NODE, EXPRESS, JSON API, CRUD

BACKEND – turi turėti tokią struktūrą:

(Aplankas “src“ ir jo failai bus – config.js, helper.js, server.js.

Šiame aplanke bus dar du aplankai.

Pirmas - “db“, o jo viduje bus duomenų bazės failas Lauryno.sql.

Antras - “routes“, o jo viduje bus – usersRoutes.js, item\_typesRoutes.js, ordersRoutes.js, shop\_itemsRoutes.js, user\_rolesRoutes.js.)

.env

.eslintrc.js

.gitignore

package-lock.json

package.json

rest.rest

FRONTEND – turi turėti tokią struktūrą:

(Aplankas “public“ ir jame failai – login.html, register.html, shop.html, add\_item.html,

orders.htm.)

(Aplankas “js“ – failai jame - login.js, register.js, shop.js, add\_item.js, orders.js.

Kuriame dar bus aplankas “modules“ kuriame bus failas – helper.js.)

(Aplankas “styles“, su failais reset.css ir style.css.)

.eslintrc.js

.gitignore

package-lock.json

package.json

1. Sukurti serverį naudojant express.js

2. Sukurti duomenų bazę naudojant mysql

Kuriant visą projektą, naudoti "git" su logiškais commits.

3. Sukurti penkias lenteles duomenų bazėje, pavadinimu - Lauryno, pagal šį kodą:

1 - CREATE TABLE `users`(

`user\_id` int(13) UNSIGNED NOT NULL AUTO\_INCREMENT,

`name` varchar(255) NOT NULL,

`email` varchar(255) NOT NULL,

`password` varchar(255) NOT NULL,

`role\_id` int(13) UNSIGNED NOT NULL,

PRIMARY KEY (`user\_id`)

) ENGINE=InnoDB;

2 - CREATE TABLE `item\_types`(

`item\_type\_id` int(13) UNSIGNED NOT NULL AUTO\_INCREMENT,

`name` varchar(255) NOT NULL,

PRIMARY KEY (`item\_type\_id`)

) ENGINE=InnoDB;

3 - CREATE TABLE `orders`(

`order\_id` int(13) UNSIGNED NOT NULL AUTO\_INCREMENT,

`user\_id` int(13) UNSIGNED NOT NULL,

`shop\_item\_id` int(13) UNSIGNED NOT NULL,

`quantity` varchar(255) NOT NULL,

`total\_price` varchar(255) NOT NULL,

`status` boolean NOT NULL DEFAULT '0',

PRIMARY KEY(`order\_id`)

) ENGINE=InnoDB;

4 - CREATE TABLE `shop\_items`(

`shop\_item\_id` int(13) UNSIGNED NOT NULL AUTO\_INCREMENT,

`name` varchar(255) NOT NULL,

`price` decimal(65) NOT NULL,

`description` text NOT NULL,

`image` varchar(255) NOT NULL,

`item\_type\_id` int(13) UNSIGNED NOT NULL,

PRIMARY KEY (`shop\_item\_id`)

) ENGINE=InnoDB;

5 - CREATE TABLE `user\_roles`(

`role\_id` int(13) UNSIGNED NOT NULL AUTO\_INCREMENT,

`name` varchar(255) NOT NULL,

PRIMARY KEY (`role\_id`)

) ENGINE=InnoDB;

4. Sukurti CRUD operacijas vartotojams.

4.1. POST /api/auth/register - registruoti vartotoją su "name", "email", "password", "role\_id".

4.2. POST /api/auth/login - prisijungti vartotoją naudojantis "email" ir "password".

5. Sukurti CRUD operacijas parduotuvės prekėms.

5.1. POST /api/shop\_items - sukurti parduotuvės prekę su "name", "price", "description", "image", "item\_type\_id".

5.2. GET /api/shop\_items - gauti visas parduotuvės prekes. ("shop\_items“)

5.3. GET /api/shop\_items/:id - gauti parduotuvės prekę pagal "shop\_item\_id".

5.4. DELETE /api/shop\_items/:id - ištrinti parduotuvės prekę, pagal "shop\_item\_id".

6. Įterpti, 4 ar daugiau, į "item\_types\_id": food, drink, electronic, clothes

7. Sukurti CRUD operacijas užsakymams.

7.1. POST /api/orders - sukurti užsakymą su "user\_id", "shop\_item\_id", "quantity", "total\_price", "status".

7.2. GET /api/orders - gauti visus padarytus užsakymus. Gauta informacija, turi būti su "user\_id", "shop\_item\_id" ir "price".

7.3. GET /api/orders/:id - gauti užsakymą pagal "order\_id".

7.4. GET /api/orders/user/:user\_id - gauti visus užsakymus, priklausančius vienam "user\_id". Gauta informacija, turi būti su "user\_id", "name", "shop\_item\_id" ir "price".

