



FER3

Preddiplomski studij

Računarstvo

Razvoj programske potpore za web

4. laboratorijska vježba — Sjednice

1 Uvod

Tema 4. laboratorijske vježbe je uporaba sjednica u dinamičkom webu. U sklopu pripreme za vježbu potrebno je nadopuniti osnovni programski kôd s funkcionalnošću vezanom uz sjednice kako bi realizirali jednostavan primjer web trgovine. Na samoj vježbi bit će potrebno izmijeniti pripremu, odnosno bit će potrebno dodati nove ili izmijeniti postojeće funkcionalnosti.

Napomena: Kanal za sva pitanja vezana uz 4. laboratorijsku vježbu na Teamsu je dostupan na **poveznici**.

2 Razvojno okruženje

Preporučuje se korištenje editora Visual Studio Code te ekstenzija i alata za otkrivanje i ispravljanje grešaka navedenih u prijašnjim laboratorijskim vježbama.

U sklopu laboratorijske vježbe koristit će se **PostgreSQL**. Upute za PostgreSQL su dostupne u materijalima kolegija “Baze podataka” [ovdje](#).

U laboratorijskoj vježbi se koriste **node.js (v16.15.0 LTS)**, **express** i dodatni express moduli za ostvarenje specifičnih funkcionalnosti (pristup bazi podataka, upravljanje sjednicama, pohrana sjednica u bazi podataka itd.). Pretpostavljamo da ste, zahvaljujući 3. laboratorijskoj vježbi, već upoznati s modulima i postupcima potrebnim za rad s bazom podataka. Prije početka rada na 4. vježbi preporučujemo pogledati dokumentaciju sljedećih modula:

- **express-session** za upravljanje sjednicama
- **connect-pg-simple** za pohranu sjednica u PostgreSQL bazu podataka

3 Organizacija projekta

Slika 1 prikazuje organizaciju projekta. Sve datoteke (odnosno projekt) priložene su na [poveznici](#). Organizacija projekta slijedi organizaciju iz 3. laboratorijske vježbe, s dodatnom mapom *models* koja sadrži definicije modela (uglavnom razreda) osnovnih entiteta korištenih u vježbi.

Najveći dio izmjena u pripremi 4. laboratorijske vježbe bit će potrebno obaviti unutar mape *routes*, te neke manje prilagodbe unutar datoteka mapa *db* i osnovne mape (datoteka *server.js*).

