FER3

Preddiplomski studij

Računarstvo

Razvoj programske potpore za web i pokretne uređaje

4. laboratorijska vježba - Dinamički web - Zadatci

Ak. god. 2019./2020.

1 Uvod

Uz izradu ranije zadane pripreme, potrebno je odraditi dodatni zadatak (uz prilagodbu pripreme). Zadaci se temelje na promjenama ili proširenju funkcionalnosti pripremnog zadatka za 4. laboratorijsku vježbu, a mogu uključivati i korištenje dijelova programskog kôda primjera danih u sklopu predavanja.

Odredite koji zadatak vam je pridodijeljen prema datoteci na Moodleu (**lab4-JMBAG-zadatak.txt**). U datoteci su sadržani JMBAG i redni broj zadatka, npr. studentu s JMBAG-om 0036511119 određen je 1.zadatak:

- 0036511119 1

Od vas se očekuje da predate arhivu projekta odnosno .zip putem sustava Moodle prije isteka roka. Imenujte zip: **JMBAG_4_BROJ_ZADATKA**, npr. student s JMBAG-om 0036511119, a koji rješava 1. zadatak imenuje arhivu 0036511119_4_1.

VAŽNO:

- mapu node_modules ne uključujte u arhivu!
- datoteku package-lock.json UKLJUČITE u arhivu, bez nje nećemo moći dobaviti potrebne module i pokrenuti vašu vježbu, stoga je nećemo moći ni ocijeniti.

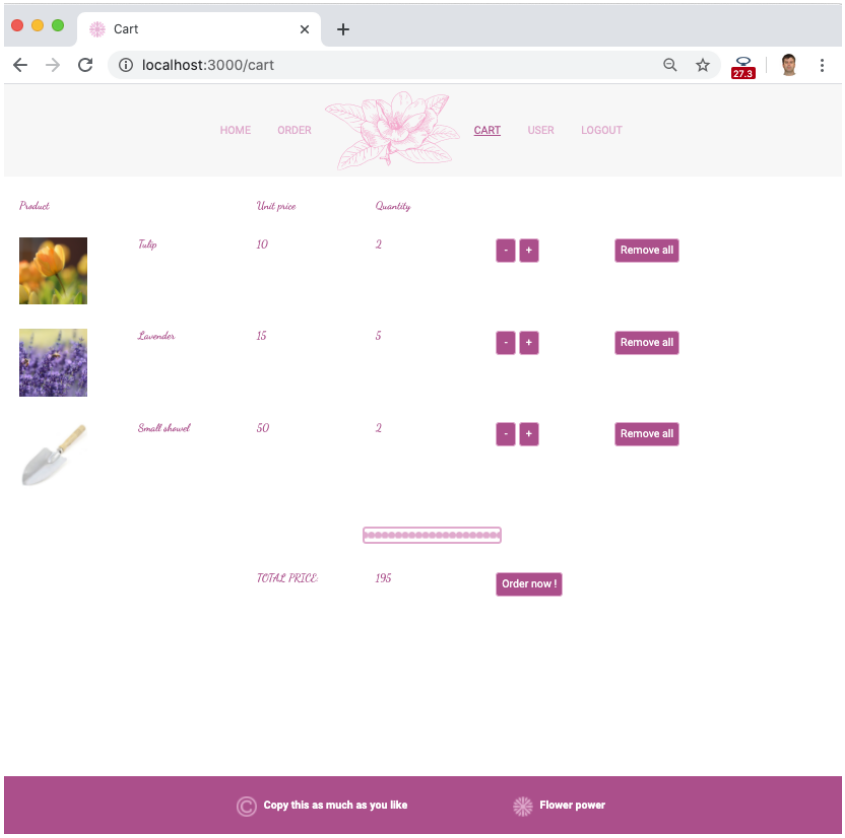
2 Zadatci

U nastavku su detaljno objašnjeni svi zadatci.

