prasības** 8](#_Toc41999823)

[**2.3.** **Nefunkcionālās prasības** 9](#_Toc41999824)

[**3.** **UzdevumU RISINĀŠANAS LĪDZEKĻU IZVĒLE UN PAMATOJUMS** 10](#_Toc41999825)

[**Izmantotie aizgultie moduļi** 10](#_Toc41999826)

[**4. PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA** 11](#_Toc41999827)

[**4.1** **Sistēmas struktūras modelis** 11](#_Toc41999828)

[**4.2** **Funkcionālas sistēmas modelis** 15](#_Toc41999829)

[**5** **DATU STRUKTŪRAS APRAKSTS** 23](#_Toc41999830)

[**6** **LIETOTĀJA CEĻVEDIS** 25](#_Toc41999831)

[**6.1** **Sistēmas nepieciešamības programmatūrai** 25](#_Toc41999832)

[**6.2** **Sistēmas instalācija un palaišana** 26](#_Toc41999833)

[**6.3** **Programmas apraksts** 26](#_Toc41999834)

[**6.4** **Testa piemērs** 30](#_Toc41999835)

[**NOBEIGUMS** 34](#_Toc41999836)

[**7** **INFORMĀCIJAS AVOTI** 35](#_Toc41999837)

[**PIELIKUMI** 36](#_Toc41999838)

**IEVADS**

Mūsdienās cilvēkiem ir ļoti daudz un dažādu veidu kā pavadīt savu brīvu laiku. Un kā zināms liela daļa šīs valsts iedzīvotāju dzīvo lielpilsētās, tāpēc arī populārāka kļūst brīvā laika pavadīšana pie dabas krūts. Bet diemžēl ne visiem ir lauku mājas, kur aizbraukt un pavadīt savu brīvo laiku.

Un kā jau zināms, retajam mājās ir telts, āras galdi, vai citas lietas, kas palīdzētu noorganizēt kādu pasākumu, vai arī vienkāršu saiešanos ar draugiem. Protams ir iespējams atrast internet vietni, kurā var izīrēt piemēram telti, vai arī atrast citu vietni, kurā varētu noīrēt baļļu, bet tas prasīs ļoti daudz laika, meklējot katrā vietnē kādu citu produktu. Piedevām ar katru no šīm vietnēm būs individuāli jāsazinās, vēl vairāk jātērē savs laiks tikai lai atrastu sev vajadzīgās lietas pasākumam. Šīs sistēmas mērķis ir samazināt cilvēku patērēto laiku pie interneta pārlūka meklējot sev nepieciešamo savam pasākumam.

Sistēma būs pieejama gan viesiem, gan arī reģistrētiem lietotājiem. Bet sistēmu izmantojot kā viesim būs pieejama tikai maza daļa no visām sistēmas funkcijām. Apskatīt produktus varēs ikviens, bet pievienot produktus grozam un tos pasūtīt varēs tikai reģistrēti lietotāji.

1. **Uzdevuma nostādne**

Kvalifikācijas darba uzdevums ir izveidot nomas sistēmu. Ar kuras palīdzību lietotāji varēs apskatīt produktus un tos iznomāt. Šī sistēma palīdzēs cilvēkiem samazināt pie datora pavadīto laiku, meklējot sev vajadzīgās lietas pasākumam. Sistēmai ir jānodrošina iespēju pievienot lietotājus ar dažādām lomām, piedāvātos produktus, sagrupēt sev nepieciešamos produktus vienuviet un tos pasūtīt.

Nomāt prieks sistēmai ir jāizpilda vairākas funkcionalitātes:

* Datu – lietotāja, produktu – pievienošana, rediģēšana, apstrāde un dzēšana;
* Produktu izvēle
* Lietotāju pieslēgšanos sistēmai
* Lietotāju administrēšana
* Produktu administrēšana

1. **PRASĪBU SPECIFIKĀCIJA**
   1. **Ieejas un izejas informācijas apraksts**
      1. *Ieejas informācijas apraksts*

Sistēmā tiks nodrošināta šāda ieejas informāciju apstrāde.

1. Informācija par lietotāju sastāvēs no šādiem datiem
   1. Lietotājvārds - burtu un ciparu teksts ar izmēru līdz 255 rakstzīmēm
   2. E-pasts – burtu un ciparu teksts ar izmēru līdz 100 rakstzīmēm.
   3. Parole - burtu un ciparu teksts ar izmēru līdz 255 rakstzīmēm.
2. Informācija par produktiem sastāvēs no šādiem datiem.
   1. Nosaukums – burtu un ciparu teksts ar izmēru līdz 255 rakstzīmēm.
   2. Cena – burtu un ciparu teksts ar izmēru līdz 50 rakstzīmēm
   3. Pieejamība - burtu un ciparu teksts ar izmēru līdz 100 rakstzīmēm
   4. Tips – burtu un ciparu teksts ar izmēru līdz 100 rakstzīmēm
   5. Krājums – cipars ar izmēru 11
   6. Bilde – Bildes fails
3. Informācija par kontaktu formu sastāvēs no šādiem datiem
   1. Vārds - burtu un ciparu teksts ar izmēru līdz 100 rakstzīmēm
   2. E-pasts - burtu un ciparu teksts ar izmēru līdz 100 rakstzīmēm.
   3. Iemesls - burtu un ciparu teksts ar izmēru līdz 100 rakstzīmēm.
   4. Apraksts - burtu teksts ar izmēru līdz 400 rakstzīmēm.
      1. *Izejas informācijas apraksts*
4. Kopējā informācija par pasūtījumu, kas tika veikts, tā kopējā cena un klienta pasūtītās preces.
   * 1. *Arējās informācijas apraksts*

Šajā sistēmā nav ārējās informācijas apstrādes.

* 1. **Funkcionālās prasības**

1. Jānodrošina lietotāja autorizācija
   1. Sistēmai jānodrošina iespēju reģistrēties klientam.
   2. Sistēmai jānodrošina, ka lietotājvārds ir unikāls.
   3. Sistēmai jānodrošina, ka E-pasts ir unikāls.
   4. Sistēmai jānodrošina E-pasta apstiprināšanu.
2. Jānodrošina administratoru autorizācija.
   1. Sistēmai jānodrošina iespēju pievienot administratoru.
   2. Sistēmai jānodrošina atsevišķas pieejas administratoram.
   3. Sistēmai jānodrošina atsevišķas darbības administratoram.
3. Jānodrošina administratora un lietotāja piekļuvi
   1. Sistēmai jānodrošina piekļuvi, ievadot lietotājvārdu un paroli.
   2. Sistēmai nedrīkst atļaut piekļuvi, ja dati ir ievadīti nepareizi.
   3. Sistēmai jāatšķir administratoru un parastu lietotāju.
   4. Sistēmai jāliedz piekļuvi administratora funkcijām parastam lietotājam.
   5. Sistēmai jāizvada ziņojums, ja dati tika ievadīti nepareizi.
   6. Sistēmai jāpārbauda, vai ir aizpildīti visi nepieciešamie lauki.
4. Jānodrošina lietotāja informācijas apstrāde.
   1. Sistēmai jānodrošina lietotājam atjaunot paroli.
   2. Sistēmai jānodrošina pārsūtīt verifikācijas E-pastu.
   3. Sistēmai jāizvada ziņojums, ja dati tika ievadīti nepareizi.
   4. Sistēmai jānodrošina administratoram iespēju mainīt lietotājvārdu.
   5. Sistēmai jānodrošina administratoram lietotāja E-pasta maiņu.
   6. Sistēmai jānodrošina administratoram padarīt citu lietotāju par administratoru.
   7. Sistēmai jānodrošina administratoram iespēju dzēst lietotāju.
5. Jānodrošina produkta informācijas apstrāde.
   1. Sistēmai jānodrošina, ka produktu var pievienot, ja ir aizpildīti visi nepieciešamie lauki.
   2. Sistēmai jānodrošina, ka produktam var pievienot attēlu.
   3. Sistēmai jānodrošina produkta meklēšana pēc pievienošanas ID.
   4. Sistēmai jānodrošina iespēju meklēt produktus, ko ir pievienojis kāds administrators.
   5. Sistēmai jānodrošina iespēju meklēt produktu pēc nosaukuma.
   6. Sistēmai jānodrošina iespēju rediģēt pievienotos produktus.
   7. Sistēmai jānodrošina iespēju apskatīt pievienotā produkta detaļas.
   8. Sistēmai jānodrošina administratoram iespēju dzēst produktu.
   9. **Nefunkcionālās prasības**
6. Sistēmas saskares valodai ir jābūt angļu valodā.
7. Jānodrošina iespēju lietotājam piekļūt pie mājaslapas no jebkuras ierīces, vai ekrāna izmēra.
8. Jānodrošina viegla un patīkama produktu apskatīšana.
9. Mājaslapas dizainā ir jābūt nosaukumam – Nomāt prieks!
10. Izvēlnei, kas attiecas uz mājaslapu ir jābūt ekrāna augšā un arī jābūt responsīvam.
11. **UzdevumU RISINĀŠANAS LĪDZEKĻU IZVĒLE UN PAMATOJUMS**

Veidojot šo sistēmu es izmantoju dažādus rīkus.

Šīs sistēmas galvenā izstrādes vide bija PhpStorm(2020.1.1). Šo vidi izvēlējos, jo to mēs izmantojām jau skolā izstrādājot mājaslapas, kā arī viņa piedāvā visādus noderīgus rīkus, kas jau ir sistēmā ir jau iekļauti (kompozeri, kļūdu meklētāju, koda priekšāteicēju utt.). Un protams šī izstrādes vide ir pieejama studentiem bezmaksas.

Sistēma tika bāzēta un atvērtā koda izstrādes vidi PHP “Yii2”. Izmantojot šo programmēšanas ietvaru tiek izmantota ļoti zināma programmēšanas arhitektūra – MVC(Model-View-Controller), kas latviski tulkojas – MSK(Modelis-Skats-Kontrolieris).

Mājaslapas darbību interneta vietnēs nodrošina XAMPP(v7.4.6) programmatūra, kas sevī ietver PHP(v7.4.6). No paša XAMPP tika izmantoti tikai Apache serveris un MySQL pārvaldības sistēma datubāzei.

Visi dati tika saglabāti MySQL sistēmā un šīs datubāzes veidošanai un rediģēšanai tika izmantots “PhpMyAdmin”, jo jau bija personīgi izveidojusies pieredze skolā, taisot un rediģējot citas datubāzes.

**Izmantotie aizgultie moduļi**

Tika izmantots E-pastu sūtīšas un verificēšanas modulis - “SwiftMailer”, kas tiek izmantots pie : reģistrācijas, ielogošanās, paroles atjaunošanas, reģistrācijas tokena apstiprināšanas.

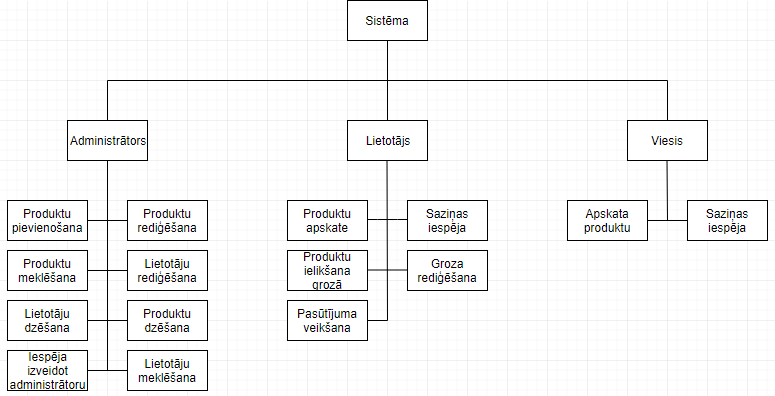
Paziņojumu izvadīšanai tika izmantots widget-alert.

**4. PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA**

* 1. **Sistēmas struktūras modelis**

**Sistēmas arhitektūra**

Mana sistēmai ir sadalīta pa 3 daļām. (Skat att. 4.1.) Viesim, lietotājam un administratoram. Administratoram ir pieejamas visas šīs mājaslapas iespējas, lietotājs var veikt tikai dažas mājaslapas funkcijas un viesis var tikai apskatīt preces.

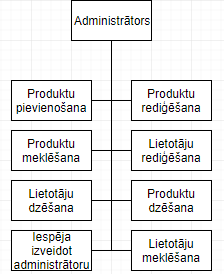


4.1 att. Sistēmas struktūras modeļa attēlojums

**Administratora sistēma**

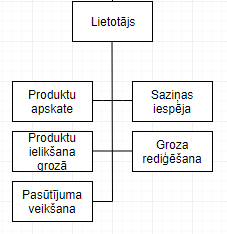
Visvairāk pieeju šajā sistēmā ir administratoram, tie var pievienot produktus, tos meklēt un rediģēt. (Skat att. 4.2.) Var apskatīt visus reģistrētos lietotājus, labot viņu profilus un padarīt tos par administratoriem nepieciešamības gadījumā.

4.2 att. Administratora sistēmas struktūras modeļa attēlojums



**Autorizēta lietotāja sistēma**

Pašam lietotājam ir vairākas iespējas šajā mājaslapā. Viņš par apskatīt preces, tās pievienot savam grozam, rediģēt savu grozu un ir iespēja pasūtīt izvēlētos produktus. (Skat att. 4.3.) Kā arī ja ir nepieciešams var sazināties ar mājaslapas administrāciju.

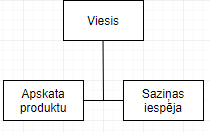


4.3 att. Autorizēta lietotāja struktūras modeļa attēlojums

**Viesu sistēma**

Parastam mājaslapas viesim ir pieejama ļoti maza daļa no visas mājaslapas

funkcionalitātes. (Skat att. 4.4.) Viņš var apskatīt pieejamos produktus un sazināties ar administrāciju.



4.4 att. Viesa struktūras modeļa attēlojums

* + 1. Sistēmas ER modelis

Pati sistēma darbojas no galvenajam iestatītajām vērtībām un no administratoru pievienotajiem produktiem.

Datu kopai lietotāji ir 1 saite.

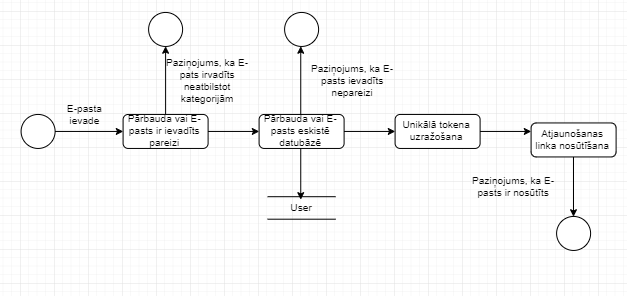
Saite ir E-pastiem viens pret vienu, jo administrators kas ir pievienojis konkrēto produktu var būt tikai viens.

Datu kopai produktu ir 1 saite.

Saite ir saistīta ar lietotājiem, jo katram produktam var būt tikai viens izveidotājs.

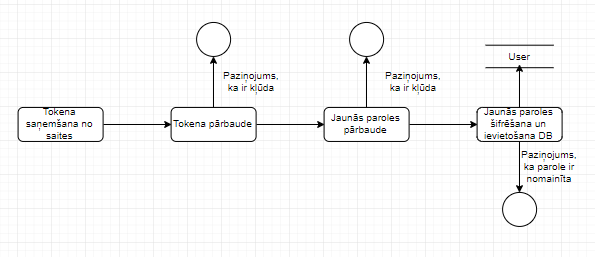
* 1. **Funkcionālas sistēmas modelis**

**Paroles maiņas pieprasīšana**

Paroles maiņas pieprasīšana (Skat att. 4.5.) pieprasot paroles maiņu atveras cits logs, kurā lietotajam ir jāievada savs E-pasts. No sākuma tiek pārbaudīts, vai E-pasts ir ievadīts atbilstoši nosacījumiem, ja nav, tad tiek izvadīts kļūdas paziņojums. Tad tiek pārbaudīts, vai tāds E-pasts ir datubāzē, ja nav, tad tiek izvadīts kļūdas paziņojums. Tad tiek uz ekrāna parādīts paziņojums, ka uz E-pasta adresi in nosūtīts ziņojums, kas satur paroles atjaunināšanas adresi. 

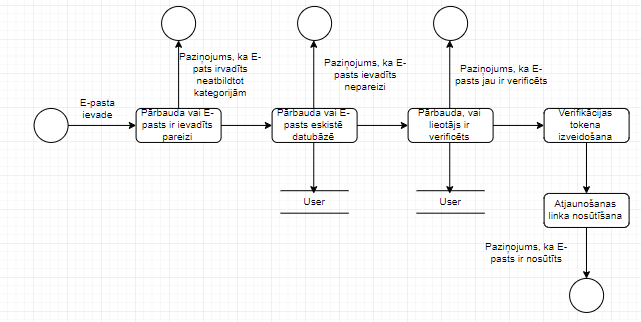
4.5 att. Paroles maiņas pieprasīšanas detalizēts attēlojums

**Paroles maiņa**

Paroles maiņa (Skat att. 4.6.) no sākuma tiek apstrādāts tokens, kas tiek saņemts atverot nosūtīto linku E-pastā. Tad tiek pārbaudīts, vai pats tokens vēl ir derīgs, ja nav, tad tiek izvadīts kļūdas paziņojums. Ja viss ir noticis sekmīgi, tad tiek atvērts jauns logs, kurā lietotājam lūdz ievadīt paroli. Tad tiek pārbaudīts, vai parole sakrīt ar iepriekš nostādītajiem nosacījumiem, ja nesakrīt tad tiek izvadīts kļūdas paziņojums. Tad parole tiek šifrēta un saglabāta datubāzē. Beigās tiek izvadīts paziņojums, ka parole ir nomainīta veiksmīgi. 

4.6 att. Paroles maiņas detalizēts attēlojums

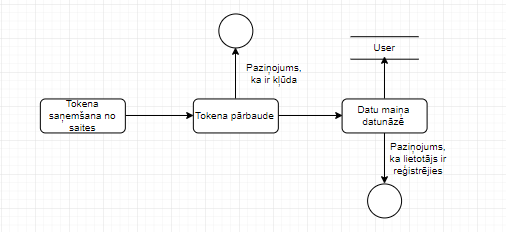
**Verifikācijas tokena pieprasīšana**

Verifikācijas tokena pieprasīšana (Skat att. 4.7.) pieprasot paroles maiņu atveras cits logs, kurā lietotajam ir jāievada savs E-pasts. No sākuma tiek pārbaudīts, vai E-pasts ir ievadīts atbilstoši nosacījumiem, ja nav, tad tiek izvadīts kļūdas paziņojums. Tad tiek pārbaudīts, vai tāds E-pasts ir datubāzē, ja nav, tad tiek izvadīts kļūdas paziņojums. Pēc tam tiek datubāzē pārbaudīts vai lietotājs jau ir verificēts, ja nav, tad tiek izvadīts kļūdas paziņojums. Pēc tam tiek izveidots tokens, kas tiek nosūtīts uz norādīto adresi. 

4.7 att. Verifikācijas tokena pieprasīšanas detalizēts attēlojums

**Verifikācijas tokena apstiprināšana un maiņa**

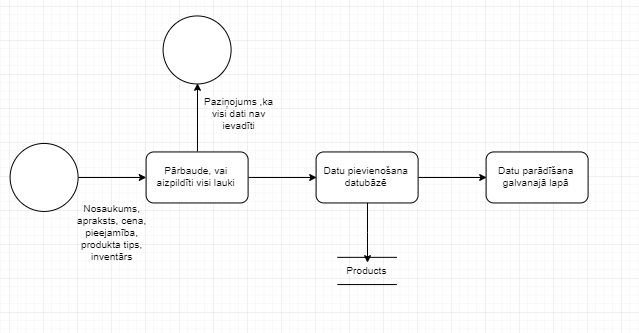
Verifikācijas tokena apstiprināšana un maiņa (Skat att. 4.8) no sākuma tiek apstrādāts tokens, kas tiek saņemts atverot nosūtīto linku E-pastā. Tad tiek pārbaudīts, vai pats tokens vēl ir derīgs, ja nav, tad tiek izvadīts kļūdas paziņojums. Ja viss ir noticis sekmīgi, tad tiek datubāzē nomainīts lietotāja status, tiek izvadīts paziņojums ka viss ir noticis sekmīgi un tiek atvērts jauns logs, kurā tiek parādīta mājaslapas galvenā lapa.