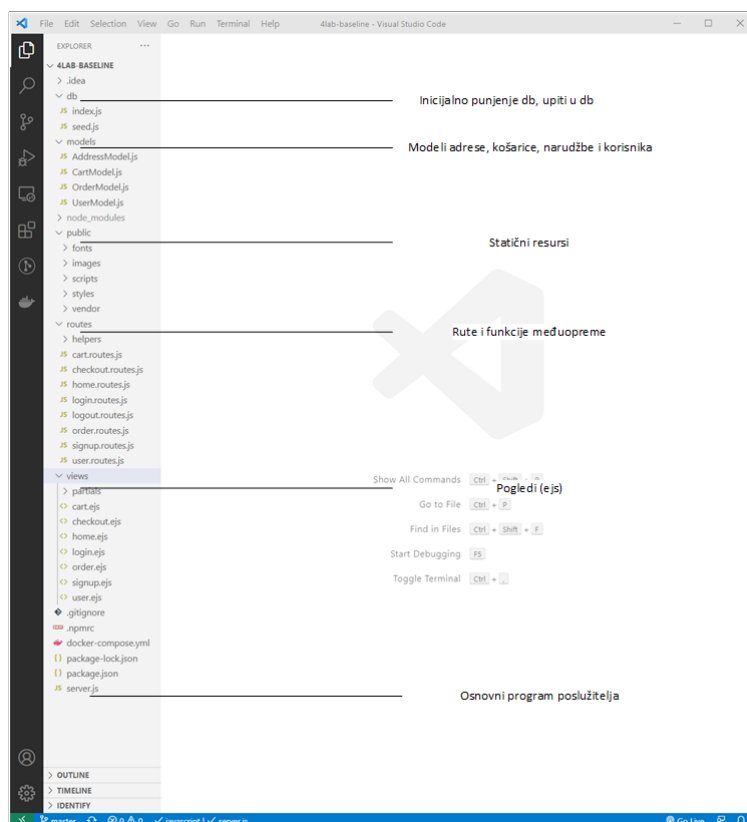
3.1 Middleware funkcije

Middleware funkcije imaju pristup objektu zahtjeva (**req**), objektu odgovora (**res**) i sljedećoj funkciji u ciklusu zahtjeva i odgovora aplikacije. Funkcija **next** je funkcija u Express usmjerivaču koja, kada se pozove, izvršava sljedeću *middleware* funkciju u zadanoj sekvenci.

Middleware funkcije mogu obavljati razne zadatke, npr:

- Izvršavanje proizvoljnog koda.
- Izmjena objekata zahtjeva i odgovora.
- Završavanje ciklusa zahtjeva i odgovora.
- Pozivanje sljedeće *middleware* funkcije u sekvenci

Ako trenutna *middleware* funkcija ne završi ciklus zahtjeva i odgovora, mora pozvati **next** kako bi predala kontrolu na sljedeću funkciju. U suprotnom, zahtjev će ostati blokiran.



Slika 1: Organizacija projekta

4 Priprema

4.1 Preuzimanje projekta i modula i pokretanje projekta

Nakon preuzimanja projekta iz repozitorija, potrebno je dobiti i vanjske ovisnosti navedene u datoteci **package.json** pokretanjem naredbe **npm install** iz korijenske mape projekta. Koristeći naredbu **npm run dev** se pokreće projekt koristeći **nodemon** paket, a pristupamo našem projektu na portu 3000, tj. `localhost:3000`. Nodemon je koristan alat za razvijanje projekta jer on automatski ponovno pokreće aplikaciju kada otkrije promjene datoteka u direktoriju.

4.2 Inicijalizacija baze podataka

Unutar sustava za upravljanje bazom podataka stvorite novu bazu podataka i imenujte ju po želji (npr. *web-lab4*). U datotekama *seed.js* i *index.js* unutar mape *db* postavite ime vaše baze podataka i pristupnu lozinku.

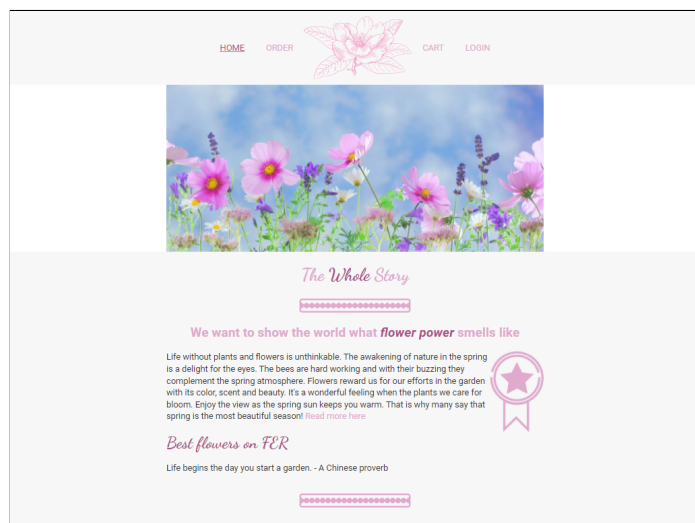
Pokrenite skriptu **seed.js** koja stvara 8 tablica u bazi podataka i neke od njih puni inicijalnim podacima. Uz kategorije i proizvode iz 3. laboratorijske vježbe, inicijalno se stvara i korisnik sustava *admin* (lozinka *admin*). Skripta će stvoriti sve potrebne tablice za rad, uključujući i one potrebne modulu *connect-pg-simple* za čuvanje sjednica u bazi podataka.