8. Sukurti GET maršrutą /api/user\_roles - gauti visas "user\_roles".

8.1 Rankiniu būdu (phpmyadmin) įterpti 3 "user\_roles" į lentele: "admin", "user", "guest".

9. Sukurti užklausą middleware POST, PUT ir PATCH maršrutams patikrinimą. Pasitikrinti ar užklausoje yra visi reikalingi laukai ir patvirtinti juos pagal tipą ir logiką.

9.1. Jei užklausoje nėra visų užpildytų reikalingų laukų, tada, parodyti klaidos pranešimą iš serverio pusės, kai pranešime vaizduojama klaidos informacija ir perspėjimas, kas negerai.

FRONTEND

1. Sukurti HTML puslapius: login.html, register.html, shop.html, add\_item.html,

orders.html

2. Pridėti stiliaus ir JS failus prie kiekvieno šio puslapio.

3. Sukurti "login.html" puslapį

3.1. Kuriame bus forma su "email" ir "password" laukais.

3.2. Puslapio kūrime naudoti JavaScript, kad būtų galima siųsti "email" ir "password" į serverį.

3.3. Jei "email" ir "password" yra teisingi, išsaugoti vartotojo "email" ir "user\_roles", direktorijoje "localStorage". Išsaugojus informaciją, vartotoją nukreipti į "shop.html".

3.4. Bet jei "email" ar "password" yra neteisingi, tada parodyti klaidos pranešima iš serverio pusės.

4. Sukurti registracijos puslapį.

4.1. Sukurti formą su "name", "email", "password", "repeatPassword" ir "user\_roles" laukais.

4.1.1. "user\_roles" turėty būti "select" įvestis su pasirinkimais iš "user\_roles" lentelės.

4.2. Naudoti JavaScript, kad batų galima siųsti "name", "email", "password" ir "user\_roles" į serverį.

4.3. Jei "email" nėra užimtas, tada sukurti vartotoją su "name", "email", "password" ir "user\_roles".

4.4. Jei "email" yra užimtas, tada parodyti klaidos pranešimą iš serverio pusės.

5. Sukurti "shop.html" puslapį

5.1. Sukurti 7 prekes ir jas atvaizduoti su prisitaikančio dizaino GRID išdėstymu.

5.2. Naudoti JavaScript, kad bitų galima gauti visas parduotuvės prekes iš backend.

5.3. Naudoti JavaScript, kad parodyti visas parduotuvės prekes, kaip prisitaikančio dizaino korteles, GRID formatu.

5.4. Sukurti mygtuką, skirtą pridėti parduotuvės prekę į krepšelį ("orders" table)

5.5. Sukurti mygtuką, skirtą ištrinti parduotuvės prekę iš "shop\_items" lentelės ir "orders" lentelės.

6. Sukurti "add\_item.html" puslapį

6.1. Sukurti formą su "name", "price", "description", "image", 'item\_type\_id" (pasirinkti iš esamų)

6.1.1. "item\_type\_id" turėtų būti "select" įvestis su pasirinkimais iš turimų reikšmių "item\_types" lentelėje.

6.2. Naudoti JavaScript, kad būtų galima siųsti naują parduotuvės prekę į serverį.

6.3. Jei parduotuvės prekė, jau yra sukurta, nukreipti į "shop.html"

6.4. Jei tokios parduotuvės prekės nėra sukurta, rodyti klaidos pranešimą iš serverio pusės

7. Sukurti "orders.html" puslapį

7.1. Sukurti lentelę su užsakymais

7.2. Naudoti JavaScript, kad būtų galima gauti visus užsakymus iš serverio

7.3. Sukurti "select" įvestį, skirtą pasirinkti "users". Jei "user\_id" yra pasirinktas, rodyti tik tuos užsakymus, kurie priklauso tam "user\_id"

8. Kiekvienas puslapis turėtų turėti naršymo juostą su nuorodomis į kitus puslapius ("footer")

8.1. Jei "user\_id" prisijungęs, rodyti atsijungimo mygtuką

8.2. Jei "user\_id" neprisijungęs, rodyti prisijungimo ir registracijos mygtukus.

9. Sukurti atsijungimo mygtuką, atjungti prisijungusiam "user\_id".

9.1. Naudoti JavaScript, kad būtų galima pašalinti "users", "email" ir "role\_id" iš direktorijos "localStorage"

9.2. Nukreipti į "login.html"

Extra užduotis.

1. Jei "user\_id" prisijungęs ir turi "role\_id" - "admin", tokiu atveju rodyti puslapį "add\_item.html" ir nuorodas į jį. Jei ne – slėpti šį puslapį.

2. Jei "user\_id" prisijungęs su "role\_id" - "admin", tada rodyti ištrynimo mygtuką. Jei ne - slėpti šį mygtuką.

3. "orders.html" puslapyje, jei "user\_id" prisijungęs ir turi "role\_id" - "admin", tada rodyti pasirinkimo įvestį, skirtą pasirinkti "user\_id". Jei ne – slėpti šį pasirinkimą.