4.8 att. Verifikācijas tokena apstiprināšanas un maiņas detalizēts attēlojums

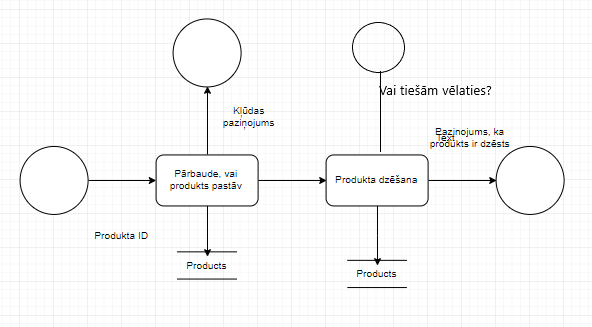
**Produktu pievienošana**

Produktu pievienošana, (Skat att. 4.9.) no sākuma administratoram tiek pieprasīts ievadīt visus nepieciešamos laukus. Ja tie netiek izvadīti paradās kļūdas paziņojums. Ja visi dati tika pievienoti pareizi tad tie tiek saglabāti datubāzē. Kad tie ir saglabāti datubāzē tos var apskatīt veikala galvenajā lapā.



4.9 att. Produktu pievienošanas detalizēts attēlojums

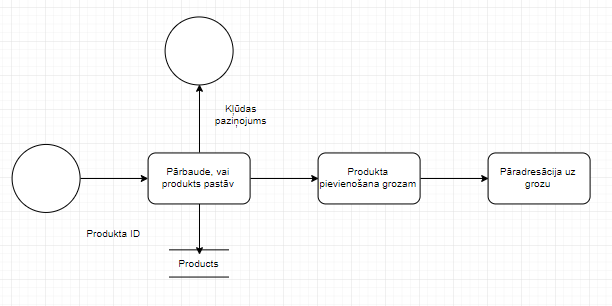
**Produktu dzēšana**

Produktu dzēšana, (Skat att. 4.10.) tiek izvēlēts produkta ID, kas tiek pārbaudīts datubāzē, vai tāds produkts vispār pastāv, ja tāda nav, tad tiek izvadīts kļūdas paziņojums. Tad uz ekrāna parādās paziņojums, vai tiešām vēlaties dzēst produktu, ja izvēlas turpināt, tikai tad produkts tiek dzēsts no datubāzes un tiek izvadīts paziņojums ka produkts tika izdzēsts sekmīgi. 

4.10 att. Produktu dzēšanas detalizēts attēlojums

**Produktu pievienošana grozam**

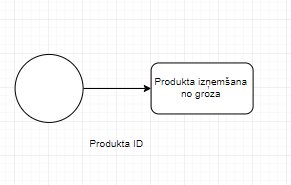
Produktu pievienošana grozam, (Skat att. 4.11.) tiek izvēlēts produkta ID, kas tiek pārbaudīts datubāzē, vai tāds produkts vispār pastāv, ja tāda nav, tad tiek izvadīts kļūdas paziņojums. Tad tiek atvērts groza logs un produkts tiek pievienots tur.



4.11 att. Produktu pievienošanas grozamdetalizēts attēlojums

**Produktu dzēšana no groza**

Produktu dzēšana no groza, (Skat att. 4.12.) tiek izvēlēts produkta ID un tas tiek izņemts no groza.



4.12 att. Produktu dzēšanas no groza detalizēts attēlojums

1. **DATU STRUKTŪRAS APRAKSTS**

Datubāze sastāv no 3 tabulām. Tās sevī ietver informāciju par produktiem, lietotājiem un kontaktu formām.

* + Tabula User satur datus par lietotājiem un administratoriem.
  + Tabula Products satur datus par visiem produktiem, kas ir veikalā.
  + Tabula Contact\_form satur datus par visām saziņām, ko ir veikuši lietotāji ar administrāciju

**Lietotāju dati “User”**

Šajā tabulā tiek saglabāti dati par visiem lietotājiem un administratoriem

5.1. tabula

Tabulas “**User**” struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | ID | Autonumber | 11 | Lietotāja ID |
| 2. | USERNAME | Varchar | 255 | Lietotājvārds |
| 3. | AUTH\_KEY | Varchar | 32 | Autorizācijas atslēga |
| 4. | PASSWORD\_HASH | Varchar | 255 | Paroles pelni |
| 5. | PASSWORD\_RESET TOKEN | Varchar | 255 | Paroles atjaunošanas tokens |
| 6. | EMAIL | Varchar | 255 | E-pasts |
| 7. | STATUS | Smallint | 6 | Vai lietotājs ir verificēts |
| 8. | CREATED\_AT | Integer | 11 | Izveides laiks |
| 9. | UPDATED\_AT | Integer | 11 | Atjaunināšanas laiks |
| 10. | VERIFICATION\_TOKEN | Varchar | 255 | Verifikācijas tokens |
| 11. | ADMIN | Integer | 1 | Vai ir administratos |

**Produktu dati “Products”**

Šajā tabulā tiek saglabāta visa informācija par produktiem. Kā arī lietotājiem kas viņus ir pievienojis

5.2. tabula

Tabulas “**Products**” struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | ID | Autonumber | 11 | Produkta ID |
| 2. | USER\_EMAIL | Varchar | 100 | Lietotāja E-pasts |
| 3. | NAME | Varchar | 255 | Produkta nosaukums |
| 4. | DESCRIPTION | Mediumtext | - | Produkta aprkasts |
| 5. | PRICE | Varchar | 50 | Cena |
| 6. | AVALIBILITY | Varchar | 100 | Pieejamība |
| 7. | TYPE | Varchar | 100 | Produkta tips |
| 8. | STOCK | Integer | 11 | Krājums |
| 9. | IMAGE | Varchar | 255 | Bildes nosaukums |
| 10. | CREATED\_AT | Timestamp | - | Izveides laiks |

**Kontaktu formas dati “Contact\_form”**

Šeit saglabājās dati par visām kontaktu formām, ko lietotāji ir aizpildījuši un nosūtījuši.

5.3. tabula

Tabulas “**Contact\_form**” struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| 1. | ID | Integer | 11 | Formas ID |
| 2. | NAME | Varchar | 100 | Sūtītāja vārds |
| 3. | EMAIL | Varchar | 100 | Sūtītāja E-pasts |
| 4. | SUBJECT | Varchar | 100 | Iemesls |
| 5. | BODY | Text | - | Apraksts |
| 6. | CREATED\_AT | Timestamp | - | Nosūtīšanas laiks |

1. **LIETOTĀJA CEĻVEDIS**
   1. **Sistēmas nepieciešamības programmatūrai**

Lai. sistēma spētu darboties, un to būtu iespējams apskatīt ir nepieciešams:

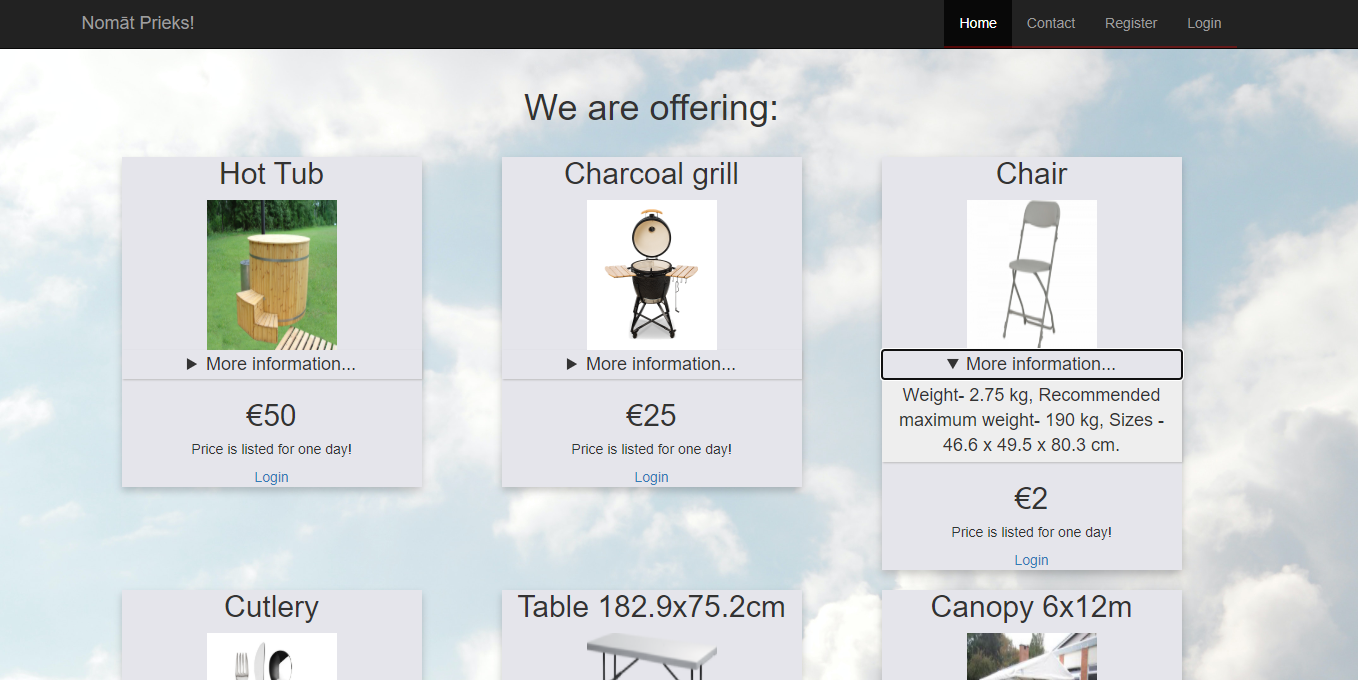
* Webserveris, kas nodrošina programmas darbību interneta vietnēs,
* Interneta pārlūkprogramma.

Sistēmas minimālās prasības:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Windows nepieciešamības | Mac nepieciešamības | Linux nepieciešamības |
| Operētājsistēma | Windows 7 vai jaunāks | Mac OS X Yosemite 10.10 vai jaunāks | 64-bit  Ubuntu 14.04+,  Debian 8+,  openSUSE 13.3+, vai  Fedora Linux 24+ |
| Procesors | Intel Pentium 4 vai jaunāks | Intel | Intel Pentium 4 vai jaunāks |
| Operatīvā atmiņa | Vismaz 1GB, ieteicams 2GB | | |
| Vieta uz HDD | Vismaz 4GB | | |

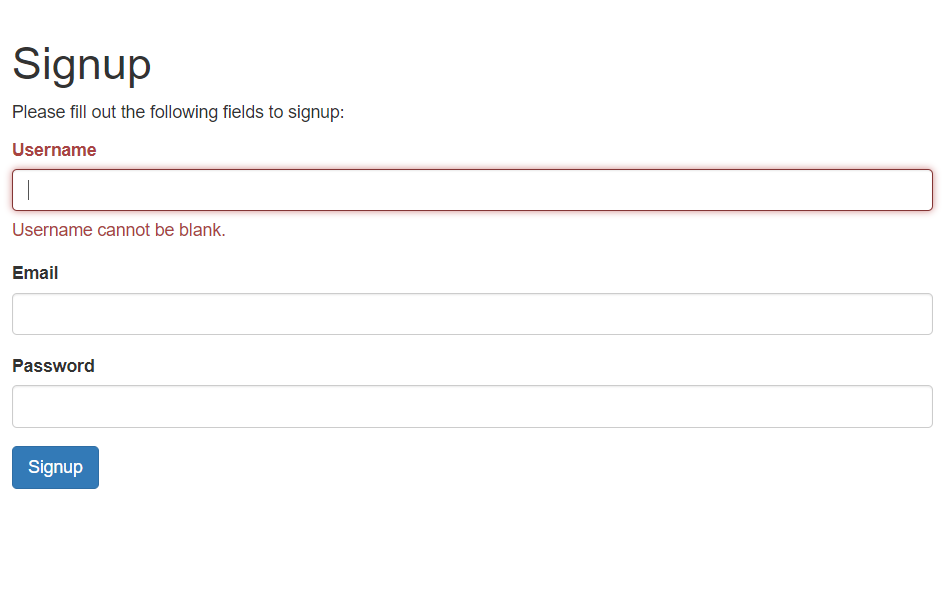
* 1. **Sistēmas instalācija un palaišana**
     + Virtuālā servera izveidošana, manā gadījūmā XAMPP
     + Nepieciešami visi sistēmas dati
     + Ievietot sistēmas failus mapē - xampp\htdocs
     + Kādā interneta pārlūkprogrammā jāievada – localhost
     + Augšējā labajā malā nepieciemsans uzspiest uz pogas - phpMyadmin
     + Importēt datubāzi yiiadvanced.sql failu, kas atrodas sistēmas mapē
     + Interneta pārlūkprogrammā nepieciešams ierakstīt - localhost/lapa/
  2. **Programmas apraksts**

Atverot mājaslapu (Skat 6.1. att.) uzreiz ir redzami visi produkti un augšēja sadaļā ir izvēles: Home, Contact, Register, Login.

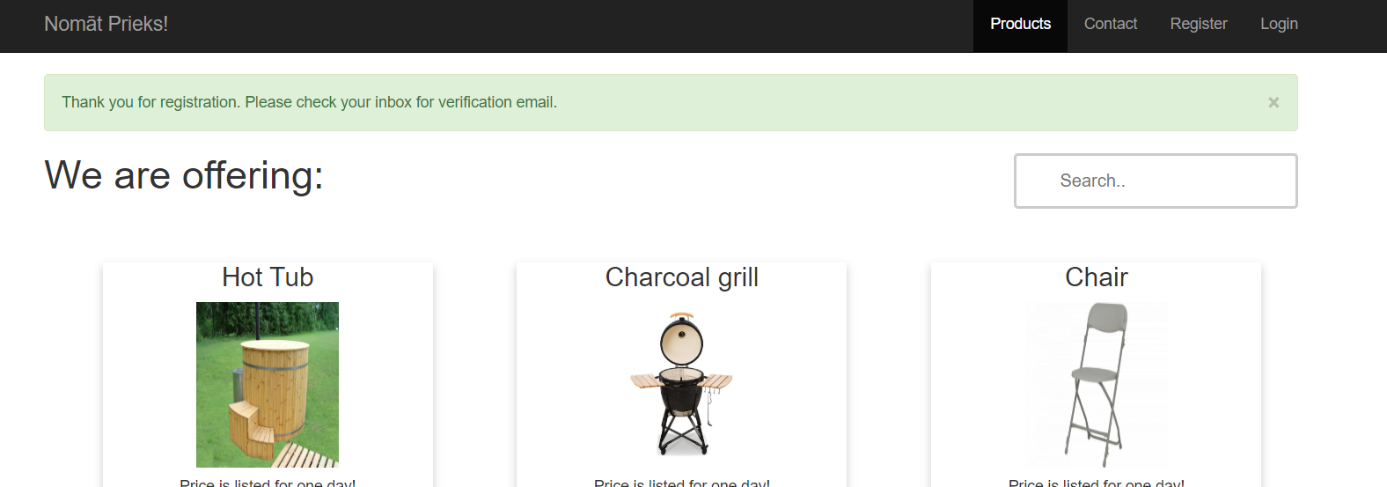


6.1 att. Sākuma ekrāna attēls

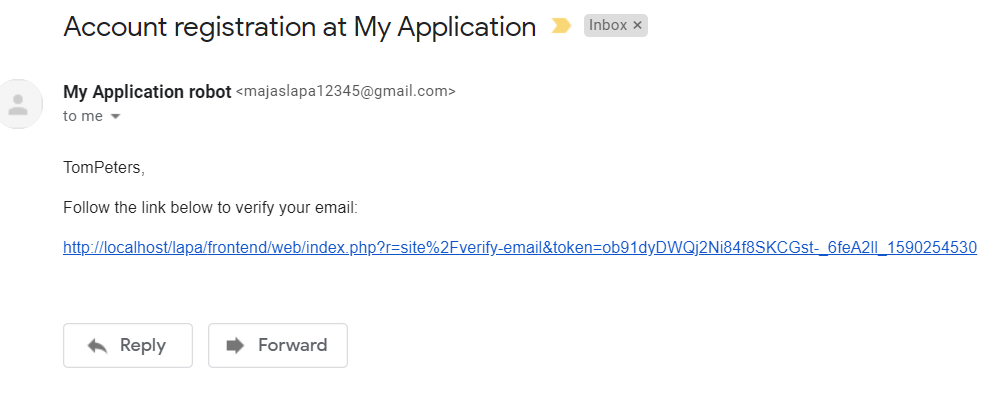
Lai reģistrētos (Skat 6.2. att.) viesim ir pieejams logs – Register, kurā ir nepieciešams aizpildīt visu nepieciešamo informāciju.



6.2 att. Reģistrācijas logs

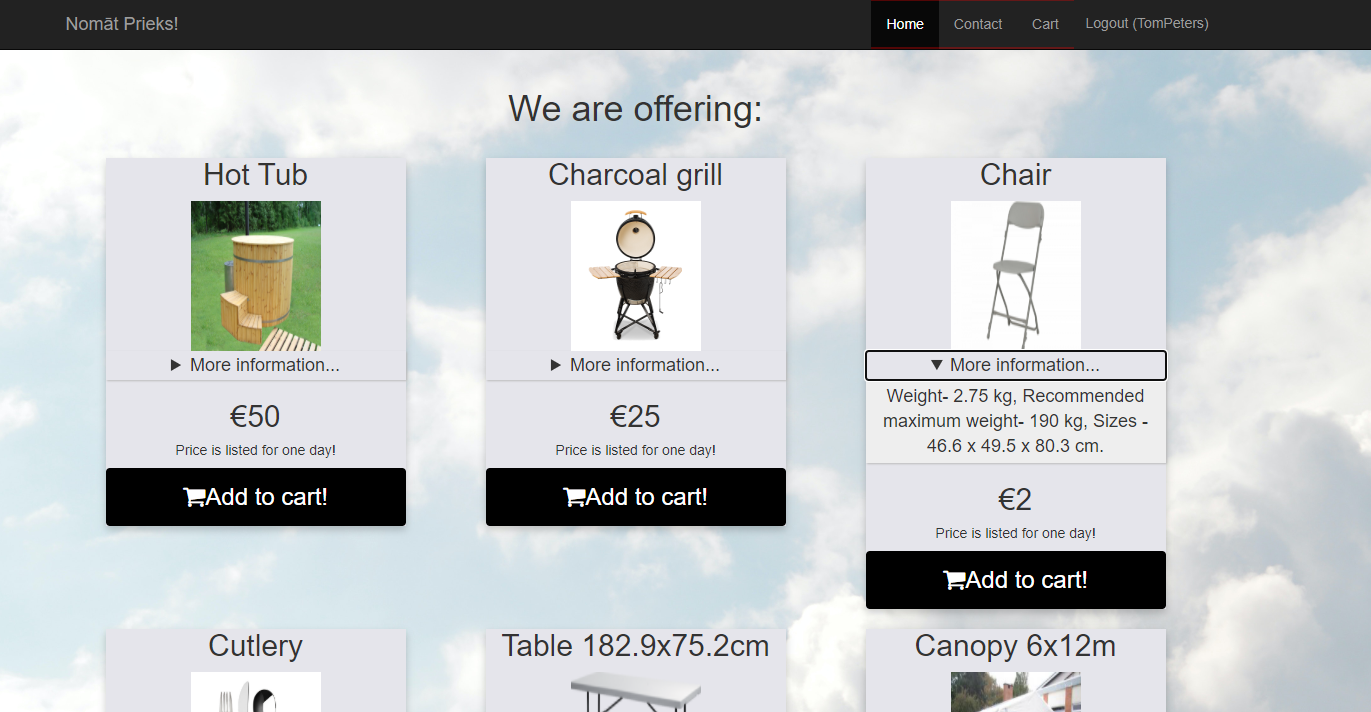
Šādu ekrānu lietotājam parāda pēc sekmīgas reģistrācijas(Skat 6.3. att.) 

6.3 att. Pēc reģistrācijas attēls

Šis ir verifikācijas E-pasts, ko saņem lietotājs pēc reģistrācijas (Skat 6.4. att.) 

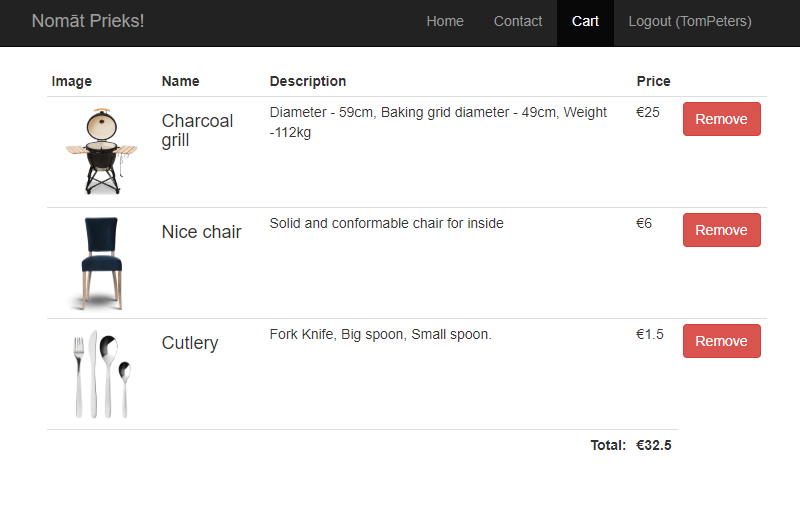
6.4 att. Reģistrācijas E-pasts

Šādi izskatās galvenā lapa reģistrētam lietotājam (Skat 6.5. att.)