4.3 Opis funkcionalnosti web trgovine

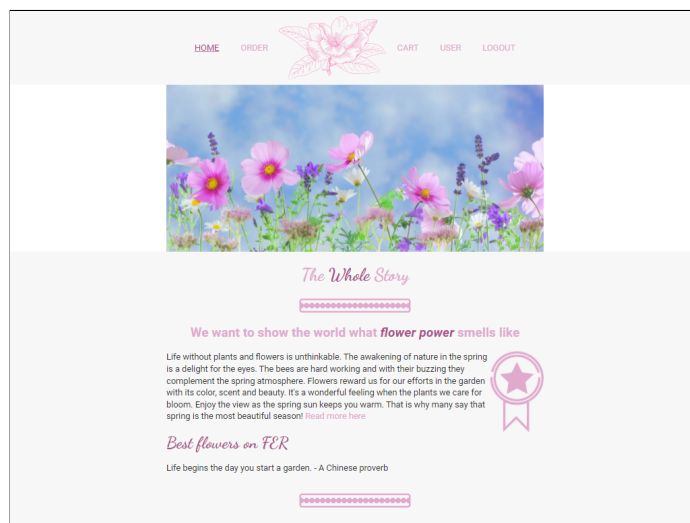
U nastavku se opisuju stranice web trgovine i njihove funkcionalnosti. Funkcionalnosti koje se opisuju su djelomično implementirane, a ostatak funkcionalnosti trebate implementirati u sklopu pripreme. Detaljniji opis zadatka za pripremu naveden je pod 4.5.

4.3.1 Stranica Home

Ako korisnik nije prijavljen, zaglavlje ne sadrži poveznicu **USER** i sadrži poveznicu **LOGIN** (Slika 2). Kada je korisnik prijavljen u sustav, zaglavlje sadrži poveznicu **USER** koja vodi na stranicu pregleda korisničkog računa i poveznicu **LOGOUT** (Slika 3). Ova pravila vrijede za zaglavlje kroz cijelo web sjedište.



Slika 2: Početna stranica — korisnik nije prijavljen



Slika 3: Početna stranica — korisnik je prijavljen

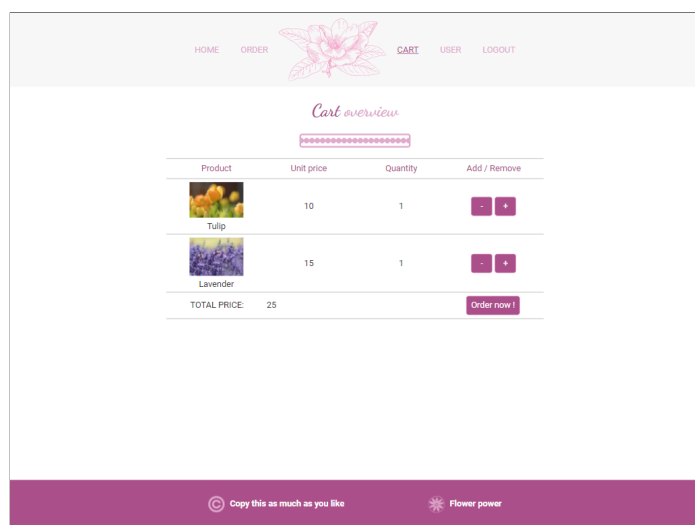
4.3.2 Stranica Order

Stranica Order, kao i u vježbi 3., sadrži popis i slike artikala raspoloživih za kupovinu u web trgovini. Razliku u odnosu na vježbu 3. čine gumbi ispod svakog artikla — njihov odabir dodaje jedan komad odabranog artikla u košaricu. Stranica Order je funkcionalna bez obzira je li trenutni kupac prijavljen u trgovinu ili ne (anoniman kupac).