2.1 1. zadatak - brisanje proizvoda iz košarice

U sklopu pripreme za 4. laboratorijsku vježbu bilo je potrebno dovršiti funkcionalnost stranice Cart, na kojoj je prikazan popis proizvoda u košarici, njihova jedinična cijena i brojnost, kao i ukupna cijena proizvoda u košarici. Korisnik je imao mogućnost povećanja i smanjenja količine pojedinog proizvoda u košarici korištenjem odgovarajućih gumbi + i -. Samo kada se brojnost pojedinog proizvoda u košarici smanjila na nulu (korištenjem gumba -), taj je proizvod brisan iz košarice.

Vaš zadatak je implementacija trenutnog brisanja proizvoda iz košarice, bez obzira na njegovu trenutnu brojnost. Na stranici Cart postavite gumb **Remove all** u stupac desno od gumba + i -. Odabirom gumba briše se proizvod iz košarice i osvježava prikaz njenog trenutnog stanja.



Slika 1: Primjer stranice za pregled sadržaja košarice s dodatnom opcijom za trenutno brisanje pojedinog proizvoda

Zadatak ne propisuje samo jedno točno rješenje, kao pomoć možemo nabrojati elemente koje bi se moglo izmijeniti/dodati u cilju postizanja željene funkcionalnosti:

- izmijeniti definiciju css-a vezanu uz prikaz sadržaja košarice
- izmijeniti predložak prikaza košarice, dodati gumb i potrebnu *callback* funkciju
- dodati novu rutu za prihvat zahtjeva za brisanja proizvoda na strani poslužitelja
- nadograditi model košarice s metodom brisanja proizvoda

Procijenjena složenost rješenja je cca. do 5-10 novih redaka u definiciji CSS, do 10 novih redaka u ejs predlošku, do 10 redaka novog JavaScript kôda.

2.2 2. zadatak - košarica u kontekstu korisnika

U sklopu pripreme za 4. laboratorijsku vježbu realizirana je web trgovina, u kojoj je košarica implementirana na razini sjednice. To je za posljedicu imalo da ako je isti korisnik bio prijavljen u dućan s dva različita preglednika (ne dijele kolačiće), u svakom pregledniku korisnik je radio s neovisnim košaricama. Vaš zadatak je prebaciti košaricu iz konteksta sjednice u kontekst korisnika; u slučaju da prijavljeni korisnik koristi dva različita preglednika, unutar oba preglednika radit će s istom košaricom. Ako korisnik nije prijavljen, radit će s košaricom na razini sjednice. Dodatno, za razliku od pripreme, u rješenju zadatka nije potrebno čuvati sadržaj košarice tijekom postupka prijave (login) anonimnog korisnika.

Predlažemo da za implementaciju košarice na razini korisnika iskoristite razred *UserData*, poznat vam iz primjera sedmog s 3. predavanja iz dinamičkog web-a. U navedenom primjeru smo u korisničkom kontekstu čuvali brojač pristupa stranici. Koristeći prikazani princip čuvanja korisničkog konteksta, u rješenju ovog zadatka brojač treba zamijeniti s košaricom. Datoteku *UserData.js* dohvatite iz repozitorija primjera uz predavanja (git) ili iz dodataka za ovu vježbu na Moodle-u.

Naglašavamo da *UserData* koristimo samo za čuvanje korisničkog konteksta - upravljanje korisničkim podacima (stvaranje korisnika, provjera passworda itd.) ostaje nepromijenjeno iz pripremne vježbe. Kod stvaranja korisnika (metoda *createUser()*) dovoljno je proslijediti samo *username* parametar, ostali argumenti će poprimiti vrijednost undefined (svojstvo jezika JavaScript), i nisu nam u ovom slučaju bitni. Za *username* parametar predlažemo da koristite korisnikov e-mail.

Za razliku od express-session modula, koji samostalno pohranjuje podatke iz sjednice netom prije slanja odgovora od poslužitelja prema pregledniku, podatke iz UserData bi bilo dobro eksplicitno pohraniti u datoteku pozivom metode *store()* prije kraja izvođenja middleware funkcije koja je izmijenila podatke u košarici. Ako i zaboravite pohraniti podatke, nećete ih izgubiti između dva poziva, osim ako poslužitelj ne prestane s radom.