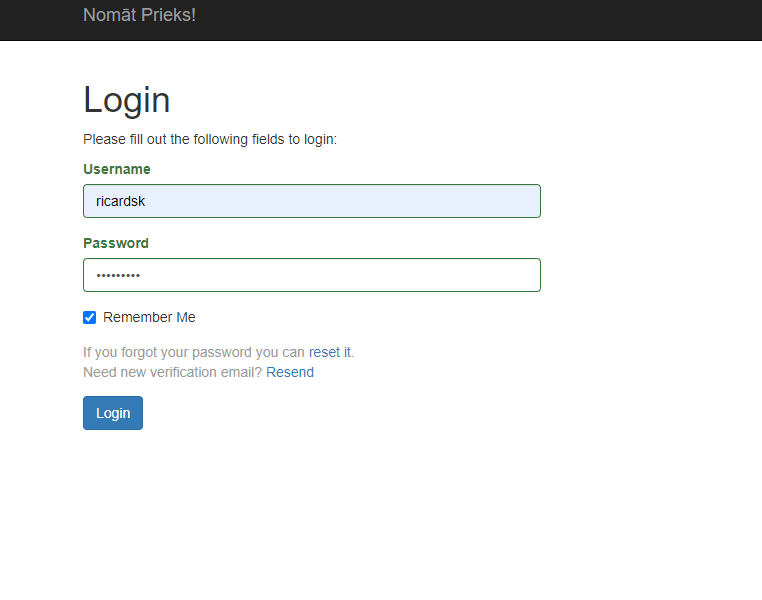
6.5 att. Galvanā lapa lietotājam

Šādi izskatās grozs, reģistrētam lietotājam (Skat 6.6. att.)



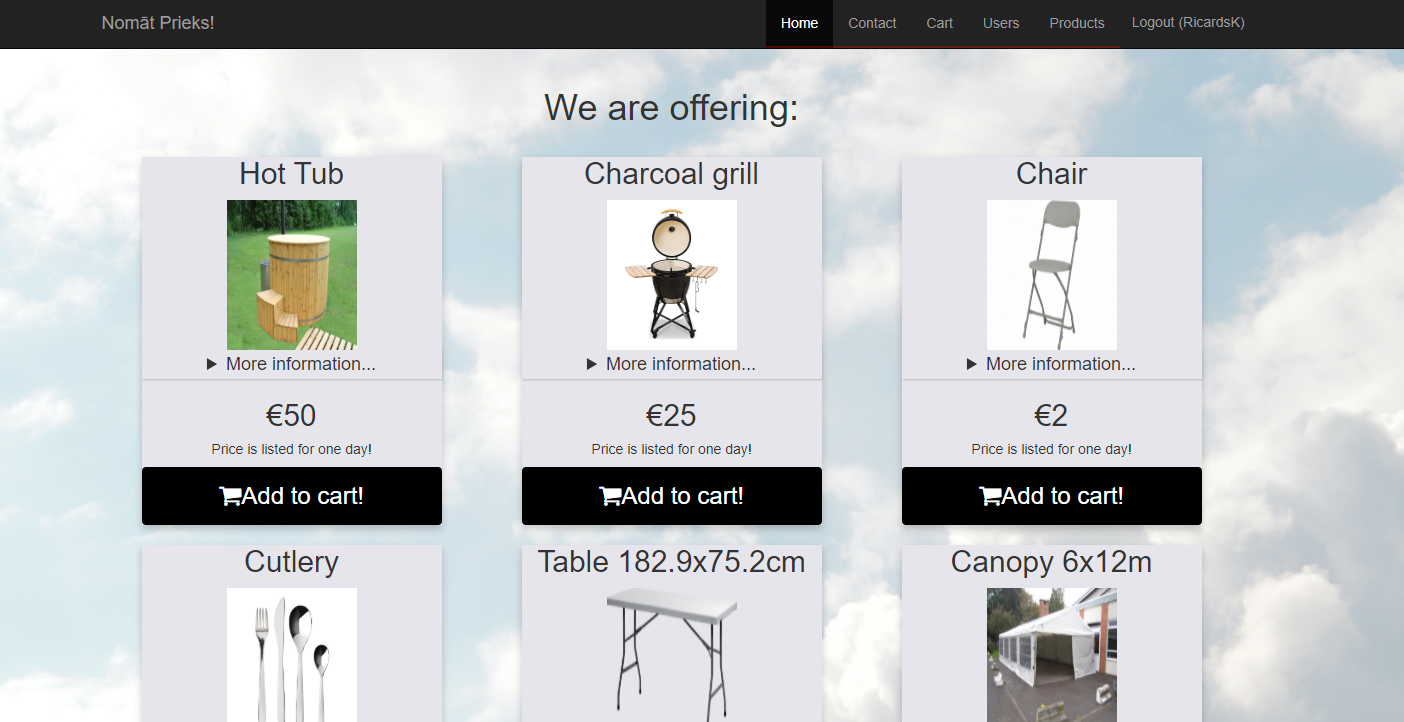
6.6 att. Lietotāja grozs

Šādi izskatās ielogošanās ekrāns reģistrācijas(Skat 6.7. att.)



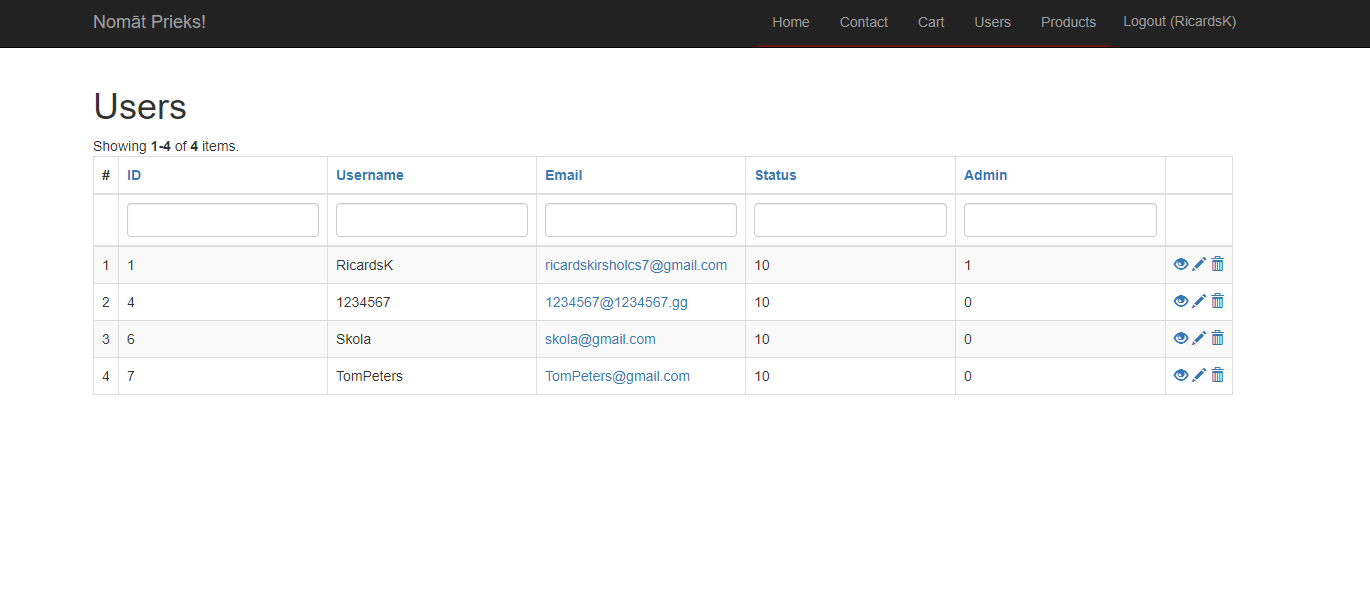
6.7. att. Pēc reģistrācijas attēls

Šādi izskatās ekrāns administratoram, pēc veiksmīgas ielogošanās sistēmā (Skat 6.8. att.)

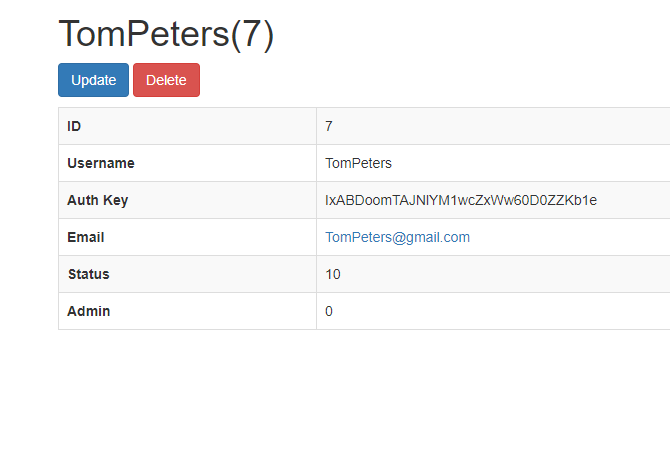


6.8 att. Administratora ekrāna attēls

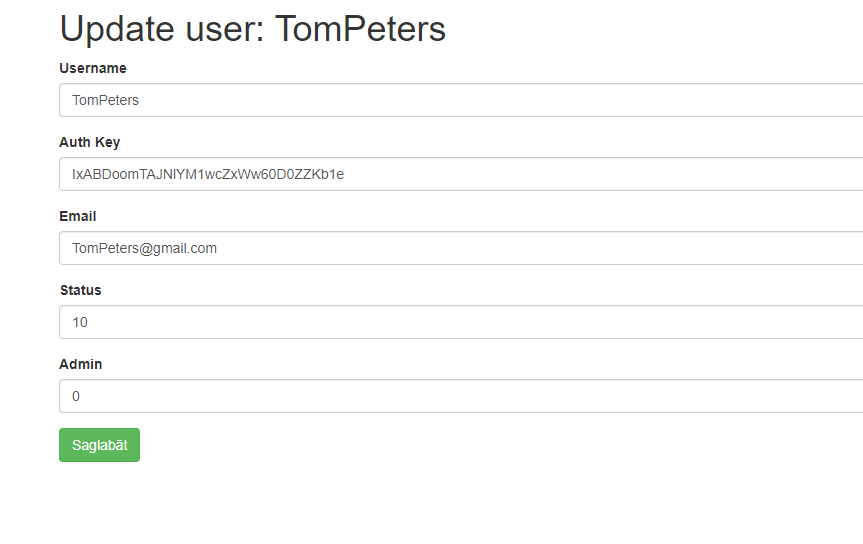
Šis ir lietotāju pārvaldīšanas ekrāns(Skat 6.9. att.)



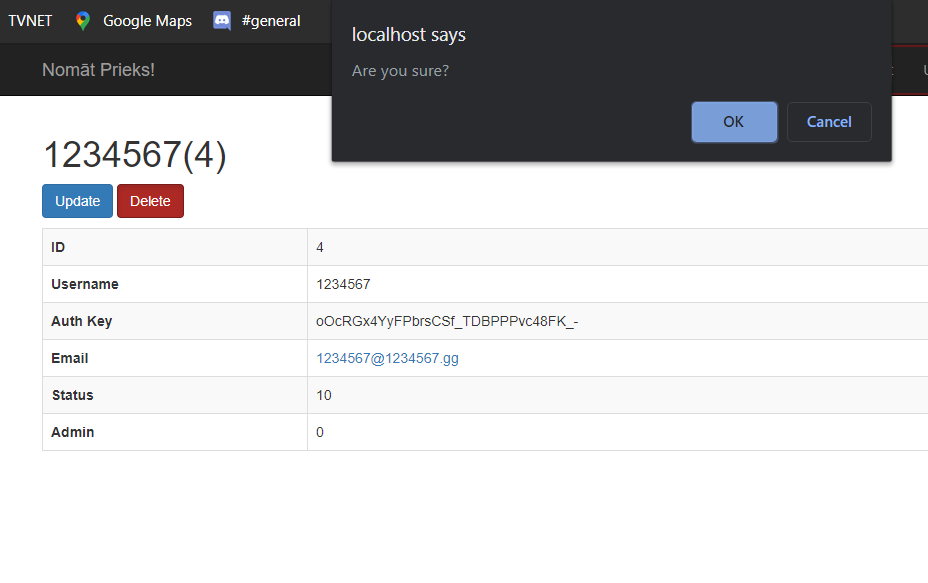
6.9 att. Lietotāju pārvaldīšanas ekrāns

Šādi izskatās lietotāju pārvaldīšanas ekrāns(Skat 6.10. att.) 

6.10 att. Lietotāju pārvaldīšanas attēls

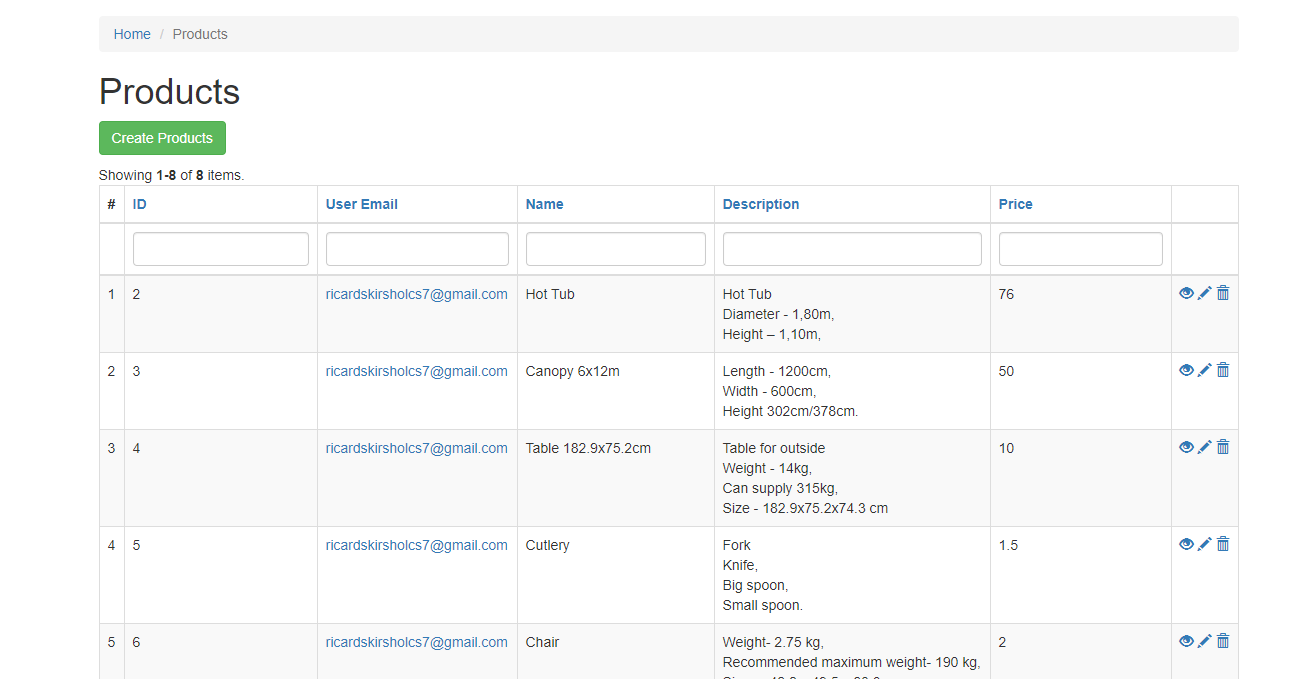
Šādi izskatās lietotāju datu rediģēsanas ekrāns (Skat 6.11. att.) 

6.11. att. Lietotāja datu rediģēsanas attēls

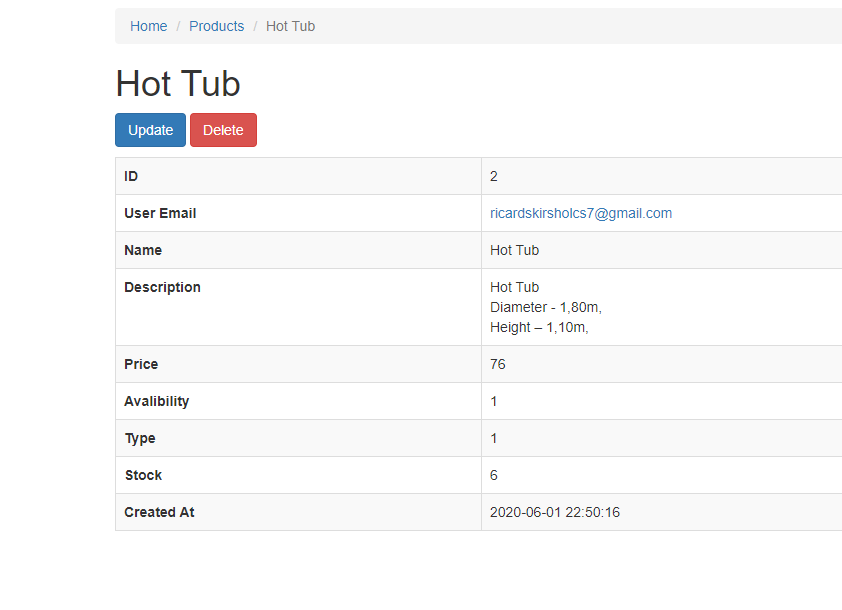
Šādi izskatās lietotāju datu dzēšanas ekrāns (Skat 6.12. att.) 

6.12. att. Lietotāja datu dzēšanas attēls

Šādi izskatās produktu ekrāns(Skat 6.13. att.)

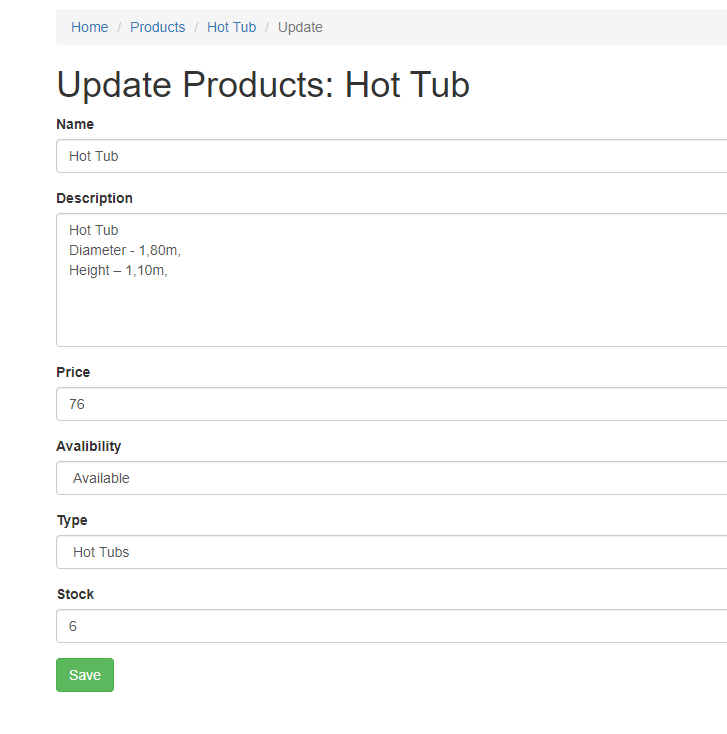


6.13. att. Produktu ekrāna attēls

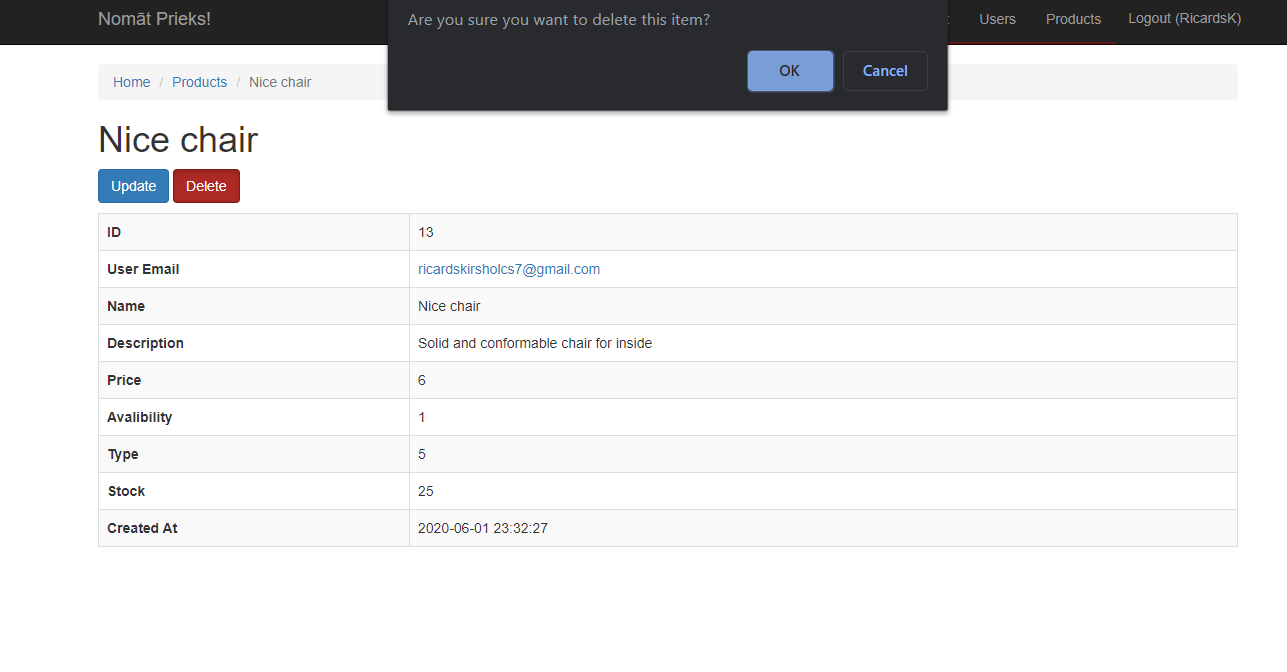
Šādi izskatās produktu apskates ekrāns (Skat 6.14. att.) 