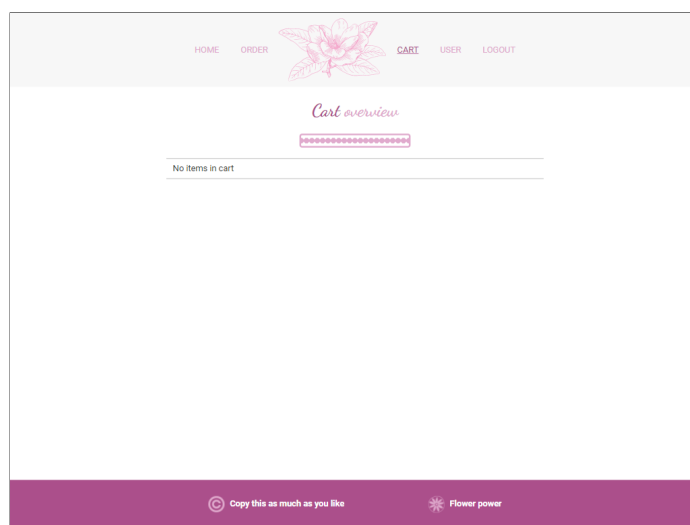
4.3.3 Stranica Cart

Stranica Cart daje pregled trenutnog stanja košarice kupca, kako prikazuje Slika 4 (odabrani artikli, broj pojedinih artikala, jedinična cijena, mogućnost dodavanja ili smanjenja brojnosti pojedinog artikla u košarici). Ako se brojnost nekog artikla smanji na nulu, on se uklanja iz košarice. Ako je košarica trenutno prazna, na stranici se ispisuje odgovarajuća poruka (Slika 5).

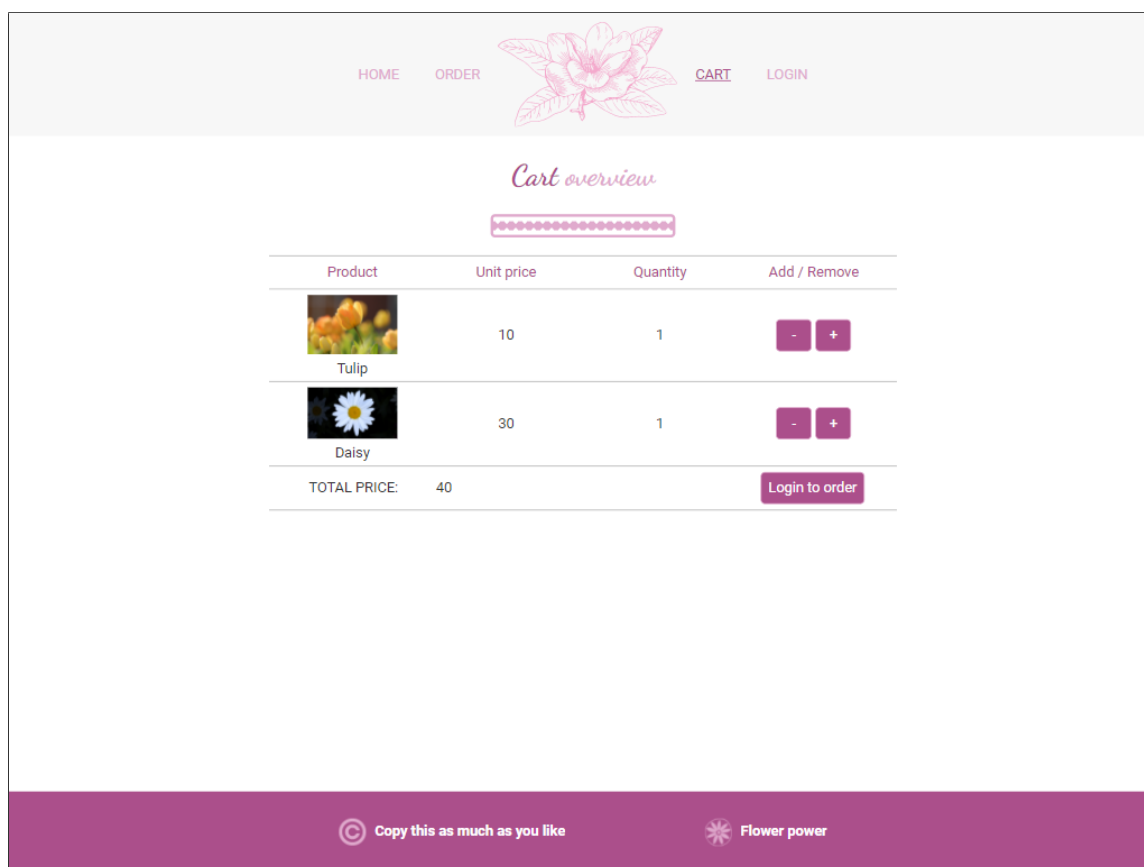
Na dnu popisa artikala nalazi se ukupna cijena svih artikala u košarici, kao i gumb *Order now!* kojim se zaključuje narudžba. U slučaju da korisnik nije prijavljen, umjesto gumba *Order now!* prikazuje se gumb *Login to order*, kojim se kupac dovodi do Login stranice (Slika 6).



Slika 4: Košarica s proizvodima



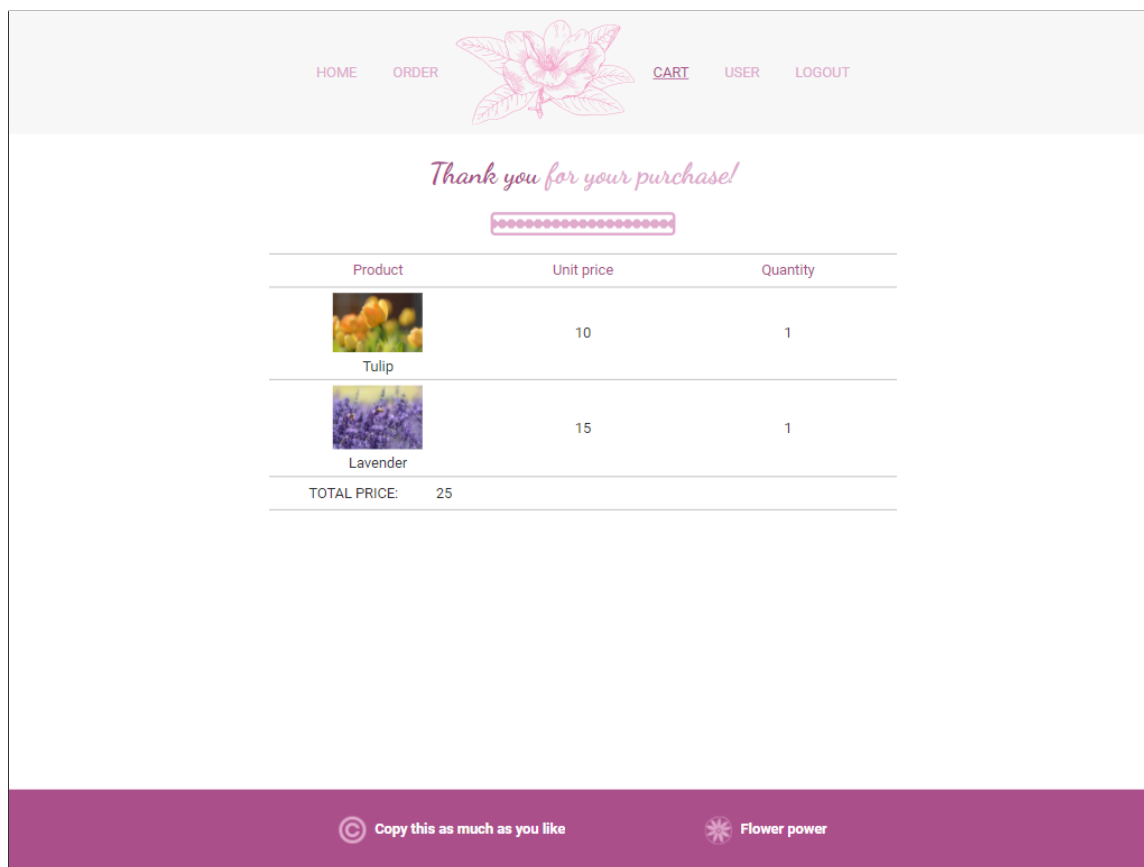
Slika 5: Košarica bez proizvoda



Slika 6: Košarica kada korisnik nije prijavljen

4.3.4 Stranica Checkout

Stranica Checkout daje pregled zaključene narudžbe, te se košarica prazni. Provedena narudžba se treba prikazati unutar povijesti narudžbi registriranog korisnika (Slika 7).



Slika 7: Checkout

4.3.5 Stranica Login

Stranici Login (Slika 8) moguće je pristupiti poveznicom prikazanom u zaglavlju stranice, samo kada trenutni korisnik stranice nije registriran (logiran), inače je na tom mjestu u zaglavlju poveznica na stranicu Logout. Anonimni korisnik se putem stranice Login prijavljuje u sustav (ako posjeduje korisnički račun). Nakon uspješne prijave, postaje prijavljeni korisnik, te ga se preusmjerava na početnu stranicu. U tom postupku se čuva trenutni sadržaj košarice.


Ako anonimni korisnik ne posjeduje korisnički račun, ispisuje se odgovarajuća poruka (Slika 9) te slijeđenjem poveznice *Create your account*, dolazi na stranicu Sign up za stvaranje korisničkog računa.

4.3.6 Stranica User

Stranica User (Slika 11) daje pregled osobnih podataka trenutnog registriranog korisnika sustava, njegovu adresu (za sada jedna adresa po korisniku), te povijest narudžbi (datum narudžbe i iznos). Ako neprijavljeni korisnik pokuša pristupiti stranici, bit će prosljeđen na stranicu Login, uz odgovarajuću poruku (Slika 10).

4.3.7 Stranica Logout

Stranica Logout ne postoji kao takva, odabirom poveznice Logout u zaglavlju odjavljuje se trenutno prijavljeni korisnik sustava, a sadržaj košarice se briše. Nakon odjave korisnik je preusmjeren na osnovnu stranicu.