6.14. att. Produktu datu apskates attēls

Šādi izskatās produktu rediģēšanas ekrāns (Skat 6.15. att.)

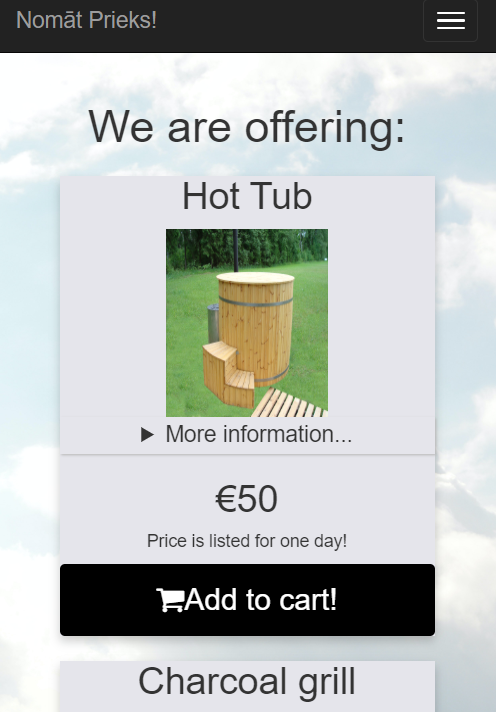


6.15. att. Produktu datu rediģēsanas attēls

Šādi izskatās produktu dzēšanas ekrāns (Skat 6.16. att.) 

6.16. att. Produktu datu dzēšanas attēls

Šādi izskatās pati sistēma kādas viedierīces ekrānā (Skat 6.17. att.)

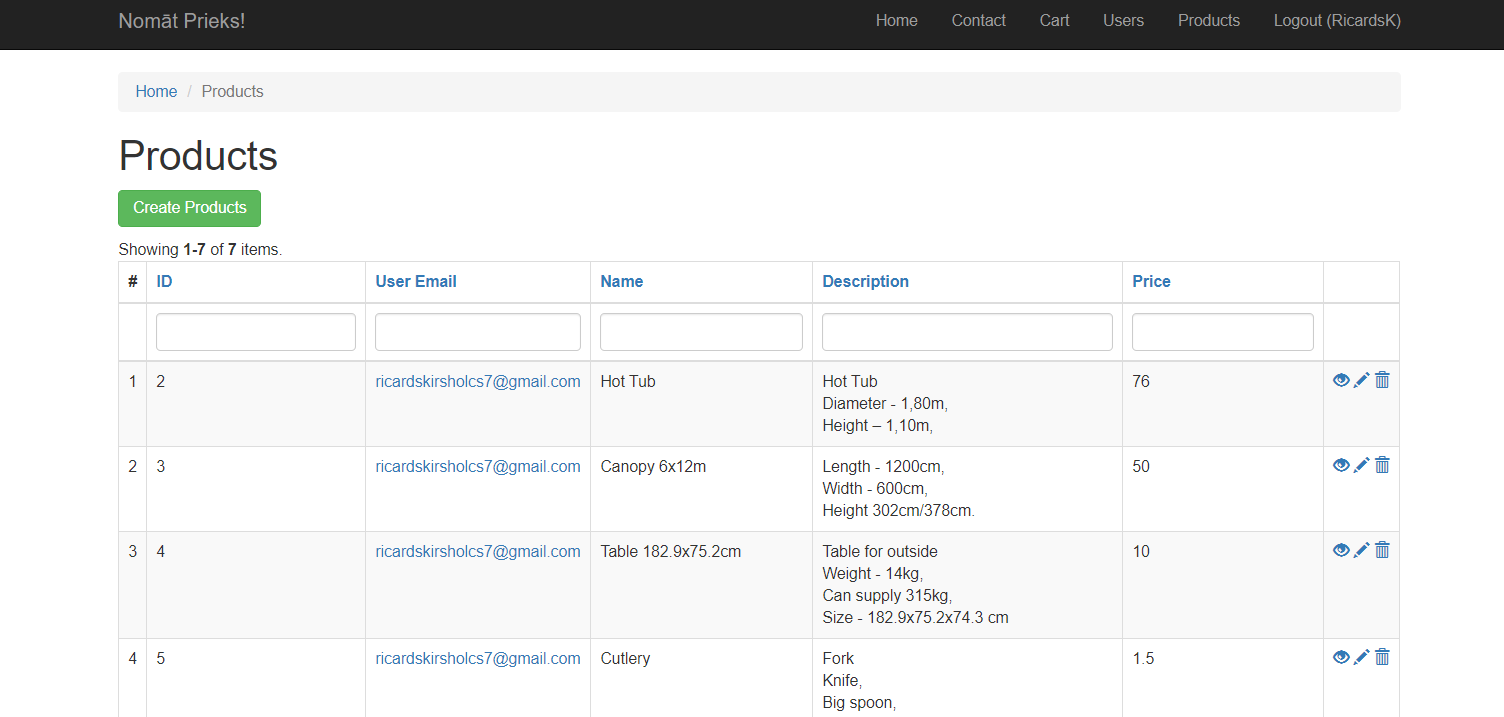


6.17. att. Produktu datu dzēšanas attēls

* 1. **Testa piemērs**

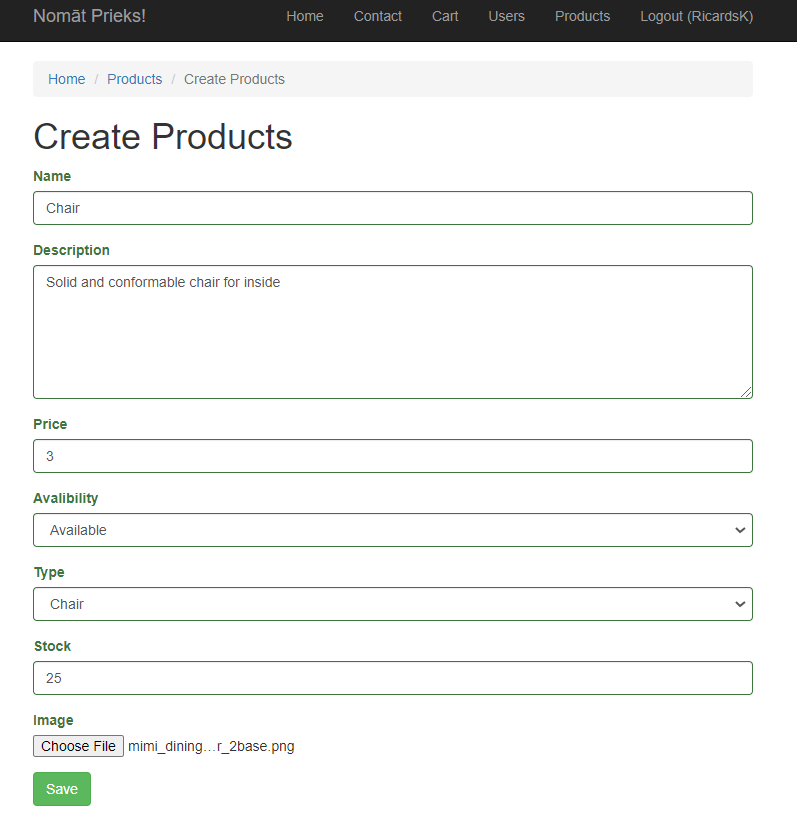
Šajā testa piemērā tiks parādīts, kā administrators var pievienot jaunu produktu un tad to rediģēt.

No sākuma ir jāatver produktu sadaļa un tad jānospiež –“Create Products” (Skat 6.5. att.).



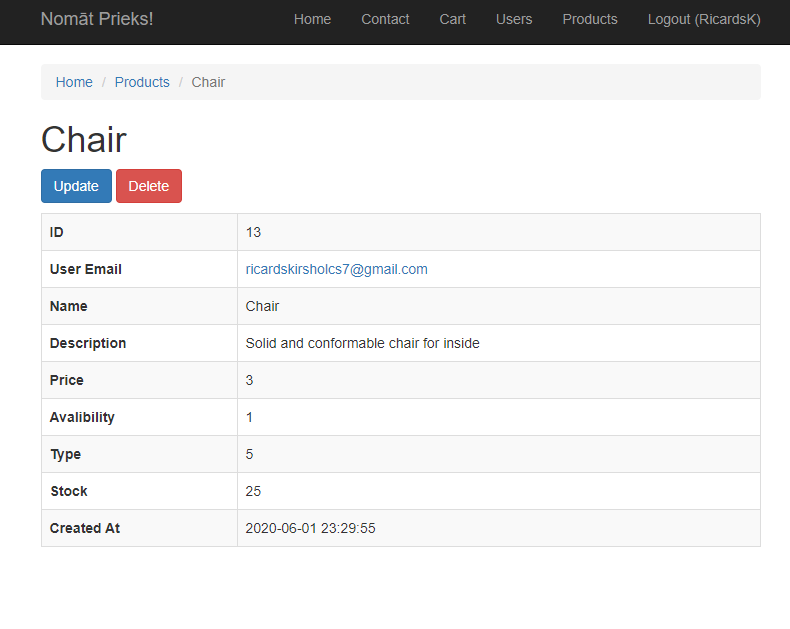
6.5 att. Produktu izveides menu

Kad tas ir izdarīts un visi nepieciešamie lauki ir aizpildīti, tad var izveidot pašu produktu (Skat 6.6. att.)



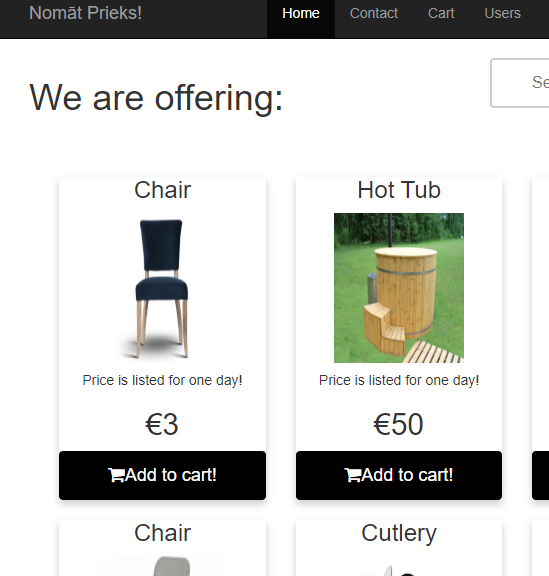
6.6 att. Produkta izveidošanas attēls

Kad sekmīgi ir izveidots produkts parādās ekrāns (Skat 6.7. att.) , kurā ir arī iespēja rediģēt produktu.



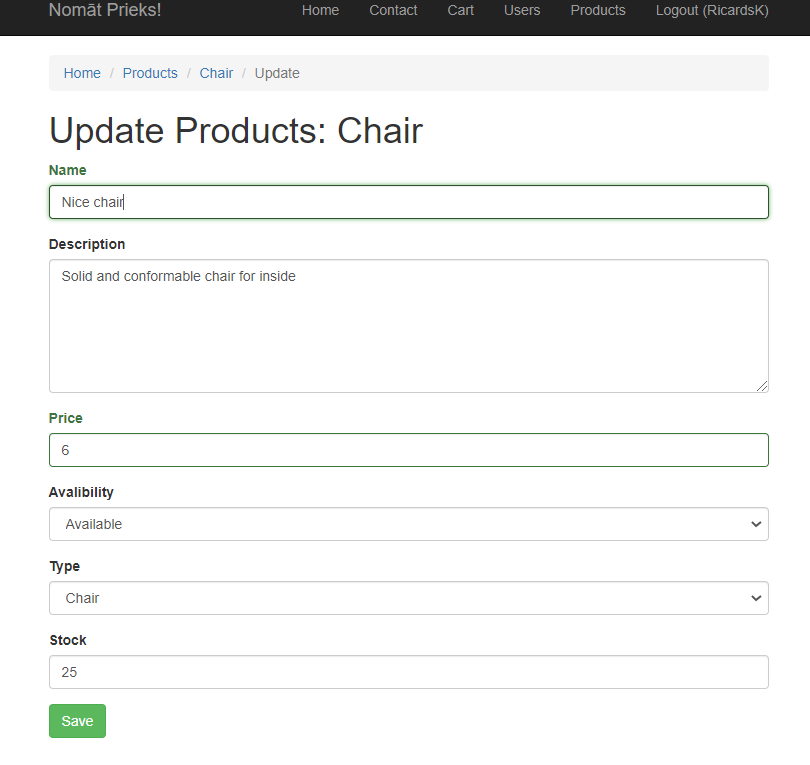
6.7 att. Produkta attēls

Tajā pašā laikā jaunais produkts jau ir pievienots galvanajā lapā (Skat 6.8. att.)



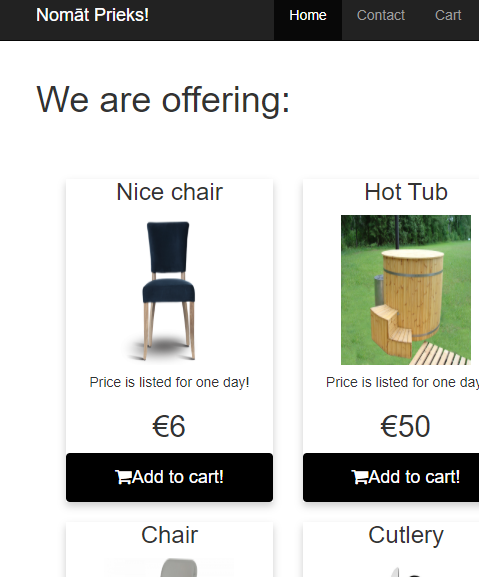
6.8 att. Atrasto braucienu tabulas attēls

Rediģējot produktu parādās šāds ekrāns, kurā ir jāaizpilda visi redzamie lauki un tad jāspiež –“save”. (Skat 6.9. att.)



6.9. att. Produkta rediģēšanas menu

Tajā brīdi tas pats produkts būs redzams jau galvanā lapā un jau redigēts (Skat 6.10. att.)



6.10. att. Rediģēta produkta attēls

**NOBEIGUMS**

Mana kvalifikācijas darba mērķis ir gandrīz sasniegts. Ir izveidota funkcionējoša mājaslapa, kurā var lietotāji reģistrēties, apskatīt sev vēlamos produktus, tos izvēlēties un salikt grozā. Ir iespējama saziņa ar administrāciju nepieciešamības gadījumā un pati mājaslapa ir viegli pārskatāma un saprotama.

Administratoram ir iespēja pārvaldīt visus lietotājus un viņu nešifrētos datus. Kā arī pievienot ir iespēja pievienot mājaslapai jaunus produktus, mainīt to detalizēto informāciju un tos dzēst.

1. **INFORMĀCIJAS AVOTI**
   1. <https://www.yiiframework.com/wiki?category=2&version=2.0>
   2. <https://www.yiiframework.com/doc/guide/2.0/en>
   3. <https://swiftmailer.symfony.com/docs/introduction.html>
   4. <https://www.w3schools.com/howto/default.asp>
   5. [http://push.phpbluedragon.eu//Yii2FrameworkCMS.pdf](http://push.phpbluedragon.eu/Yii2FrameworkCMS.pdf)

**PIELIKUMI**

<?php  
  
*/\** ***@var*** *$this \yii\web\View \*/  
/\** ***@var*** *$content string \*/*use common\models\User;  
use common\widgets\Alert;  
use frontend\assets\AppAsset;  
use yii\bootstrap\Nav;  
use yii\bootstrap\NavBar;  
use yii\helpers\Html;  
use yii\widgets\Breadcrumbs;  
  
AppAsset::*register*($this);  
?>  
<?php $this->beginPage()?>  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="<?=Yii::*$app*->charset?>">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <?php $this->registerCsrfMetaTags()?>  
 <title><?=Html::*encode*($this->title)?></title>  
 <?php $this->head()?>  
</head>  
<body>  
<html>  
<style>  
 body {  
 background-image: url('admin/uploads/background.jpg');  
 background-repeat: no-repeat;  
 background-attachment: fixed;  
 background-size: cover;  
 }  
</style>  
<body>  
<?php $this->beginBody()?>  
  
  
<div class="wrap">  
 <?php  
NavBar::*begin*([  
 'brandLabel' => 'Nomāt Prieks!',  
 'brandUrl' => Yii::*$app*->homeUrl,  
 'options' => [  
 'class' => 'navbar-inverse navbar-fixed-top lightGreen pullDown',  
 ],  
]);  
$menuItems = [  
 ['label' => 'Home', 'url' => ['/site/index']],  
 ['label' => 'Contact', 'url' => ['/site/contact']],  
];  
if (Yii::*$app*->user->isGuest) {  
 $menuItems[] = ['label' => 'Register', 'url' => ['/site/signup']];  
 $menuItems[] = ['label' => 'Login', 'url' => ['/site/login']];  
} else {  
 $menuItems[] = ['label' => 'Cart', 'url' => ['/site/cart']];  
 if (!Yii::*$app*->user->isGuest) {  
 $user = User::*find*()->where(['id' => Yii::*$app*->user->identity])->one();  
 if ($user['admin'] == '1') {  
 $menuItems[] = ['label' => 'Users', 'url' => ['/user']];  
 $menuItems[] = ['label' => 'Products', 'url' => ['/products']];  
 }  
 }  
 $menuItems[] = '<li>'  
 . Html::*beginForm*(['/site/logout'], 'post')  
 . Html::*submitButton*(  
 'Logout (' . Yii::*$app*->user->identity->username . ')',  
 ['class' => 'lightGreen btn btn-link logout']  
 )  
 . Html::*endForm*()  
 . '</li>';  
}  
echo Nav::*widget*([  
 'options' => ['class' => 'navbar-nav navbar-right lightGreen pullDown'],  
 'items' => $menuItems,  
]);  
NavBar::*end*();  
?>  
 <div class="container">  
 <?=Breadcrumbs::*widget*([  
 'links' => isset($this->params['breadcrumbs']) ? $this->params['breadcrumbs'] : [],  
])?>  
 <?=Alert::*widget*()?>  
 <?=$content?>  
 </div>  
</div>  
  
<?php $this->endBody()?>  
</body>  
</html>  
<?php $this->endPage()?>

<?php  
  
use yii\helpers\Html;  
use yii\widgets\ActiveForm;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model frontend\models\Products \*/  
/\** ***@var*** *$form yii\widgets\ActiveForm \*/*?>  
  
<div class="products-form">  
  
 <?php $form = ActiveForm::*begin*(['options'=>['enctype'=>'multipart/form-data']]); ?>  
  
 <?= $form->field($model, 'name')->textInput(['maxlength' => true]) ?>  
  
 <?= $form->field($model, 'description')->textarea(['rows' => 6]) ?>  
  
 <?= $form->field($model, 'price')->textInput(['maxlength' => true]) ?>  
  
 <?= $form->field($model, 'avalibility')->dropDownList([1=>'Available', 0=>'Not in stock']) ?>  
  
 <?= $form->field($model, 'type')->dropDownList([1=>'Hot Tubs',  
 2=>'Canopy',3=>'Tables',4=>'Cutlery',5=>'Chair',6=>'Grills'], ['prompt' => 'Select Item']  
 ) ?>  
  
 <?= $form->field($model, 'stock')->textInput() ?>  
  
 <?= $form->field($model, 'image')->fileInput() ?>  
  
 <div class="form-group">  
 <?= Html::*submitButton*('Save', ['class' => 'btn btn-success']) ?>  
 </div>  
  
 <?php ActiveForm::*end*(); ?>  
  
</div>

<?php  
  
use yii\helpers\Html;  
use yii\widgets\ActiveForm;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model app\models\ProductsSearch \*/  
/\** ***@var*** *$form yii\widgets\ActiveForm \*/*?>  
  
<div class="products-search">  
  
 <?php $form = ActiveForm::*begin*([  
 'action' => ['index'],  
 'method' => 'get',  
 ]); ?>  
  