HOME ORDER  CART LOGIN

Existing User

.....

User login form

User name:


.....

Password:


.....

Submit Reset

New customer? Create your account.

© Copy this as much as you like  Flower power

Slika 8: Login

HOME ORDER  CART LOGIN

Existing User

.....

User login form

User name:

.....


Password:

.....


Submit Reset

User does not exist or incorrect password.

New customer? Create your account.

© Copy this as much as you like  Flower power

Slika 9: Login — nepostojeći korisnik

HOME ORDER  CART LOGIN

Existing User

.....

User login form

User name:

.....


Password:

.....


Submit Reset

Please login to view the requested page.

New customer? Create your account.

© Copy this as much as you like  Flower power

Slika 10: Login — neprijavljeni korisnik pokušava pristupiti stranici koja zahtijeva da je korisnik prijavljen

[HOME](#)
[ORDER](#)

[CART](#)
[USER](#)
[LOGOUT](#)

User profile



User name	admin
First name	Adminko
Last name	Administratović
E-mail	null@admin

User addresses

Name:	Dragi Admin
Street:	Za konzolom b.b.
Postcode:	42
City:	Igdje
Country:	Globalija

Past orders


No orders yet

 Copy this as much as you like
  Flower power

Slika 11: User

4.3.8 Stranica Signup

Na stranici Signup (Slika 12) je moguće otvoriti novi korisnički račun. Korisničko ime (Username) i Email ne smiju prethodno postojati u sustavu, a obje unesene lozinke se moraju podudarati (Slika 13). Polja Username, Email, Password i Confirm password su obavezna i ispisuje se poruka da se polja moraju popuniti. Nakon uspješnog otvaranja korisničkog računa, novi korisnik je preusmjeren na osnovnu stranicu sjedišta i odmah je prijavljen u sustav.

HOME ORDER  CART LOGIN

New User

—User registration form—

Username:

E-mail:

First name:

Last name:

Password:


Confirm password:

Street:


Post/zip code:

City:

Country:

© Copy this as much as you like  Flower power

Slika 12: Signup

HOME ORDER  CART LOGIN

New User

—User registration form—

Username:

E-mail:

First name:

Last name:

Password:

Confirm password:


Street:

Post/zip code:

City:

Country:

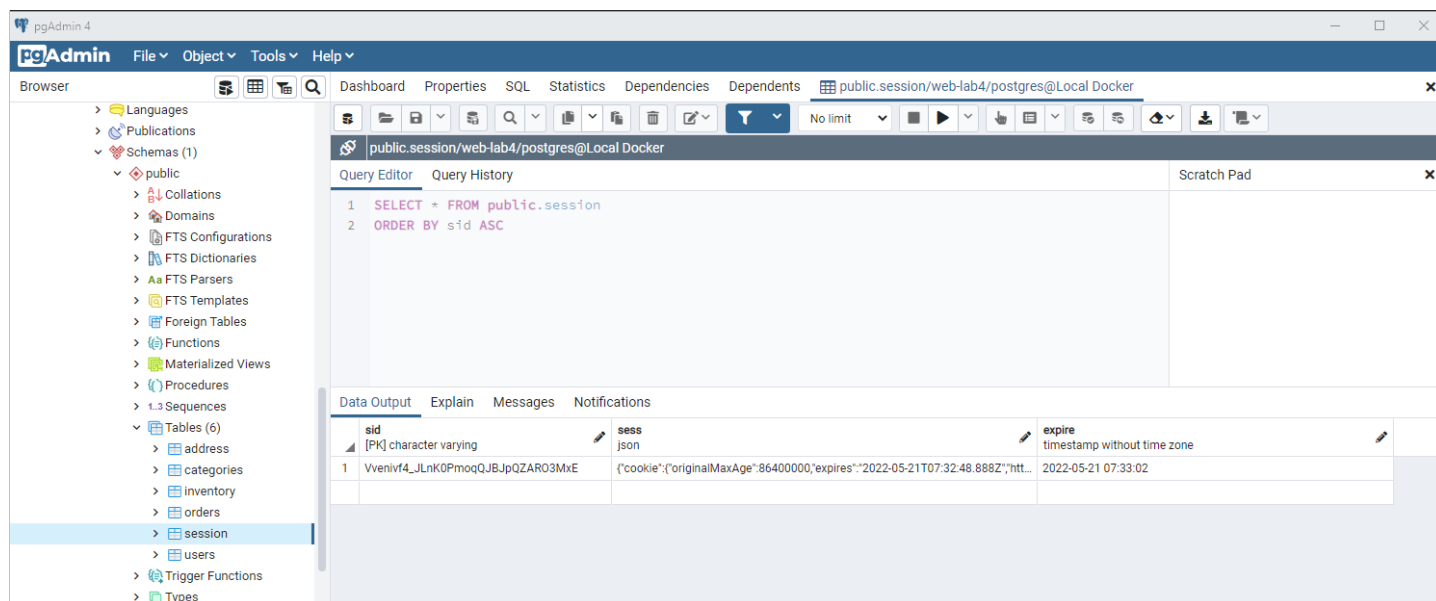
Username already exists

© Copy this as much as you like  Flower power

Slika 13: Signup — greška

4.4 Implementacija košarice

Svatom novopristiglom korisniku na stranice web trgovine automatski se dodjeljuje sjednica (stvara se sjednički kolačić i zapis u tablici *session* unutar baze podataka — Slika 14). To se događa automatski, od strane middleware-a *express-session*.