 <?= $form->field($model, 'id') ?>  
  
 <?= $form->field($model, 'user\_email') ?>  
  
 <?= $form->field($model, 'name') ?>  
  
 <?= $form->field($model, 'description') ?>  
  
 <?= $form->field($model, 'price') ?>  
  
 <?php // echo $form->field($model, 'avalibility') ?>  
  
 <?php // echo $form->field($model, 'type') ?>  
  
 <?php // echo $form->field($model, 'stock') ?>  
  
 <?php // echo $form->field($model, 'image') ?>  
  
 <?php // echo $form->field($model, 'created\_at') ?>  
  
 <div class="form-group">  
 <?= Html::*submitButton*('Search', ['class' => 'btn btn-primary']) ?>  
 <?= Html::*resetButton*('Reset', ['class' => 'btn btn-outline-secondary']) ?>  
 </div>  
  
 <?php ActiveForm::*end*(); ?>  
  
</div>

<?php  
  
use yii\helpers\Html;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model frontend\models\Products \*/*$this->title = 'Create Products';  
$this->params['breadcrumbs'][] = ['label' => 'Products', 'url' => ['index']];  
$this->params['breadcrumbs'][] = $this->title;  
?>  
<div class="products-create">  
  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <?= $this->render('\_form', [  
 'model' => $model,  
 ]) ?>  
  
</div>

<?php  
  
use yii\helpers\Html;  
use yii\grid\GridView;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$searchModel app\models\ProductsSearch \*/  
/\** ***@var*** *$dataProvider yii\data\ActiveDataProvider \*/*$this->title = 'Products';  
$this->params['breadcrumbs'][] = $this->title;  
?>  
<div class="products-index">  
  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <p>  
 <?= Html::*a*('Create Products', ['create'], ['class' => 'btn btn-success']) ?>  
 </p>  
  
 <?php // echo $this->render('\_search', ['model' => $searchModel]); ?>  
  
 <?= GridView::*widget*([  
 'dataProvider' => $dataProvider,  
 'filterModel' => $searchModel,  
 'columns' => [  
 ['class' => 'yii\grid\SerialColumn'],  
  
 'id',  
 'user\_email:email',  
 'name',  
 'description:ntext',  
 'price',  
 //'avalibility',  
 //'type',  
 //'stock',  
 //'image',  
 //'created\_at',  
  
 ['class' => 'yii\grid\ActionColumn'],  
 ],  
 ]); ?>  
  
  
</div>

<?php  
  
use yii\helpers\Html;  
use yii\widgets\ActiveForm;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model frontend\models\Products \*/*$this->title = 'Update Products: ' . $model->name;  
$this->params['breadcrumbs'][] = ['label' => 'Products', 'url' => ['index']];  
$this->params['breadcrumbs'][] = ['label' => $model->name, 'url' => ['view', 'id' => $model->id]];  
$this->params['breadcrumbs'][] = 'Update';  
?>  
<div class="products-update">  
  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <?php $form = ActiveForm::*begin*(['options'=>['enctype'=>'multipart/form-data']]); ?>  
  
 <?= $form->field($model, 'name')->textInput(['maxlength' => true]) ?>  
  
 <?= $form->field($model, 'description')->textarea(['rows' => 6]) ?>  
  
 <?= $form->field($model, 'price')->textInput(['maxlength' => true]) ?>  
  
 <?= $form->field($model, 'avalibility')->dropDownList([1=>'Available', 0=>'Not in stock']) ?>  
  
 <?= $form->field($model, 'type')->dropDownList([1=>'Hot Tubs',  
 2=>'Canopy',3=>'Tables',4=>'Cutlery',5=>'Chair',6=>'Grills'], ['prompt' => 'Select Item']  
 ) ?>  
  
 <?= $form->field($model, 'stock')->textInput() ?>  
  
 <div class="form-group">  
 <?= Html::*submitButton*('Save', ['class' => 'btn btn-success']) ?>  
 </div>  
  
 <?php ActiveForm::*end*(); ?>  
  
</div>

<?php  
  
use yii\helpers\Html;  
use yii\widgets\DetailView;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model frontend\models\Products \*/*$this->title = $model->name;  
$this->params['breadcrumbs'][] = ['label' => 'Products', 'url' => ['index']];  
$this->params['breadcrumbs'][] = $this->title;  
\yii\web\YiiAsset::*register*($this);  
?>  
<div class="products-view">  
  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <p>  
 <?= Html::*a*('Update', ['update', 'id' => $model->id], ['class' => 'btn btn-primary']) ?>  
 <?= Html::*a*('Delete', ['delete', 'id' => $model->id], [  
 'class' => 'btn btn-danger',  
 'data' => [  
 'confirm' => 'Are you sure you want to delete this item?',  
 'method' => 'post',  
 ],  
 ]) ?>  
 </p>  
  
 <?= DetailView::*widget*([  
 'model' => $model,  
 'attributes' => [  
 'id',  
 'user\_email:email',  
 'name',  
 'description:ntext',  
 'price',  
 'avalibility',  
 'type',  
 'stock',  
 // 'image',  
 'created\_at',  
 ],  
 ]) ?>  
  
</div>

<?php  
use yii\helpers\Url;  
?>  
<!DOCTYPE html>  
<html>  
<head>  
 <style>  
  
 details > summary {  
 padding: 2px;  
 border: none;  
 box-shadow: 1px 1px 2px #bbbbbb;  
 cursor: pointer;  
 font-size: 18px;  
 }  
  
 details > p {  
 background-color: #eeeeee;  
 padding: 4px;  
 margin: 0;  
 box-shadow: 1px 1px 2px #bbbbbb;  
 font-size: 18px;  
 }  
  
 .card {  
 box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);  
 max-width: 300px;  
 margin: auto;  
 text-align: center;  
 font-family: arial;  
 background-color: #E5E5EB;  
 }  
  
  
  
 .card button {  
 border: none;  
 outline: 0;  
 padding: 12px;  
 color: white;  
 background-color: #000;  
 text-align: center;  
 cursor: pointer;  
 width: 100%;  
 font-size: 24px;  
 }  
  
  
 .card button:hover {  
 opacity: 0.9;  
 }  
 </style>  
</head>  
  
  
<div class="col-sm-4">  
 <div class="single-products">  
 <div class="card">  
 <?php if(yii::*$app*->user->isGuest) {?>  
 <h2><?=$model->name?></h2>  
 <img src="admin/uploads/<?=$model->image?>" alt="" width="130" height="150"/>  
 <details>  
 <summary>More information...</summary>  
 <p><?=$model->description?></p>  
 </details>  
 <h2>€<?=$model->price?></h2>  
 <h5>Price is listed for one day!</h5>  
 <a href="/lapa/index.php?r=site%2Flogin">Login</a>  
 <?php }else{ ?>  
 <h2><?=$model->name?></h2>  
 <img src="admin/uploads/<?=$model->image?>" alt="" width="130" height="150"/>  
 <details>  
 <summary>More information...</summary>  
 <p><?=$model->description?></p>  
 </details>  
 <h2>€<?=$model->price?></h2>  
 <h5>Price is listed for one day!</h5>  
 <form action="<?=Url::*to*('/lapa/index.php?r=site%2Fcart')?>" method="post">  
 <input name="pid" type="hidden" value="<?=$model->id?>">  
 <button class="btn btn-default"><i class="fa fa-shopping-cart"></i>Add to cart!</button>  
 </form>  
 <?php } ?>  
 </div>  
</div>

<?php  
use frontend\models\Products;  
use yii\helpers\Url;  
use yii\helpers\Html;  
  
$sum = 0;  
$i = 0;  
?>  
  
<section id="cart">  
 <div class="container">  
 <table class="table table-condensed">  
 <thead>  
 <tr class="card-title" style = 'font-weight:bold;'>  
 <td class="image">Image</td>  
 <td class="name">Name</td>  
 <td class="description">Description</td>  
 <td class="price">Price</td>  
 <td></td>  
 </tr>  
 </thead>  
 <tbody>  
 <?php  
 if(!empty($test)) {  
 foreach ($test as $key => $data) {  
 $product = Products::*find*()->where(['id'=> $data])->one();  
  
 $sum += $product->price;  
 ?>  
 <tr>  
 <td class="cart\_product">  
 <a href=""><img src="admin/uploads/<?=$product->image?>" alt="" width="100" height="100"></a>  
 </td>  
 <td class="cart\_name">  
 <h4><?=$product->name?></a></h4>  
 </td>  
 <td class="cart\_description">  
 <p><?=$product->description?></p>  
 </td>  
 <td class="cart\_price">  
 <p>€<?=$product->price?></p>  
 </td>  
 <td>  
 <?php  
 echo Html::*a*(  
 'Remove',  
 ['/site/cart', 'id' =>$key],  
 ['class' => 'btn btn-danger']  
 );  
 ?>  
  
 </td>  
 </tr>  
 <?php }  
 } ?>  
 <tr>  
 <td class = 'text-right' colspan = '3' style = 'font-weight:bold;'>  
 Total:  
 </td>  
 <td style = 'font-weight:bold;'>  
 <?= '€'.$sum ?>  
 </td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
</section>

<?php  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$form yii\bootstrap\ActiveForm \*/  
/\** ***@var*** *$model \frontend\models\ContactForm \*/*use yii\helpers\Html;  
use yii\bootstrap\ActiveForm;  
use yii\captcha\Captcha;  
  
$this->title = 'Contact';  
?>  
<div class="site-contact">  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <p>  
 If you have any questions, you can contact us here:  
 </p>  
  
 <div class="row">  
 <div class="col-lg-5">  
 <?php $form = ActiveForm::*begin*(['id' => 'contact-form']); ?>  
  
 <?= $form->field($model, 'name')->textInput(['autofocus' => true]) ?>  
  
 <?= $form->field($model, 'email') ?>  
  
 <?= $form->field($model, 'subject') ?>  
  
 <?= $form->field($model, 'body')->textarea(['rows' => 6]) ?>  
  
 <?= $form->field($model, 'verifyCode')->widget(Captcha::*className*(), [  
 'template' => '<div class="row"><div class="col-lg-3">{image}</div><div class="col-lg-6">{input}</div></div>',  
 ]) ?>  
  
 <div class="form-group">  
 <?= Html::*submitButton*('Submit', ['class' => 'btn btn-primary', 'name' => 'contact-button']) ?>  
 </div>  
  
 <?php ActiveForm::*end*(); ?>  
 </div>  
 </div>  
  
</div>

<?php  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$name string \*/  
/\** ***@var*** *$message string \*/  
/\** ***@var*** *$exception Exception \*/*use yii\helpers\Html;  
  
$this->title = $name;  
?>  
<div class="site-error">  
  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <div class="alert alert-danger">  
 <?= nl2br(Html::*encode*($message)) ?>  
 </div>  
  
 <p>  
 The above error occurred while the Web server was processing your request.  
 </p>  
 <p>  
 Please contact us if you think this is a server error. Thank you.  
 </p>  
  
</div>

<?php  
use yii\widgets\LinkPager;  
use yii\grid\ListView;  
use yii\helpers\Url;  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$searchModel app\models\ProductsSearch \*/  
/\** ***@var*** *$dataProvider yii\data\ActiveDataProvider \*/*$this->title = 'NomātPrieks!';  
?>  
<!DOCTYPE html>  
<html>  
<head>  
 <style>  
 h1 {text-align: center;}  
  
  
 }  
 .topnav .search-container {  
 float: right;  
 }  
 input[type=text] {  
 /\* width: 130px; \*/  
 box-sizing: border-box;  
 border: 2px solid #ccc;  
 border-radius: 4px;  
 font-size: 16px;  
 background-color: white;  
 background-image: url();  
 background-position: 10px 10px;  
 background-repeat: no-repeat;  
 padding: 12px 20px 12px 40px;  
 -webkit-transition: width 0.4s ease-in-out;  
 transition: width 0.4s ease-in-out;  
 }  
  
 input[type=text]:focus {  
 width: 100%;  
 }  
 .topnav .search-container button {  
 float: right;  
 padding: 6px 10px;  
 margin-top: 2px;  
 margin-right: 16px;  
 background: gray;  
 font-size: 18px;  
 border: 2px;  
 cursor: pointer;  
 }  
 </style>  
</head>  
<head>  
 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
</head>  
  
  
<h1 class="topnav">We are offering:</h1>  
  
<div class="topnav">  
 <div class="flex-column">  
  
 <?php  
 echo \yii\widgets\ListView::*widget*([  
 'dataProvider'=>$products,  
 'layout'=>"<div class\"items\">{items}</div>",  
 'itemView'=>'\_product',  
 ])  
 ?>  
 </div>  
</div>

<?php  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$form yii\bootstrap\ActiveForm \*/  
/\** ***@var*** *$model \common\models\LoginForm \*/*use yii\helpers\Html;  
use yii\bootstrap\ActiveForm;  
  
$this->title = 'Login';  
?>  
<div class="site-login">  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <p>Please fill out the following fields to login:</p>  
  
 <div class="row">  
 <div class="col-lg-5">  
 <?php $form = ActiveForm::*begin*(['id' => 'login-form']); ?>  
  
 <?= $form->field($model, 'username')->textInput(['autofocus' => true]) ?>  
  
 <?= $form->field($model, 'password')->passwordInput() ?>  
  
 <?= $form->field($model, 'rememberMe')->checkbox() ?>  
  
 <div style="color:#999;margin:1em 0">  
 If you forgot your password you can <?= Html::*a*('reset it', ['site/request-password-reset']) ?>.  
 <br>  
 Need new verification email? <?= Html::*a*('Resend', ['site/resend-verification-email']) ?>  
 </div>  
  
 <div class="form-group">  
 <?= Html::*submitButton*('Login', ['class' => 'btn btn-primary', 'name' => 'login-button']) ?>  
 </div>  
  
 <?php ActiveForm::*end*(); ?>  
 </div>  
 </div>  
</div>

<?php  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$form yii\bootstrap\ActiveForm \*/  
/\** ***@var*** *$model \frontend\models\PasswordResetRequestForm \*/*use yii\helpers\Html;  
use yii\bootstrap\ActiveForm;  
  
$this->title = 'Request password reset';  
$this->params['breadcrumbs'][] = $this->title;  
?>  
<div class="site-request-password-reset">  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <p>Please fill out your email. A link to reset password will be sent there.</p>  
  
 <div class="row">  
 <div class="col-lg-5">  
 <?php $form = ActiveForm::*begin*(['id' => 'request-password-reset-form']); ?>  
  
 <?= $form->field($model, 'email')->textInput(['autofocus' => true]) ?>  
  
 <div class="form-group">  
 <?= Html::*submitButton*('Send', ['class' => 'btn btn-primary']) ?>  
 </div>  
  
 <?php ActiveForm::*end*(); ?>  
 </div>  
 </div>  
</div>

<?php  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$form yii\bootstrap\ActiveForm \*/  
/\** ***@var*** *$model \frontend\models\ResetPasswordForm \*/*use yii\helpers\Html;  
use yii\bootstrap\ActiveForm;  
  
$this->title = 'Resend verification email';  
?>  
<div class="site-resend-verification-email">  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <p>Please fill out your email. A verification email will be sent there.</p>  
  
 <div class="row">  
 <div class="col-lg-5">  
 <?php $form = ActiveForm::*begin*(['id' => 'resend-verification-email-form']); ?>  
  
 <?= $form->field($model, 'email')->textInput(['autofocus' => true]) ?>  
  
 <div class="form-group">  
 <?= Html::*submitButton*('Send', ['class' => 'btn btn-primary']) ?>  
 </div>  
  
 <?php ActiveForm::*end*(); ?>  
 </div>  
 </div>  
</div>

<?php  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$form yii\bootstrap\ActiveForm \*/  
/\** ***@var*** *$model \frontend\models\ResetPasswordForm \*/*use yii\helpers\Html;  
use yii\bootstrap\ActiveForm;  
  
$this->title = 'Reset password';  
?>  
<div class="site-reset-password">  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <p>Please choose your new password:</p>  
  
 <div class="row">  
 <div class="col-lg-5">  
 <?php $form = ActiveForm::*begin*(['id' => 'reset-password-form']); ?>  
  
 <?= $form->field($model, 'password')->passwordInput(['autofocus' => true]) ?>  
  
 <div class="form-group">  
 <?= Html::*submitButton*('Save', ['class' => 'btn btn-primary']) ?>  
 </div>  
  
 <?php ActiveForm::*end*(); ?>  
 </div>  
 </div>  
</div>

<?php  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$form yii\bootstrap\ActiveForm \*/  
/\** ***@var*** *$model \frontend\models\SignupForm \*/*use yii\helpers\Html;  
use yii\bootstrap\ActiveForm;  
  
$this->title = 'Signup';  
?>  
<div class="site-signup">  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <p>Please fill out the following fields to signup:</p>  
  
 <div class="row">  
 <div class="col-lg-5">  
 <?php $form = ActiveForm::*begin*(['id' => 'form-signup']); ?>  
  
 <?= $form->field($model, 'username')->textInput(['autofocus' => true]) ?>  
  
 <?= $form->field($model, 'email') ?>  
  
 <?= $form->field($model, 'password')->passwordInput() ?>  
  
 <div class="form-group">  
 <?= Html::*submitButton*('Signup', ['class' => 'btn btn-primary', 'name' => 'signup-button']) ?>  
 </div>  
  
 <?php ActiveForm::*end*(); ?>  
 </div>  
 </div>  
</div>

<?php  
  
use yii\helpers\Html;  
use yii\widgets\ActiveForm;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model app\models\User \*/  
/\** ***@var*** *$form yii\widgets\ActiveForm \*/*?>  
  
<div class="user-form">  
  
 <?php $form = ActiveForm::*begin*(); ?>  
  
 <?= $form->field($model, 'username')->textInput(['maxlength' => true]) ?>  
  
 <?= $form->field($model, 'auth\_key')->textInput(['maxlength' => true]) ?>  
  
 <!-- <?= $form->field($model, 'password\_hash')->textInput(['maxlength' => true]) ?> -->  
  
 <!-- <?= $form->field($model, 'password\_reset\_token')->textInput(['maxlength' => true]) ?> -->  
  
 <?= $form->field($model, 'email')->textInput(['maxlength' => true]) ?>  
  
 <?= $form->field($model, 'status')->textInput() ?>  
  
 <!-- <?= $form->field($model, 'created\_at')->textInput() ?> -->  
  
 <!-- <?= $form->field($model, 'updated\_at')->textInput() ?> -->  
  
 <!-- <?= $form->field($model, 'verification\_token')->textInput(['maxlength' => true]) ?> -->  
  
 <?= $form->field($model, 'admin')->textInput() ?>  
  
 <div class="form-group">  
 <?= Html::*submitButton*('Saglabāt', ['class' => 'btn btn-success']) ?>  
 </div>  
  
 <?php ActiveForm::*end*(); ?>  
  
</div>

<?php  
  
use yii\helpers\Html;  
use yii\widgets\ActiveForm;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model app\models\UserSearch \*/  
/\** ***@var*** *$form yii\widgets\ActiveForm \*/*?>  
  
<div class="user-search">  
  
 <?php $form = ActiveForm::*begin*([  
 'action' => ['index'],  
 'method' => 'get',  
 ]); ?>  
  
 <?= $form->field($model, 'id') ?>  
  
 <?= $form->field($model, 'username') ?>  
  
 <?= $form->field($model, 'auth\_key') ?>  
  
 <?= $form->field($model, 'password\_hash') ?>  
  
 <?= $form->field($model, 'password\_reset\_token') ?>  
  
 <?php // echo $form->field($model, 'email') ?>  
  
 <?php // echo $form->field($model, 'status') ?>  
  
 <?php // echo $form->field($model, 'created\_at') ?>  
  
 <?php // echo $form->field($model, 'updated\_at') ?>  
  
 <?php // echo $form->field($model, 'verification\_token') ?>  
  
 <?php // echo $form->field($model, 'admin') ?>  
  
 <div class="form-group">  
 <?= Html::*submitButton*('Search', ['class' => 'btn btn-primary']) ?>  
 <?= Html::*resetButton*('Reset', ['class' => 'btn btn-outline-secondary']) ?>  
 </div>  
  
 <?php ActiveForm::*end*(); ?>  
  
</div>

<?php  
  
use yii\helpers\Html;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model app\models\User \*/*$this->title = 'New user';  
// $this->params['breadcrumbs'][] = ['label' => 'Users', 'url' => ['index']];  
// $this->params['breadcrumbs'][] = $this->title;  
?>  
<div class="user-create">  
  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <?= $this->render('\_form', [  
 'model' => $model,  
 ]) ?>  
  
</div>

<?php  
  
use yii\helpers\Html;  
use yii\grid\GridView;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$searchModel app\models\UserSearch \*/  
/\** ***@var*** *$dataProvider yii\data\ActiveDataProvider \*/*$this->title = 'Users';  
// $this->params['breadcrumbs'][] = $this->title;  
?>  
<div class="user-index">  
  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <!-- <p>  
 <?= Html::*a*('Create User', ['create'], ['class' => 'btn btn-success']) ?>  
 </p> -->  
  
 <?php // echo $this->render('\_search', ['model' => $searchModel]); ?>  
  
 <?= GridView::*widget*([  
 'dataProvider' => $dataProvider,  
 'filterModel' => $searchModel,  
 'columns' => [  
 ['class' => 'yii\grid\SerialColumn'],  
  