Slika 14: Sjednice u bazi podataka

Stanje košarice proizvoda treba pamtit na razini pojedine sjednice, kao i informaciju je li korisnik sjednice anonim (nije logiran u sustav) ili je registrirani (koji?) korisnik sustava (prošao login). Sadržaj košarice se briše ili provedenim postupkom kupovine (kada se sadržaj košarice seli u zapis narudžbe, koja zapisuje u bazu podataka) ili nakon odjave iz sustava (logout).

Implementacijski detalj: middleware *express-session* dodaje objektu *request* objekt *session*, u koji je onda moguće dodavati druge varijable ili objekte, a koji će biti sačuvani između transakcija (HTTP upit-odgovor) koje pripadaju istoj sjednici (tj. obilježene istim kolačićem). Sadržaj sjednice se čuva unutar *sess* polja tablice *session* u JSON formatu — što znači da će svi elementi objekta *session* koje nije moguće zapisati u formatu JSON biti izgubljeni na završetku svake transakcije! Pogledajte detaljnije vrijednosti sačuvane u spomenutom polju tablice *session*!

4.5 Vaš zadatak za pripremu

Vaš zadatak za pripremu je:

- proučiti dostupan programski kôd vježbe
- nadopuniti **cart.routes.js**
- nadopuniti **login.routes.js**
- nadopuniti **logout.routes.js**
- nadopuniti **signup.routes.js**
- nadopuniti **server.js**

Mjesta na kojima treba nadopuniti programski kôd su označena ključnom riječi **ZADATAK** i komentarima/uputama koje treba slijediti.

U datoteci **server.js** potrebno je:

1. postaviti *express-session* middleware u lanac middleware funkcija.

U datoteci **signup.routes.js** potrebno je, ako je prijava uspjela, dodati sljedeću funkcionalnost:

1. povezati sjednicu s registriranim korisnikom.
2. napraviti redirect na početnu stranicu.

U datoteci **logout.routes.js** potrebno je:

1. obrisati sadržaj košarice.
2. napraviti redirect na osnovnu stranicu.

U datoteci **login.routes.js** potrebno je:

1. implementirati mehanizam prijave korisnika u sustav (provjeru postojanja, provjera lozinke, povezivanje korisnika).
2. ako je provjera uspjela, osigurati očuvanje sadržaja košarice.

U datoteci **cart.routes.js** potrebno je:

1. implementirati prikaz sadržaja košarice uz pomoć `cart.ejs`.
2. implementirati rutu `/cart/add/:id` za dodavanje jednog komada artikla (određenog id-om) u košaricu.
3. implementirati rutu `/cart/remove/:id` za brisanje jednog komada artikla (određenog id-om) iz košarice. Ako broj komada navedenog artikla padne na nulu, taj artikl treba ukloniti iz košarice. Funkcionalnost dodavanja i brisanja artikala obavlja se gumbima na stranicama Order i Cart.

Preporuka je da u vašim rješenjima što je više moguće koristite modele iz mape *models*, zbog riješene pohrane i dohvata podataka iz baze podataka. Također ispitajte ponašanje sustava s obzirom na (i) registriranog korisnika koji istovremeno koristi dva različita preglednika za pristup web trgovini, kao i na slučaj (ii) dva različita registrirana korisnika koji istovremeno pristupaju web trgovini (ne smije doći do miješanja sadržaja košarica).

Upozoravamo na asinkronost operacija, osobito onih u kojima je uključen pristup bazi podataka. Prisjetite se procesnog modela koji koristi `node.js`!

5 Često postavljena pitanja

- **Smijem li izmijeniti neku od datoteka koja nije zadana za izmjenjivanje?**
 - Nije predviđeno, pa nije ni preporučljivo. Izmjena može izazvati greške u radu u drugim datotekama, koje nisu predviđene za mijenjanje.
- **Kako želimo da se košarica ponaša u scenariju: uđemo na stranicu (anonimna sjednica) i odaberemo par proizvoda, košarica nije prazna, zatim napravimo signup što napravi automatski login. Želimo da sadržaj košarica ostane ili se isprazni?**
 - Sadržaj košarice je očuvan.