 'id',  
 'username',  
 // 'auth\_key',  
 // 'password\_hash',  
 // 'password\_reset\_token',  
 'email:email',  
 'status',  
 //'created\_at',  
 //'updated\_at',  
 //'verification\_token',  
 'admin',  
  
 ['class' => 'yii\grid\ActionColumn'],  
 ],  
 ]); ?>  
  
  
</div>

<?php  
  
use yii\helpers\Html;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model app\models\User \*/*$this->title = 'Update user: ' . $model->username;  
// $this->params['breadcrumbs'][] = ['label' => 'Users', 'url' => ['index']];  
// $this->params['breadcrumbs'][] = ['label' => $model->id, 'url' => ['view', 'id' => $model->id]];  
// $this->params['breadcrumbs'][] = 'Update';  
?>  
<div class="user-update">  
  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <?= $this->render('\_form', [  
 'model' => $model,  
 ]) ?>  
  
</div>

<?php  
  
use yii\helpers\Html;  
use yii\widgets\DetailView;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model app\models\User \*/*$this->title = $model->username . '(' . $model->id . ')';  
// $this->params['breadcrumbs'][] = ['label' => 'Users', 'url' => ['index']];  
// $this->params['breadcrumbs'][] = $this->title;  
\yii\web\YiiAsset::*register*($this);  
?>  
<div class="user-view">  
  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <p>  
 <?= Html::*a*('Update', ['update', 'id' => $model->id], ['class' => 'btn btn-primary']) ?>  
 <?= Html::*a*('Delete', ['delete', 'id' => $model->id], [  
 'class' => 'btn btn-danger',  
 'data' => [  
 'confirm' => 'Are you sure?',  
 'method' => 'post',  
 ],  
 ]) ?>  
 </p>  
  
 <?= DetailView::*widget*([  
 'model' => $model,  
 'attributes' => [  
 'id',  
 'username',  
 'auth\_key',  
 // 'password\_hash',  
 // 'password\_reset\_token',  
 'email:email',  
 'status',  
 // 'created\_at',  
 // 'updated\_at',  
 // 'verification\_token',  
 'admin',  
 ],  
 ]) ?>  
  
</div>

<?php  
  
namespace frontend\models;  
  
use Yii;  
use yii\base\Model;  
use yii\db\ActiveRecord;  
  
*/\*\*  
 \* ContactForm is the model behind the contact form.  
 \*/*class ContactForm extends ActiveRecord  
{  
  
 public $verifyCode;  
  
 public static function tableName()  
 {  
 return 'contact\_form';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 // name, email, subject and body are required  
 [['name', 'email', 'subject', 'body'], 'required'],  
 // email has to be a valid email address  
 ['email', 'email'],  
 // verifyCode needs to be entered correctly  
 ['verifyCode', 'captcha'],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function attributeLabels()  
 {  
 return [  
 'verifyCode' => 'Verification Code',  
 ];  
 }  
  
 */\*\*  
 \* Sends an email to the specified email address using the information collected by this model.  
 \*  
 \** ***@param*** *string $email the target email address  
 \** ***@return*** *bool whether the email was sent  
 \*/* public function sendEmail($email)  
 {  
 $Model = new ContactForm();  
 $Model ->name = $this->name;  
 $Model ->email = $this->email;  
 $Model ->subject = $this->subject;  
 $Model ->body = $this->body;  
 $Model ->save(false);  
  
  
 return Yii::*$app*->mailer->compose()  
 ->setTo($email)  
 ->setFrom([Yii::*$app*->params['senderEmail'] => Yii::*$app*->params['senderName']])  
 ->setReplyTo([$this->email => $this->name])  
 ->setSubject($this->subject)  
 ->setTextBody($this->body)  
 ->send();  
 }  
}

<?php  
namespace frontend\models;  
  
use Yii;  
use yii\base\Model;  
use common\models\User;  
  
*/\*\*  
 \* Password reset request form  
 \*/*class PasswordResetRequestForm extends Model  
{  
 public $email;  
  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 ['email', 'trim'],  
 ['email', 'required'],  
 ['email', 'email'],  
 ['email', 'exist',  
 'targetClass' => '\common\models\User',  
 'filter' => ['status' => User::*STATUS\_ACTIVE*],  
 'message' => 'There is no user with this email address.'  
 ],  
 ];  
 }  
  
 */\*\*  
 \* Sends an email with a link, for resetting the password.  
 \*  
 \** ***@return*** *bool whether the email was send  
 \*/* public function sendEmail()  
 {  
 */\** ***@var*** *$user User \*/* $user = User::*findOne*([  
 'status' => User::*STATUS\_ACTIVE*,  
 'email' => $this->email,  
 ]);  
  
 if (!$user) {  
 return false;  
 }  
   
 if (!User::*isPasswordResetTokenValid*($user->password\_reset\_token)) {  
 $user->generatePasswordResetToken();  
 if (!$user->save()) {  
 return false;  
 }  
 }  
  
 return Yii::*$app* ->mailer  
 ->compose(  
 ['html' => 'passwordResetToken-html', 'text' => 'passwordResetToken-text'],  
 ['user' => $user]  
 )  
 ->setFrom([Yii::*$app*->params['supportEmail'] => Yii::*$app*->name . ' robot'])  
 ->setTo($this->email)  
 ->setSubject('Password reset for ' . Yii::*$app*->name)  
 ->send();  
 }  
}

<?php  
  
namespace frontend\models;  
  
use Yii;  
  
  
*/\*\*  
 \* This is the model class for table "products".  
 \*  
 \** ***@property*** *int $id  
 \** ***@property*** *string $user\_email  
 \** ***@property*** *string $name  
 \** ***@property*** *string $description  
 \** ***@property*** *string $price  
 \** ***@property*** *string $avalibility  
 \** ***@property*** *string $type  
 \** ***@property*** *int $stock  
 \** ***@property*** *string $image  
 \** ***@property*** *string $created\_at  
 \*/*class Products extends \yii\db\ActiveRecord {  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function tableName()  
 {  
 return 'products';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 [['name', 'description', 'price', 'avalibility', 'type', 'stock', 'image'], 'required','skipOnEmpty'=>false],  
 [['description'], 'string'],  
 [['stock'], 'integer'],  
 [['created\_at','user\_email'], 'safe'],  
 [['user\_email', 'avalibility', 'type'], 'string', 'max' => 100],  
 [['name', 'image'], 'string', 'max' => 255],  
 [['price'], 'string', 'max' => 50],  
 [['image'], 'file','extensions'=>'jpg,png,gif','skipOnEmpty'=>false],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function attributeLabels()  
 {  
 return [  
 'id' => 'ID',  
 'user\_email' => 'User Email',  
 'name' => 'Name',  
 'description' => 'Description',  
 'price' => 'Price',  
 'avalibility' => 'Avalibility',  
 'type' => 'Type',  
 'stock' => 'Stock',  
 'image' => 'Image',  
 'created\_at' => 'Created At',  
 ];  
 }  
}

<?php  
  
namespace app\models;  
  
use yii\base\Model;  
use yii\data\ActiveDataProvider;  
use frontend\models\Products;  
  
*/\*\*  
 \* ProductsSearch represents the model behind the search form of `frontend\models\Products`.  
 \*/*class ProductsSearch extends Products  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 [['id', 'stock'], 'integer'],  
 [['user\_email', 'name', 'description', 'price', 'avalibility', 'type', 'image', 'created\_at'], 'safe'],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function scenarios()  
 {  
 // bypass scenarios() implementation in the parent class  
 return Model::*scenarios*();  
 }  
  
 */\*\*  
 \* Creates data provider instance with search query applied  
 \*  
 \** ***@param*** *array $params  
 \*  
 \** ***@return*** *ActiveDataProvider  
 \*/* public function search($params)  
 {  
 $query = Products::*find*();  
  
 // add conditions that should always apply here  
  
 $dataProvider = new ActiveDataProvider([  
 'query' => $query,  
 ]);  
  
 $this->load($params);  
  
 if (!$this->validate()) {  
 // uncomment the following line if you do not want to return any records when validation fails  
 // $query->where('0=1');  
 return $dataProvider;  
 }  
  
 // grid filtering conditions  
 $query->andFilterWhere([  
 'id' => $this->id,  
 'stock' => $this->stock,  
 'created\_at' => $this->created\_at,  
 ]);  
  
 $query->andFilterWhere(['like', 'user\_email', $this->user\_email])  
 ->andFilterWhere(['like', 'name', $this->name])  
 ->andFilterWhere(['like', 'description', $this->description])  
 ->andFilterWhere(['like', 'price', $this->price])  
 ->andFilterWhere(['like', 'avalibility', $this->avalibility])  
 ->andFilterWhere(['like', 'type', $this->type])  
 ->andFilterWhere(['like', 'image', $this->image]);  
  
 return $dataProvider;  
 }  
}

<?php  
  
  
namespace frontend\models;  
  
use Yii;  
use common\models\User;  
use yii\base\Model;  
  
class ResendVerificationEmailForm extends Model  
{  
 */\*\*  
 \** ***@var*** *string  
 \*/* public $email;  
  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 ['email', 'trim'],  
 ['email', 'required'],  
 ['email', 'email'],  
 ['email', 'exist',  
 'targetClass' => '\common\models\User',  
 'filter' => ['status' => User::*STATUS\_INACTIVE*],  
 'message' => 'There is no user with this email address.'  
 ],  
 ];  
 }  
  
 */\*\*  
 \* Sends confirmation email to user  
 \*  
 \** ***@return*** *bool whether the email was sent  
 \*/* public function sendEmail()  
 {  
 $user = User::*findOne*([  
 'email' => $this->email,  
 'status' => User::*STATUS\_INACTIVE* ]);  
  
 if ($user === null) {  
 return false;  
 }  
  
 return Yii::*$app* ->mailer  
 ->compose(  
 ['html' => 'emailVerify-html', 'text' => 'emailVerify-text'],  
 ['user' => $user]  
 )  
 ->setFrom([Yii::*$app*->params['supportEmail'] => Yii::*$app*->name . ' robot'])  
 ->setTo($this->email)  
 ->setSubject('Account registration at ' . Yii::*$app*->name)  
 ->send();  
 }  
}

<?php  
namespace frontend\models;  
  
use yii\base\InvalidArgumentException;  
use yii\base\Model;  
use common\models\User;  
  
*/\*\*  
 \* Password reset form  
 \*/*class ResetPasswordForm extends Model  
{  
 public $password;  
  
 */\*\*  
 \** ***@var*** *\common\models\User  
 \*/* private $\_user;  
  
  
 */\*\*  
 \* Creates a form model given a token.  
 \*  
 \** ***@param*** *string $token  
 \** ***@param*** *array $config name-value pairs that will be used to initialize the object properties  
 \** ***@throws*** *InvalidArgumentException if token is empty or not valid  
 \*/* public function \_\_construct($token, $config = [])  
 {  
 if (empty($token) || !is\_string($token)) {  
 throw new InvalidArgumentException('Password reset token cannot be blank.');  
 }  
 $this->\_user = User::*findByPasswordResetToken*($token);  
 if (!$this->\_user) {  
 throw new InvalidArgumentException('Wrong password reset token.');  
 }  
 parent::*\_\_construct*($config);  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 ['password', 'required'],  
 ['password', 'string', 'min' => 6],  
 ];  
 }  
  
 */\*\*  
 \* Resets password.  
 \*  
 \** ***@return*** *bool if password was reset.  
 \*/* public function resetPassword()  
 {  
 $user = $this->\_user;  
 $user->setPassword($this->password);  
 $user->removePasswordResetToken();  
  
 return $user->save(false);  
 }  
}

<?php  
namespace frontend\models;  
  
use Yii;  
use yii\base\Model;  
use common\models\User;  
  
*/\*\*  
 \* Signup form  
 \*/*class SignupForm extends Model  
{  
 public $username;  
 public $email;  
 public $password;  
  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 ['username', 'trim'],  
 ['username', 'required'],  
 ['username', 'unique', 'targetClass' => '\common\models\User', 'message' => 'This username has already been taken.'],  
 ['username', 'string', 'min' => 2, 'max' => 255],  
  
 ['email', 'trim'],  
 ['email', 'required'],  
 ['email', 'email'],  
 ['email', 'string', 'max' => 255],  
 ['email', 'unique', 'targetClass' => '\common\models\User', 'message' => 'This email address has already been taken.'],  
  
 ['password', 'required'],  
 ['password', 'string', 'min' => 6],  
 ];  
 }  
  
 */\*\*  
 \* Signs user up.  
 \*  
 \** ***@return*** *bool whether the creating new account was successful and email was sent  
 \*/* public function signup()  
 {  
 if (!$this->validate()) {  
 return null;  
 }  
   
 $user = new User();  
 $user->username = $this->username;  
 $user->email = $this->email;  
 $user->setPassword($this->password);  
 $user->generateAuthKey();  
 $user->generateEmailVerificationToken();  
 return $user->save() && $this->sendEmail($user);  
  
 }  
  
 */\*\*  
 \* Sends confirmation email to user  
 \** ***@param*** *User $user user model to with email should be send  
 \** ***@return*** *bool whether the email was sent  
 \*/* protected function sendEmail($user)  
 {  
 return Yii::*$app* ->mailer  
 ->compose(  
 ['html' => 'emailVerify-html', 'text' => 'emailVerify-text'],  
 ['user' => $user]  
 )  
 ->setFrom([Yii::*$app*->params['supportEmail'] => Yii::*$app*->name . ' robot'])  
 ->setTo($this->email)  
 ->setSubject('Account registration at ' . Yii::*$app*->name)  
 ->send();  
 }  
}

<?php  
  
namespace app\models;  
  
use Yii;  
  
*/\*\*  
 \* This is the model class for table "user".  
 \*  
 \** ***@property*** *int $id  
 \** ***@property*** *string $username  
 \** ***@property*** *string $auth\_key  
 \** ***@property*** *string $password\_hash  
 \** ***@property*** *string|null $password\_reset\_token  
 \** ***@property*** *string $email  
 \** ***@property*** *int $status  
 \** ***@property*** *int $created\_at  
 \** ***@property*** *int $updated\_at  
 \** ***@property*** *string|null $verification\_token  
 \** ***@property*** *int $admin  
 \*/*class User extends \yii\db\ActiveRecord  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function tableName()  
 {  
 return 'user';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 [['username', 'auth\_key', 'password\_hash', 'email', 'created\_at', 'updated\_at'], 'required'],  
 [['status', 'created\_at', 'updated\_at', 'admin'], 'integer'],  
 [['username', 'password\_hash', 'password\_reset\_token', 'email', 'verification\_token'], 'string', 'max' => 255],  
 [['auth\_key'], 'string', 'max' => 32],  
 [['username'], 'unique'],  
 [['email'], 'unique'],  
 [['password\_reset\_token'], 'unique'],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function attributeLabels()  
 {  
 return [  
 'id' => 'ID',  
 'username' => 'Username',  
 'auth\_key' => 'Auth Key',  
 'password\_hash' => 'Password Hash',  
 'password\_reset\_token' => 'Password Reset Token',  
 'email' => 'Email',  
 'status' => 'Status',  
 'created\_at' => 'Created At',  
 'updated\_at' => 'Updated At',  
 'verification\_token' => 'Verification Token',  
 'admin' => 'Admin',  
 ];  
 }  
 public static function isUserAdmin($username)  
 {  
 if (static::*findOne*(['username' => $username, 'admin' => '1'])) {  
  
 return true;  
  
 } else {  
  
 return false;  
 }  
  
 }  
}

<?php  
  
namespace app\models;  
  
use yii\base\Model;  
use yii\data\ActiveDataProvider;  
use app\models\User;  
  
*/\*\*  
 \* UserSearch represents the model behind the search form of `app\models\User`.  
 \*/*class UserSearch extends User  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 [['id', 'status', 'created\_at', 'updated\_at', 'admin'], 'integer'],  
 [['username', 'auth\_key', 'password\_hash', 'password\_reset\_token', 'email', 'verification\_token'], 'safe'],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function scenarios()  
 {  
 // bypass scenarios() implementation in the parent class  
 return Model::*scenarios*();  
 }  
  
 */\*\*  
 \* Creates data provider instance with search query applied  
 \*  
 \** ***@param*** *array $params  
 \*  
 \** ***@return*** *ActiveDataProvider  
 \*/* public function search($params)  
 {  
 $query = User::*find*();  
  
 // add conditions that should always apply here  
  
 $dataProvider = new ActiveDataProvider([  
 'query' => $query,  
 ]);  
  
 $this->load($params);  
  
 if (!$this->validate()) {  
 // uncomment the following line if you do not want to return any records when validation fails  
 // $query->where('0=1');  
 return $dataProvider;  
 }  
  
 // grid filtering conditions  
 $query->andFilterWhere([  
 'id' => $this->id,  
 'status' => $this->status,  
 'created\_at' => $this->created\_at,  
 'updated\_at' => $this->updated\_at,  
 'admin' => $this->admin,  
 ]);  
  
 $query->andFilterWhere(['like', 'username', $this->username])  
 ->andFilterWhere(['like', 'auth\_key', $this->auth\_key])  
 ->andFilterWhere(['like', 'password\_hash', $this->password\_hash])  
 ->andFilterWhere(['like', 'password\_reset\_token', $this->password\_reset\_token])  
 ->andFilterWhere(['like', 'email', $this->email])  
 ->andFilterWhere(['like', 'verification\_token', $this->verification\_token]);  
  
 return $dataProvider;  
 }  
}

<?php  
  
namespace frontend\models;  
  
use common\models\User;  
use yii\base\InvalidArgumentException;  
use yii\base\Model;  
  
class VerifyEmailForm extends Model  
{  
 */\*\*  
 \** ***@var*** *string  
 \*/* public $token;  
  
 */\*\*  
 \** ***@var*** *User  
 \*/* private $\_user;  
  
  
 */\*\*  
 \* Creates a form model with given token.  
 \*  
 \** ***@param*** *string $token  
 \** ***@param*** *array $config name-value pairs that will be used to initialize the object properties  
 \** ***@throws*** *InvalidArgumentException if token is empty or not valid  
 \*/* public function \_\_construct($token, array $config = [])  
 {  
 if (empty($token) || !is\_string($token)) {  
 throw new InvalidArgumentException('Verify email token cannot be blank.');  
 }  
 $this->\_user = User::*findByVerificationToken*($token);  
 if (!$this->\_user) {  
 throw new InvalidArgumentException('Wrong verify email token.');  
 }  
 parent::*\_\_construct*($config);  
 }  
  
 */\*\*  
 \* Verify email  
 \*  
 \** ***@return*** *User|null the saved model or null if saving fails  
 \*/* public function verifyEmail()  
 {  
 $user = $this->\_user;  
 $user->status = User::*STATUS\_ACTIVE*;  
 return $user->save(false) ? $user : null;  
 }  
}

<?php  
  
namespace frontend\controllers;  
  
use Yii;  
use frontend\models\Products;  
use app\models\ProductsSearch;  
use yii\web\Controller;  
use app\models\User;  
use yii\web\NotFoundHttpException;  
use yii\web\UploadedFile;  
use yii\filters\VerbFilter;  
Use yii\filters\AccessControl;  
  
*/\*\*  
 \* ProductsController implements the CRUD actions for Products model.  
 \*/*class ProductsController extends Controller  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function behaviors()  
 {  
 return [  
 'access' => [  
 'class' => AccessControl::*className*(),  
 // 'only' => ['logout', 'signup', 'about'],  
 'rules' => [  
 [  
 'allow' => true,  
 'roles' => ['@'],  
 'matchCallback' => function ($rule, $action) {  
 return User::*isUserAdmin*(Yii::*$app*->user->identity->username);  
 },  
 ],  
 ],  
 ],  
 'verbs' => [  
 'class' => VerbFilter::*className*(),  
 'actions' => [  
 'delete' => ['POST'],  
 ],  
 ],  
 ];  
 return [  
 'verbs' => [  
 'class' => VerbFilter::*className*(),  
 'actions' => [  
 'delete' => ['POST'],  
 ],  
 ],  
 ];  
 }  
  
 */\*\*  
 \* Lists all Products models.  
 \** ***@return*** *mixed  
 \*/* public function actionIndex()  
 {  
 $searchModel = new ProductsSearch();  
 $dataProvider = $searchModel->search(Yii::*$app*->request->queryParams);  
  
 return $this->render('index', [  
 'searchModel' => $searchModel,  
 'dataProvider' => $dataProvider,  
 ]);  
 }  
  
 */\*\*  
 \* Displays a single Products model.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionView($id)  
 {  
 return $this->render('view', [  
 'model' => $this->findModel($id),  
 ]);  
 }  
  
 */\*\*  
 \* Creates a new Products model.  
 \* If creation is successful, the browser will be redirected to the 'view' page.  
 \** ***@return*** *mixed  
 \*/* public function actionCreate()  
 {  
 $model = new Products();  
  
 if ($model->load(Yii::*$app*->request->post())) {  
 $imageFile = UploadedFile::*getInstance*($model,'image');  
 if(isset($imageFile->size)){  
 $imageFile->saveAs('admin/uploads/'.$imageFile->baseName.'.'.$imageFile->extension);  
 }  
 $model->user\_email = Yii::*$app*->user->identity->email;  
 $model->image = $imageFile->baseName.'.'.$imageFile->extension;  
 $model->save(false);  
 return $this->redirect(['view', 'id' => $model->id]);  
 }  
  
 return $this->render('create', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Updates an existing Products model.  
 \* If update is successful, the browser will be redirected to the 'view' page.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionUpdate($id)  
 {  
 $model = $this->findModel($id);  
  
 if ($model->load(Yii::*$app*->request->post())) {  
  
 $model->user\_email = Yii::*$app*->user->identity->email;  
 $model->save(false);  
 return $this->redirect(['view', 'id' => $model->id]);  
 }  
  
 return $this->render('update', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Deletes an existing Products model.  
 \* If deletion is successful, the browser will be redirected to the 'index' page.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionDelete($id)  
 {  
 $this->findModel($id)->delete();  
  
 return $this->redirect(['index']);  
 }  
  
 */\*\*  
 \* Finds the Products model based on its primary key value.  
 \* If the model is not found, a 404 HTTP exception will be thrown.  
 \** ***@param*** *integer $id  
 \** ***@return*** *Products the loaded model  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* protected function findModel($id)  
 {  
 if (($model = Products::*findOne*($id)) !== null) {  
 return $model;  
 }  
  
 throw new NotFoundHttpException('The requested page does not exist.');  
 }  
}

<?php  
namespace frontend\controllers;  
  
// use backend\models\Products;  
use app\models\ProductsSearch;  
use common\models\LoginForm;  
use common\models\User;  
use frontend\models\ContactForm;  
use frontend\models\PasswordResetRequestForm;  
use frontend\models\Products;  
use frontend\models\ResendVerificationEmailForm;  
use frontend\models\ResetPasswordForm;  
use frontend\models\SignupForm;  
use frontend\models\VerifyEmailForm;  
use Yii;  
use yii\base\InvalidArgumentException;  
use yii\data\ActiveDataProvider;  
use yii\filters\AccessControl;  
use yii\filters\VerbFilter;  
use yii\web\BadRequestHttpException;  
use yii\web\Controller;  
use yii\helpers\ArrayHelper;  
use yii\web\NotFoundHttpException;  
  
*/\*\*  
 \* Site controller  
 \*/*class SiteController extends Controller  
{  
 public $enableCsrfValidation = false;  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function behaviors()  
 {  
 return [  
 'access' => [  
 'class' => AccessControl::*className*(),  
 'only' => ['logout', 'signup'],  
 'rules' => [  
 [  
 'actions' => ['signup'],  
 'allow' => true,  
 'roles' => ['?'],  
 ],  
 [  
 'actions' => ['logout'],  
 'allow' => true,  
 'roles' => ['@'],  
 ],  
 ],  
 ],  
 'verbs' => [  
 'class' => VerbFilter::*className*(),  
 'actions' => [  
 'logout' => ['post'],  
 ],  
 ],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function actions()  
 {  
 return [  
 'error' => [  
 'class' => 'yii\web\ErrorAction',  
 ],  
 'captcha' => [  
 'class' => 'yii\captcha\CaptchaAction',  
 'fixedVerifyCode' => *YII\_ENV\_TEST* ? 'testme' : null,  
 ],  
 ];  
 }  
  
 */\*\*  
 \* Displays homepage.  
 \*  
 \** ***@return*** *mixed  
 \*/* public function actionIndex()  
 {  
  
 $products = new ActiveDataProvider(['query' => Products::*find*()->where(['avalibility' => 1])->orderBy('created\_at DESC'),  
 'pagination' => ['pageSize' => 10],  
 ]);  
 return $this->render('index', ['products' => $products]);  
  
 }  
  
 public function actionCart()  
 {  
  
 $session = Yii::*$app*->session;  
  
 $id = Yii::*$app*->getRequest()->getQueryParam('id');  
  
 // Preces noņemšana no groza  
 if (isset($id)) {  
 $prodArray = $session['cart'];  
 unset($prodArray[$id]);  
 $cart = $prodArray;  
 $session['cart']= $cart;  
  
 } else {  
 $cart = $session['cart'];  
 }  
 $check\_pid\_post = Yii::*$app*->request->post('pid');  
 if (isset($check\_pid\_post)) {  
 $cart[] = $check\_pid\_post; //product to cart  
 $session['cart'] = $cart;  
 }  
  
 return $this->render('cart', [  
 // 'carts' => $carts,  
 'test' => $cart,  
 ]);  
  
 }  
  
 public function actionAddToCart($id)  
 {  
 $cart = Yii::*$app*->cart;  
  
 $model = ShopItems::*findOne*($id);  
 if ($model) {  
 $cart->put($model, 1);  
 return $this->redirect(['cart-view']);  
 }  
 throw new NotFoundHttpException();  
 }  
  
 */\*\*  
 \* Logs in a user.  
 \*  
 \** ***@return*** *mixed  
 \*/* public function actionLogin()  
 {  
 if (!Yii::*$app*->user->isGuest) {  
 return $this->goHome();  
 }  
  
 $model = new LoginForm();  
 if ($model->load(Yii::*$app*->request->post()) && $model->login()) {  
 return $this->goBack();  
 } else {  
 $model->password = '';  
  
 return $this->render('login', [  
 'model' => $model,  
 ]);  
 }  
 }  
  
 */\*\*  
 \* Logs out the current user.  
 \*  
 \** ***@return*** *mixed  
 \*/* public function actionLogout()  
 {  
 Yii::*$app*->user->logout();  
  
 return $this->goHome();  
 }  
  
 */\*\*  
 \* Displays contact page.  
 \*  
 \** ***@return*** *mixed  
 \*/* public function actionContact()  
 {  
 $model = new ContactForm();  
 if ($model->load(Yii::*$app*->request->post()) && $model->validate()) {  
 if ($model->sendEmail(Yii::*$app*->params['adminEmail'])) {  
 Yii::*$app*->session->setFlash('success', 'Thank you for contacting us. We will respond to you as soon as possible.');  
 } else {  
 Yii::*$app*->session->setFlash('error', 'There was an error sending your message.');  
 }  
  
 return $this->refresh();  
 } else {  
 return $this->render('contact', [  
 'model' => $model,  
 ]);  
 }  
 }  
  
 */\*\*  
 \* Displays about page.  
 \*  
 \** ***@return*** *mixed  
 \*/* public function actionAbout()  
 {  
 return $this->render('about');  
 }  
  
 */\*\*  
 \* Signs user up.  
 \*  
 \** ***@return*** *mixed  
 \*/* public function actionSignup()  
 {  
 $model = new SignupForm();  
 if ($model->load(Yii::*$app*->request->post()) && $model->signup()) {  
 Yii::*$app*->session->setFlash('success', 'Thank you for registration. Please check your inbox for verification email.');  
 return $this->goHome();  
 }  
  
 return $this->render('signup', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Requests password reset.  
 \*  
 \** ***@return*** *mixed  
 \*/* public function actionRequestPasswordReset()  
 {  
 $model = new PasswordResetRequestForm();  
 if ($model->load(Yii::*$app*->request->post()) && $model->validate()) {  
 if ($model->sendEmail()) {  
 Yii::*$app*->session->setFlash('success', 'Check your email for further instructions.');  
  
 return $this->goHome();  
 } else {  
 Yii::*$app*->session->setFlash('error', 'Sorry, we are unable to reset password for the provided email address.');  
 }  
 }  
  
 return $this->render('requestPasswordResetToken', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Resets password.  
 \*  
 \** ***@param*** *string $token  
 \** ***@return*** *mixed  
 \** ***@throws*** *BadRequestHttpException  
 \*/* public function actionResetPassword($token)  
 {  
 try {  
 $model = new ResetPasswordForm($token);  
 } catch (InvalidArgumentException $e) {  
 throw new BadRequestHttpException($e->getMessage());  
 }  
  
 if ($model->load(Yii::*$app*->request->post()) && $model->validate() && $model->resetPassword()) {  
 Yii::*$app*->session->setFlash('success', 'New password saved.');  
  
 return $this->goHome();  
 }  
  
 return $this->render('resetPassword', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Verify email address  
 \*  
 \** ***@param*** *string $token  
 \** ***@throws*** *BadRequestHttpException  
 \** ***@return*** *yii\web\Response  
 \*/* public function actionVerifyEmail($token)  
 {  
 try {  
 $model = new VerifyEmailForm($token);  
 } catch (InvalidArgumentException $e) {  
 throw new BadRequestHttpException($e->getMessage());  
 }  
 if ($user = $model->verifyEmail()) {  
 if (Yii::*$app*->user->login($user)) {  
 Yii::*$app*->session->setFlash('success', 'Your email has been confirmed!');  
 return $this->goHome();  
 }  
 }  
  
 Yii::*$app*->session->setFlash('error', 'Sorry, we are unable to verify your account with provided token.');  
 return $this->goHome();  
 }  
  
 */\*\*  
 \* Resend verification email  
 \*  
 \** ***@return*** *mixed  
 \*/* public function actionResendVerificationEmail()  
 {  
 $model = new ResendVerificationEmailForm();  
 if ($model->load(Yii::*$app*->request->post()) && $model->validate()) {  
 if ($model->sendEmail()) {  
 Yii::*$app*->session->setFlash('success', 'Check your email for further instructions.');  
 return $this->goHome();  
 }  
 Yii::*$app*->session->setFlash('error', 'Sorry, we are unable to resend verification email for the provided email address.');  
 }  
  
 return $this->render('resendVerificationEmail', [  
 'model' => $model,  
 ]);  
 }  
  
 protected function findModel($id)  
 {  
 $model = $this->findModel($id);  
  
 if (($model = Products::*findOne*($id)) !== null) {  
 return $model;  
 }  
  
 throw new NotFoundHttpException('The requested page does not exist.');  
 }  
}

<?php  
  
namespace frontend\controllers;  
  
use app\models\User;  
// use common\models\User;  
use app\models\UserSearch;  
use Yii;  
Use yii\filters\AccessControl;  
use yii\filters\VerbFilter;  
use yii\web\Controller;  
use yii\web\NotFoundHttpException;  
  
*/\*\*  
 \* UserController implements the CRUD actions for User model.  
 \*/*class UserController extends Controller  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function behaviors()  
 {  
 return [  
 'access' => [  
 'class' => AccessControl::*className*(),  
 // 'only' => ['logout', 'signup', 'about'],  
 'rules' => [  
 [  
 'allow' => true,  
 'roles' => ['@'],  
 'matchCallback' => function ($rule, $action) {  
 return User::*isUserAdmin*(Yii::*$app*->user->identity->username);  
 },  
 ],  
 ],  
 ],  
 'verbs' => [  
 'class' => VerbFilter::*className*(),  
 'actions' => [  
 'delete' => ['POST'],  
 ],  
 ],  
 ];  
 }  
  
 */\*\*  
 \* Lists all User models.  
 \** ***@return*** *mixed  
 \*/* public function actionIndex()  
 {  
 $searchModel = new UserSearch();  
 $dataProvider = $searchModel->search(Yii::*$app*->request->queryParams);  
  
 return $this->render('index', [  
 'searchModel' => $searchModel,  
 'dataProvider' => $dataProvider,  
 ]);  
 }  
  
 */\*\*  
 \* Displays a single User model.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionView($id)  
 {  
 return $this->render('view', [  
 'model' => $this->findModel($id),  
 ]);  
 }  
  
 */\*\*  
 \* Creates a new User model.  
 \* If creation is successful, the browser will be redirected to the 'view' page.  
 \** ***@return*** *mixed  
 \*/* public function actionCreate()  
 {  
 $model = new User();  
  
 if ($model->load(Yii::*$app*->request->post()) && $model->save()) {  
 return $this->redirect(['view', 'id' => $model->id]);  
 }  
  
 return $this->render('create', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Updates an existing User model.  
 \* If update is successful, the browser will be redirected to the 'view' page.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionUpdate($id)  
 {  
 $model = $this->findModel($id);  
  
 if ($model->load(Yii::*$app*->request->post()) && $model->save()) {  
 return $this->redirect(['view', 'id' => $model->id]);  
 }  
  
 return $this->render('update', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Deletes an existing User model.  
 \* If deletion is successful, the browser will be redirected to the 'index' page.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionDelete($id)  
 {  
 $this->findModel($id)->delete();  
  
 return $this->redirect(['index']);  
 }  
  
 */\*\*  
 \* Finds the User model based on its primary key value.  
 \* If the model is not found, a 404 HTTP exception will be thrown.  
 \** ***@param*** *integer $id  
 \** ***@return*** *User the loaded model  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* protected function findModel($id)  
 {  
 if (($model = User::*findOne*($id)) !== null) {  
 return $model;  
 }  
  
 throw new NotFoundHttpException('The requested page does not exist.');  
 }  
}

<?php  
$params = array\_merge(  
 require *\_\_DIR\_\_* . '/../../common/config/params.php',  
 require *\_\_DIR\_\_* . '/../../common/config/params-local.php',  
 require *\_\_DIR\_\_* . '/params.php',  
 require *\_\_DIR\_\_* . '/params-local.php'  
);  
  
return [  
 'id' => 'app-frontend',  
 'basePath' => dirname(*\_\_DIR\_\_*),  
 'bootstrap' => ['log'],  
 'controllerNamespace' => 'frontend\controllers',  
 'components' => [  
 'request' => [  
 'csrfParam' => '\_csrf-frontend',  
 ],  
 'user' => [  
 'identityClass' => 'common\models\User',  
 'enableAutoLogin' => true,  
 'identityCookie' => ['name' => '\_identity-frontend', 'httpOnly' => true],  
 ],  
 'session' => [  
 // this is the name of the session cookie used for login on the frontend  
 'name' => 'advanced-frontend',  
 ],  
 'log' => [  
 'traceLevel' => *YII\_DEBUG* ? 3 : 0,  
 'targets' => [  
 [  
 'class' => 'yii\log\FileTarget',  
 'levels' => ['error', 'warning'],  
 ],  
 ],  
 ],  
 'errorHandler' => [  
 'errorAction' => 'site/error',  
 ],  
 'cart' => [  
 'class' => 'yz\shoppingcart\ShoppingCart',  
 'cartId' => 'my\_application\_cart',  
 ]  
 /\*  
 'urlManager' => [  
 'enablePrettyUrl' => true,  
 'showScriptName' => false,  
 'rules' => [  
 ],  
 ],  
 \*/  
 ],  
 'params' => $params,  
];

<?php  
  
namespace frontend\assets;  
  
use yii\web\AssetBundle;  
  
*/\*\*  
 \* Main frontend application asset bundle.  
 \*/*class AppAsset extends AssetBundle  
{  
 public $basePath = '@webroot';  
 public $baseUrl = '@web';  
 public $css = [  
 'css/site.css',  
 ];  
 public $js = [  
 ];  
 public $depends = [  
 'yii\web\YiiAsset',  
 'yii\bootstrap\BootstrapAsset',  
 ];  
}

<?php  
use yii\helpers\Html;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$user common\models\User \*/*$verifyLink = Yii::*$app*->urlManager->createAbsoluteUrl(['site/verify-email', 'token' => $user->verification\_token]);  
?>  
<div class="verify-email">  
 <p>Hello <?= Html::*encode*($user->username) ?>,</p>  
  
 <p>Follow the link below to verify your email:</p>  
  
 <p><?= Html::*a*(Html::*encode*($verifyLink), $verifyLink) ?></p>  
</div>

<?php  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$user common\models\User \*/*$verifyLink = Yii::*$app*->urlManager->createAbsoluteUrl(['site/verify-email', 'token' => $user->verification\_token]);  
?>  
Hello <?= $user->username ?>,  
  
Follow the link below to verify your email:  
  
<?= $verifyLink ?>

<?php  
use yii\helpers\Html;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$user common\models\User \*/*$resetLink = Yii::*$app*->urlManager->createAbsoluteUrl(['site/reset-password', 'token' => $user->password\_reset\_token]);  
?>  
<div class="password-reset">  
 <p>Hello <?= Html::*encode*($user->username) ?>,</p>  
  
 <p>Follow the link below to reset your password:</p>  
  
 <p><?= Html::*a*(Html::*encode*($resetLink), $resetLink) ?></p>  
</div>

<?php  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$user common\models\User \*/*$resetLink = Yii::*$app*->urlManager->createAbsoluteUrl(['site/reset-password', 'token' => $user->password\_reset\_token]);  
?>  
Hello <?= $user->username ?>,  
  
Follow the link below to reset your password:  
  
<?= $resetLink ?>

<?php  
namespace common\models;  
  
use Yii;  
use yii\base\NotSupportedException;  
use yii\behaviors\TimestampBehavior;  
use yii\db\ActiveRecord;  
use yii\web\IdentityInterface;  
  
*/\*\*  
 \* User model  
 \*  
 \** ***@property*** *integer $id  
 \** ***@property*** *string $username  
 \** ***@property*** *string $password\_hash  
 \** ***@property*** *string $password\_reset\_token  
 \** ***@property*** *string $verification\_token  
 \** ***@property*** *string $email  
 \** ***@property*** *string $auth\_key  
 \** ***@property*** *integer $status  
 \** ***@property*** *integer $created\_at  
 \** ***@property*** *integer $updated\_at  
 \** ***@property*** *string $password write-only password  
 \*/*class Admin extends ActiveRecord implements IdentityInterface  
{  
 const *STATUS\_DELETED* = 0;  
 const *STATUS\_INACTIVE* = 9;  
 const *STATUS\_ACTIVE* = 10;  
  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function tableName()  
 {  
 return '{{%admin}}';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function behaviors()  
 {  
 return [  
 TimestampBehavior::*className*(),  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 ['status', 'default', 'value' => self::*STATUS\_INACTIVE*],  
 ['status', 'in', 'range' => [self::*STATUS\_ACTIVE*, self::*STATUS\_INACTIVE*, self::*STATUS\_DELETED*]],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function findIdentity($id)  
 {  
 return static::*findOne*(['id' => $id, 'status' => self::*STATUS\_ACTIVE*]);  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function findIdentityByAccessToken($token, $type = null)  
 {  
 throw new NotSupportedException('"findIdentityByAccessToken" is not implemented.');  
 }  
  
 */\*\*  
 \* Finds user by username  
 \*  
 \** ***@param*** *string $username  
 \** ***@return*** *static|null  
 \*/* public static function findByUsername($username)  
 {  
 return static::*findOne*(['username' => $username, 'status' => self::*STATUS\_ACTIVE*]);  
 }  
  
 */\*\*  
 \* Finds user by password reset token  
 \*  
 \** ***@param*** *string $token password reset token  
 \** ***@return*** *static|null  
 \*/* public static function findByPasswordResetToken($token)  
 {  
 if (!static::*isPasswordResetTokenValid*($token)) {  
 return null;  
 }  
  
 return static::*findOne*([  
 'password\_reset\_token' => $token,  
 'status' => self::*STATUS\_ACTIVE*,  
 ]);  
 }  
  
 */\*\*  
 \* Finds user by verification email token  
 \*  
 \** ***@param*** *string $token verify email token  
 \** ***@return*** *static|null  
 \*/* public static function findByVerificationToken($token) {  
 return static::*findOne*([  
 'verification\_token' => $token,  
 'status' => self::*STATUS\_INACTIVE* ]);  
 }  
  
 */\*\*  
 \* Finds out if password reset token is valid  
 \*  
 \** ***@param*** *string $token password reset token  
 \** ***@return*** *bool  
 \*/* public static function isPasswordResetTokenValid($token)  
 {  
 if (empty($token)) {  
 return false;  
 }  
  
 $timestamp = (int) substr($token, strrpos($token, '\_') + 1);  
 $expire = Yii::*$app*->params['user.passwordResetTokenExpire'];  
 return $timestamp + $expire >= time();  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function getId()  
 {  
 return $this->getPrimaryKey();  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function getAuthKey()  
 {  
 return $this->auth\_key;  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function validateAuthKey($authKey)  
 {  
 return $this->getAuthKey() === $authKey;  
 }  
  
 */\*\*  
 \* Validates password  
 \*  
 \** ***@param*** *string $password password to validate  
 \** ***@return*** *bool if password provided is valid for current user  
 \*/* public function validatePassword($password)  
 {  
 return Yii::*$app*->security->validatePassword($password, $this->password\_hash);  
 }  
  
 */\*\*  
 \* Generates password hash from password and sets it to the model  
 \*  
 \** ***@param*** *string $password  
 \*/* public function setPassword($password)  
 {  
 $this->password\_hash = Yii::*$app*->security->generatePasswordHash($password);  
 }  
  
 */\*\*  
 \* Generates "remember me" authentication key  
 \*/* public function generateAuthKey()  
 {  
 $this->auth\_key = Yii::*$app*->security->generateRandomString();  
 }  
  
 */\*\*  
 \* Generates new password reset token  
 \*/* public function generatePasswordResetToken()  
 {  
 $this->password\_reset\_token = Yii::*$app*->security->generateRandomString() . '\_' . time();  
 }  
  
 */\*\*  
 \* Generates new token for email verification  
 \*/* public function generateEmailVerificationToken()  
 {  
 $this->verification\_token = Yii::*$app*->security->generateRandomString() . '\_' . time();  
 }  
  
 */\*\*  
 \* Removes password reset token  
 \*/* public function removePasswordResetToken()  
 {  
 $this->password\_reset\_token = null;  
 }  
}

<?php  
namespace common\models;  
  
use Yii;  
use yii\base\Model;  
  
  
*/\*\*  
 \* Login form  
 \*/*class AdminForm extends Model  
{  
 public $username;  
 public $password;  
 public $rememberMe = true;  
  
 private $\_user;  
  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 // username and password are both required  
 [['username', 'password'], 'required'],  
 // rememberMe must be a boolean value  
 ['rememberMe', 'boolean'],  
 // password is validated by validatePassword()  
 ['password', 'validatePassword'],  
 ];  
 }  
  
 */\*\*  
 \* Validates the password.  
 \* This method serves as the inline validation for password.  
 \*  
 \** ***@param*** *string $attribute the attribute currently being validated  
 \** ***@param*** *array $params the additional name-value pairs given in the rule  
 \*/* public function validatePassword($attribute, $params)  
 {  
 if (!$this->hasErrors()) {  
 $user = $this->getUser();  
 if (!$user || !$user->validatePassword($this->password)) {  
 $this->addError($attribute, 'Incorrect username or password.');  
 }  
 }  
 }  
  
 */\*\*  
 \* Logs in a user using the provided username and password.  
 \*  
 \** ***@return*** *bool whether the user is logged in successfully  
 \*/* public function login()  
 {  
 if ($this->validate()) {  
 return Yii::*$app*->user->login($this->getUser(), $this->rememberMe ? 3600 \* 24 \* 30 : 0);  
 }  
   
 return false;  
 }  
  
 */\*\*  
 \* Finds user by [[username]]  
 \*  
 \** ***@return*** *User|null  
 \*/* protected function getUser()  
 {  
 if ($this->\_user === null) {  
 $this->\_user = Admin::*findByUsername*($this->username);  
 }  
  
 return $this->\_user;  
 }  
}

<?php  
namespace common\models;  
  
use Yii;  
use yii\base\Model;  
  
*/\*\*  
 \* Login form  
 \*/*class LoginForm extends Model  
{  
 public $username;  
 public $password;  
 public $rememberMe = true;  
  
 private $\_user;  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 // username and password are both required  
 [['username', 'password'], 'required'],  
 // rememberMe must be a boolean value  
 ['rememberMe', 'boolean'],  
 // password is validated by validatePassword()  
 ['password', 'validatePassword'],  
 ];  
 }  
  
 */\*\*  
 \* Validates the password.  
 \* This method serves as the inline validation for password.  
 \*  
 \** ***@param*** *string $attribute the attribute currently being validated  
 \** ***@param*** *array $params the additional name-value pairs given in the rule  
 \*/* public function validatePassword($attribute, $params)  
 {  
 if (!$this->hasErrors()) {  
 $user = $this->getUser();  
 if (!$user || !$user->validatePassword($this->password)) {  
 $this->addError($attribute, 'Incorrect username or password.');  
 }  
 }  
 }  
  
 */\*\*  
 \* Logs in a user using the provided username and password.  
 \*  
 \** ***@return*** *bool whether the user is logged in successfully  
 \*/* public function login()  
 {  
 if ($this->validate()) {  
 return Yii::*$app*->user->login($this->getUser(), $this->rememberMe ? 3600 \* 24 \* 30 : 0);  
 }  
  
 return false;  
 }  
  
 */\*\*  
 \* Finds user by [[username]]  
 \*  
 \** ***@return*** *User|null  
 \*/* protected function getUser()  
 {  
 if ($this->\_user === null) {  
 $this->\_user = User::*findByUsername*($this->username);  
 }  
  
 return $this->\_user;  
 }  
 public function loginAdmin()  
 {  
 if ($this->validate() && User::*isUserAdmin*($this->username)) {  
 return Yii::*$app*->user->login($this->getUser(), $this->rememberMe ? 3600 \* 24 \* 30 : 0);  
 } else {  
 return false;  
 }  
 }  
}

<?php  
namespace common\models;  
  
use Yii;  
use yii\base\NotSupportedException;  
use yii\behaviors\TimestampBehavior;  
use yii\db\ActiveRecord;  
use yii\web\IdentityInterface;  
  
*/\*\*  
 \* User model  
 \*  
 \** ***@property*** *integer $id  
 \** ***@property*** *string $username  
 \** ***@property*** *string $password\_hash  
 \** ***@property*** *string $password\_reset\_token  
 \** ***@property*** *string $verification\_token  
 \** ***@property*** *string $email  
 \** ***@property*** *string $auth\_key  
 \** ***@property*** *integer $status  
 \** ***@property*** *integer $created\_at  
 \** ***@property*** *integer $updated\_at  
 \** ***@property*** *string $password write-only password  
 \*/*class User extends ActiveRecord implements IdentityInterface  
{  
 const *STATUS\_DELETED* = 0;  
 const *STATUS\_INACTIVE* = 9;  
 const *STATUS\_ACTIVE* = 10;  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function tableName()  
 {  
 return '{{%user}}';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function behaviors()  
 {  
 return [  
 TimestampBehavior::*className*(),  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 ['status', 'default', 'value' => self::*STATUS\_INACTIVE*],  
 ['status', 'in', 'range' => [self::*STATUS\_ACTIVE*, self::*STATUS\_INACTIVE*, self::*STATUS\_DELETED*]],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function findIdentity($id)  
 {  
 return static::*findOne*(['id' => $id, 'status' => self::*STATUS\_ACTIVE*]);  
 }  
  
 public static function isUserAdmin($username)  
 {  
 if (static::*findOne*(['username' => $username, 'admin' => '1'])) {  
  
 return true;  
  
 } else {  
  
 return false;  
 }  
  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function findIdentityByAccessToken($token, $type = null)  
 {  
 throw new NotSupportedException('"findIdentityByAccessToken" is not implemented.');  
 }  
  
 */\*\*  
 \* Finds user by username  
 \*  
 \** ***@param*** *string $username  
 \** ***@return*** *static|null  
 \*/* public static function findByUsername($username)  
 {  
 return static::*findOne*(['username' => $username, 'status' => self::*STATUS\_ACTIVE*]);  
 }  
  
 */\*\*  
 \* Finds user by password reset token  
 \*  
 \** ***@param*** *string $token password reset token  
 \** ***@return*** *static|null  
 \*/* public static function findByPasswordResetToken($token)  
 {  
 if (!static::*isPasswordResetTokenValid*($token)) {  
 return null;  
 }  
  
 return static::*findOne*([  
 'password\_reset\_token' => $token,  
 'status' => self::*STATUS\_ACTIVE*,  
 ]);  
 }  
  
 */\*\*  
 \* Finds user by verification email token  
 \*  
 \** ***@param*** *string $token verify email token  
 \** ***@return*** *static|null  
 \*/* public static function findByVerificationToken($token)  
 {  
 return static::*findOne*([  
 'verification\_token' => $token,  
 'status' => self::*STATUS\_INACTIVE*,  
 ]);  
 }  
  
 */\*\*  
 \* Finds out if password reset token is valid  
 \*  
 \** ***@param*** *string $token password reset token  
 \** ***@return*** *bool  
 \*/* public static function isPasswordResetTokenValid($token)  
 {  
 if (empty($token)) {  
 return false;  
 }  
  
 $timestamp = (int) substr($token, strrpos($token, '\_') + 1);  
 $expire = Yii::*$app*->params['user.passwordResetTokenExpire'];  
 return $timestamp + $expire >= time();  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function getId()  
 {  
 return $this->getPrimaryKey();  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function getAuthKey()  
 {  
 return $this->auth\_key;  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function validateAuthKey($authKey)  
 {  
 return $this->getAuthKey() === $authKey;  
 }  
  
 */\*\*  
 \* Validates password  
 \*  
 \** ***@param*** *string $password password to validate  
 \** ***@return*** *bool if password provided is valid for current user  
 \*/* public function validatePassword($password)  
 {  
 return Yii::*$app*->security->validatePassword($password, $this->password\_hash);  
 }  
  
 */\*\*  
 \* Generates password hash from password and sets it to the model  
 \*  
 \** ***@param*** *string $password  
 \*/* public function setPassword($password)  
 {  
 $this->password\_hash = Yii::*$app*->security->generatePasswordHash($password);  
 }  
  
 */\*\*  
 \* Generates "remember me" authentication key  
 \*/* public function generateAuthKey()  
 {  
 $this->auth\_key = Yii::*$app*->security->generateRandomString();  
 }  
  
 */\*\*  
 \* Generates new password reset token  
 \*/* public function generatePasswordResetToken()  
 {  
 $this->password\_reset\_token = Yii::*$app*->security->generateRandomString() . '\_' . time();  
 }  
  
 */\*\*  
 \* Generates new token for email verification  
 \*/* public function generateEmailVerificationToken()  
 {  
 $this->verification\_token = Yii::*$app*->security->generateRandomString() . '\_' . time();  
 }  
  
 */\*\*  
 \* Removes password reset token  
 \*/* public function removePasswordResetToken()  
 {  
 $this->password\_reset\_token = null;  
 }  
}

<?php  
defined('YII\_DEBUG') or define('YII\_DEBUG', true);  
defined('YII\_ENV') or define('YII\_ENV', 'dev');  
  
require *\_\_DIR\_\_* . '/../vendor/autoload.php';  
require *\_\_DIR\_\_* . '/../vendor/yiisoft/yii2/Yii.php';  
require *\_\_DIR\_\_* . '/../common/config/bootstrap.php';  
require *\_\_DIR\_\_* . '/../backend/config/bootstrap.php';  
  
$config = yii\helpers\ArrayHelper::*merge*(  
 require *\_\_DIR\_\_* . '/../common/config/main.php',  
 require *\_\_DIR\_\_* . '/../common/config/main-local.php',  
 require *\_\_DIR\_\_* . '/../backend/config/main.php',  
 require *\_\_DIR\_\_* . '/../backend/config/main-local.php'  
);  
  
(new yii\web\Application($config))->run();

<?php  
defined('YII\_DEBUG') or define('YII\_DEBUG', true);  
defined('YII\_ENV') or define('YII\_ENV', 'dev');  
  
require *\_\_DIR\_\_* . '/vendor/autoload.php';  
require *\_\_DIR\_\_* . '/vendor/yiisoft/yii2/Yii.php';  
require *\_\_DIR\_\_* . '/common/config/bootstrap.php';  
require *\_\_DIR\_\_* . '/frontend/config/bootstrap.php';  
  
$config = yii\helpers\ArrayHelper::*merge*(  
 require *\_\_DIR\_\_* . '/common/config/main.php',  
 require *\_\_DIR\_\_* . '/common/config/main-local.php',  
 require *\_\_DIR\_\_* . '/frontend/config/main.php',  
 require *\_\_DIR\_\_* . '/frontend/config/main-local.php'  
);  
  
(new yii\web\Application($config))->run();

#!/usr/bin/env php  
<?php  
*/\*\*  
 \* Yii Application Initialization Tool  
 \*  
 \* In order to run in non-interactive mode:  
 \*  
 \* init --env=Development --overwrite=n  
 \*/*if (!extension\_loaded('openssl')) {  
 die('The OpenSSL PHP extension is required by Yii2.');  
}  
  
$params = getParams();  
$root = str\_replace('\\', '/', *\_\_DIR\_\_*);  
$envs = require "$root/environments/index.php";  
$envNames = array\_keys($envs);  
  
echo "Yii Application Initialization Tool v1.0\n\n";  
  
$envName = null;  
if (empty($params['env']) || $params['env'] === true) {  
 echo "Which environment do you want the application to be initialized in?\n\n";  
 foreach ($envNames as $i => $name) {  
 echo " [$i] $name\n";  
 }  
 echo "\n Your choice [0-" . (count($envs) - 1) . ', or "q" to quit] ';  
 $answer = trim(fgets(*STDIN*));  
  
 if (!ctype\_digit($answer) || !in\_array($answer, range(0, count($envs) - 1))) {  
 echo "\n Quit initialization.\n";  
 exit(0);  
 }  
  
 if (isset($envNames[$answer])) {  
 $envName = $envNames[$answer];  
 }  
} else {  
 $envName = $params['env'];  
}  
  
if (!in\_array($envName, $envNames, true)) {  
 $envsList = implode(', ', $envNames);  
 echo "\n $envName is not a valid environment. Try one of the following: $envsList. \n";  
 exit(2);  
}  
  
$env = $envs[$envName];  
  
if (empty($params['env'])) {  
 echo "\n Initialize the application under '{$envNames[$answer]}' environment? [yes|no] ";  
 $answer = trim(fgets(*STDIN*));  
 if (strncasecmp($answer, 'y', 1)) {  
 echo "\n Quit initialization.\n";  
 exit(0);  
 }  
}  
  
$rootPath = "$root/environments/{$env['path']}";  
if (!is\_dir($rootPath)) {  
 printError("$rootPath directory does not exist. Check path in $envName environment.");  
 exit(3);  
}  
  
echo "\n Start initialization ...\n\n";  
  
$files = getFileList($rootPath);  
if (isset($env['skipFiles'])) {  
 $skipFiles = $env['skipFiles'];  
 array\_walk($skipFiles, function(&$value) use($env, $root) { $value = "$root/$value"; });  
 $files = array\_diff($files, array\_intersect\_key($env['skipFiles'], array\_filter($skipFiles, 'file\_exists')));  
}  
$all = false;  
foreach ($files as $file) {  
 if (!copyFile($root, "environments/{$env['path']}/$file", $file, $all, $params)) {  
 break;  
 }  
}  
  
$callbacks = ['setCookieValidationKey', 'setWritable', 'setExecutable', 'createSymlink'];  
foreach ($callbacks as $callback) {  
 if (!empty($env[$callback])) {  
 $callback($root, $env[$callback]);  
 }  
}  
  
echo "\n ... initialization completed.\n\n";  
  
function getFileList($root, $basePath = '')  
{  
 $files = [];  
 $handle = opendir($root);  
 while (($path = readdir($handle)) !== false) {  
 if ($path === '.git' || $path === '.svn' || $path === '.' || $path === '..') {  
 continue;  
 }  
 $fullPath = "$root/$path";  
 $relativePath = $basePath === '' ? $path : "$basePath/$path";  
 if (is\_dir($fullPath)) {  
 $files = array\_merge($files, getFileList($fullPath, $relativePath));  
 } else {  
 $files[] = $relativePath;  
 }  
 }  
 closedir($handle);  
 return $files;  
}  
  
function copyFile($root, $source, $target, &$all, $params)  
{  
 if (!is\_file($root . '/' . $source)) {  
 echo " skip $target ($source not exist)\n";  
 return true;  
 }  
 if (is\_file($root . '/' . $target)) {  
 if (file\_get\_contents($root . '/' . $source) === file\_get\_contents($root . '/' . $target)) {  
 echo " unchanged $target\n";  
 return true;  
 }  
 if ($all) {  
 echo " overwrite $target\n";  
 } else {  
 echo " exist $target\n";  
 echo " ...overwrite? [Yes|No|All|Quit] ";  
  
  
 $answer = !empty($params['overwrite']) ? $params['overwrite'] : trim(fgets(*STDIN*));  
 if (!strncasecmp($answer, 'q', 1)) {  
 return false;  
 } else {  
 if (!strncasecmp($answer, 'y', 1)) {  
 echo " overwrite $target\n";  
 } else {  
 if (!strncasecmp($answer, 'a', 1)) {  
 echo " overwrite $target\n";  
 $all = true;  
 } else {  
 echo " skip $target\n";  
 return true;  
 }  
 }  
 }  
 }  
 file\_put\_contents($root . '/' . $target, file\_get\_contents($root . '/' . $source));  
 return true;  
 }  
 echo " generate $target\n";  
 @mkdir(dirname($root . '/' . $target), 0777, true);  
 file\_put\_contents($root . '/' . $target, file\_get\_contents($root . '/' . $source));  
 return true;  
}  
  
function getParams()  
{  
 $rawParams = [];  
 if (isset($\_SERVER['argv'])) {  
 $rawParams = $\_SERVER['argv'];  
 array\_shift($rawParams);  
 }  
  
 $params = [];  
 foreach ($rawParams as $param) {  
 if (preg\_match('/^--([\w-]\*\w)(=(.\*))?$/', $param, $matches)) {  
 $name = $matches[1];  
 $params[$name] = isset($matches[3]) ? $matches[3] : true;  
 } else {  
 $params[] = $param;  
 }  
 }  
 return $params;  
}  
  
function setWritable($root, $paths)  
{  
 foreach ($paths as $writable) {  
 if (is\_dir("$root/$writable")) {  
 if (@chmod("$root/$writable", 0777)) {  
 echo " chmod 0777 $writable\n";  
 } else {  
 printError("Operation chmod not permitted for directory $writable.");  
 }  
 } else {  
 printError("Directory $writable does not exist.");  
 }  
 }  
}  
  
function setExecutable($root, $paths)  
{  
 foreach ($paths as $executable) {  
 if (file\_exists("$root/$executable")) {  
 if (@chmod("$root/$executable", 0755)) {  
 echo " chmod 0755 $executable\n";  
 } else {  
 printError("Operation chmod not permitted for $executable.");  
 }  
 } else {  
 printError("$executable does not exist.");  
 }  
 }  
}  
  
function setCookieValidationKey($root, $paths)  
{  
 foreach ($paths as $file) {  
 echo " generate cookie validation key in $file\n";  
 $file = $root . '/' . $file;  
 $length = 32;  
 $bytes = openssl\_random\_pseudo\_bytes($length);  
 $key = strtr(substr(base64\_encode($bytes), 0, $length), '+/=', '\_-.');  
 $content = preg\_replace('/(("|\')cookieValidationKey("|\')\s\*=>\s\*)(""|\'\')/', "\\1'$key'", file\_get\_contents($file));  
 file\_put\_contents($file, $content);  
 }  
}  
  
function createSymlink($root, $links)  
{  
 foreach ($links as $link => $target) {  
 //first removing folders to avoid errors if the folder already exists  
 @rmdir($root . "/" . $link);  
 //next removing existing symlink in order to update the target  
 if (is\_link($root . "/" . $link)) {  
 @unlink($root . "/" . $link);  
 }  
 if (@symlink($root . "/" . $target, $root . "/" . $link)) {  
 echo " symlink $root/$target $root/$link\n";  
 } else {  
 printError("Cannot create symlink $root/$target $root/$link.");  
 }  
 }  
}  
  
*/\*\*  
 \* Prints error message.  
 \** ***@param*** *string $message message  
 \*/*function printError($message)  
{  
 echo "\n " . formatMessage("Error. $message", ['fg-red']) . " \n";  
}  
  
*/\*\*  
 \* Returns true if the stream supports colorization. ANSI colors are disabled if not supported by the stream.  
 \*  
 \* - windows without ansicon  
 \* - not tty consoles  
 \*  
 \** ***@return*** *boolean true if the stream supports ANSI colors, otherwise false.  
 \*/*function ansiColorsSupported()  
{  
 return *DIRECTORY\_SEPARATOR* === '\\'  
 ? getenv('ANSICON') !== false || getenv('ConEmuANSI') === 'ON'  
 : function\_exists('posix\_isatty') && @posix\_isatty(*STDOUT*);  
}  
  
*/\*\*  
 \* Get ANSI code of style.  
 \** ***@param*** *string $name style name  
 \** ***@return*** *integer ANSI code of style.  
 \*/*function getStyleCode($name)  
{  
 $styles = [  
 'bold' => 1,  
 'fg-black' => 30,  
 'fg-red' => 31,  
 'fg-green' => 32,  
 'fg-yellow' => 33,  
 'fg-blue' => 34,  
 'fg-magenta' => 35,  
 'fg-cyan' => 36,  
 'fg-white' => 37,  
 'bg-black' => 40,  
 'bg-red' => 41,  
 'bg-green' => 42,  
 'bg-yellow' => 43,  
 'bg-blue' => 44,  
 'bg-magenta' => 45,  
 'bg-cyan' => 46,  
 'bg-white' => 47,  
 ];  
 return $styles[$name];  
}  
  
*/\*\*  
 \* Formats message using styles if STDOUT supports it.  
 \** ***@param*** *string $message message  
 \** ***@param*** *string[] $styles styles  
 \** ***@return*** *string formatted message.  
 \*/*function formatMessage($message, $styles)  
{  
 if (empty($styles) || !ansiColorsSupported()) {  
 return $message;  
 }  
  
 return sprintf("\x1b[%sm", implode(';', array\_map('getStyleCode', $styles))) . $message . "\x1b[0m";  
}