U programskom kôdu pripreme podacima u košarici većinom pristupate korištenjem izraza *req.session.cart*. Kako podaci o košarici u rješenju zadataka više ne mogu biti na razini sjednice, možete odabrati dva pristupa:

- Prilagoditi sve izraze u programu s kojima pristupate košarici (nešto tipa *req.app.users.cart*, slično kao kod pristupa vrijednosti brojača u sedmom primjeru)
- Ostaviti programski kôd koji služi radu s košaricom nepromijenjen (*req.session.cart*), ali napisati middleware funkciju koja će, kod svakog zahtjeva od strane klijenta, mapirati odgovarajuću košaricu iz korisničkog konteksta (iz UserData) u sjednički kontekst! Ovaj pristup je preporučen.

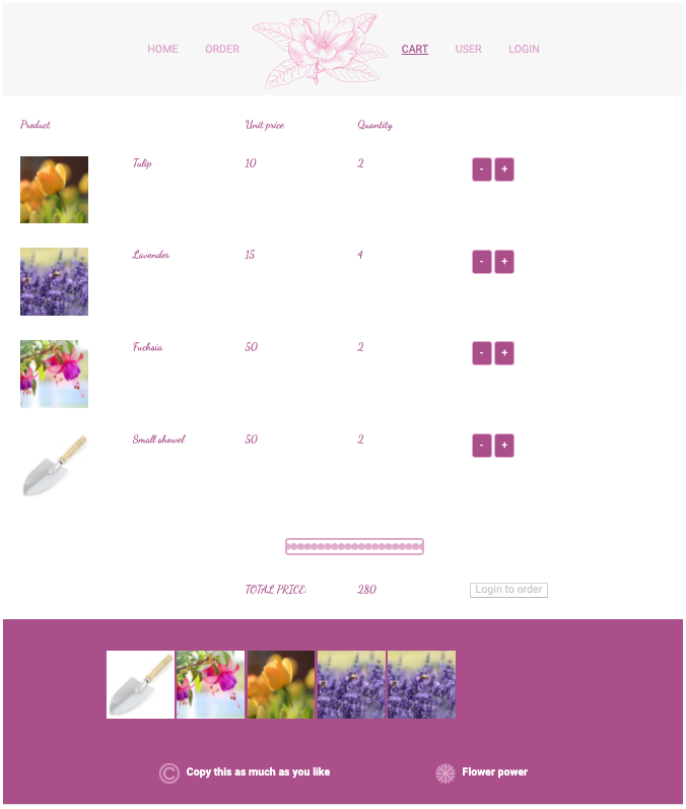
Zadatak ne propisuje samo jedno točno rješenje, kao pomoć možemo navesti elemente koje bi se moglo izmijeniti/dodati u cilju postizanja željene funkcionalnosti:

- U projekt dodati UserData modul iz 7. primjera 3. predavanja o dinamičkom webu
- U server.js dodati middleware funkciju koja mapira košaricu trenutnog korisnika u sjednicu
- U server.js inicijalizirati UserData objekt (redci 69-70 app.js iz 7. primjera)
  app.users = new userData('users.json');
  app.users.initialize(true);
- Prilikom dodavanja i oduzimanja proizvoda iz košarice pohraniti trenutni korisnički kontekst
- Nakon stvaranja narudžbe, obrisati sadržaj košarice i pohraniti trenutni korisnički kontekst

Procijenjena složenost (preporučenog) rješenja je: preslikan UserData modul iz primjera, 20-tak redaka JavaScript kôda.

2.3 3. zadatak - povijest dodavanja proizvoda u košaricu

Zadatak je dodati element stranice neposredno iznad footera na stranicama **Cart** i **Checkout** (kao na slici 2) u kojem se prikazuju slike do pet zadnje dodanih proizvoda u košaricu. Pod pojmom dodavanja u košaricu podrazumijevamo dodavanje proizvoda na stranici Order (gumb Add to Cart) i povećavanje brojnosti proizvoda koji je već u košarici (gumb + na stranici Cart). Krajnje lijevi prikazani proizvod u ovakvoj "FIFO strukturi" zadnje je dodani proizvod u košaricu. Podaci o zadnje dodanim proizvodima čuvaju se u **kontekstu sjednice**.



Slika 2: Primjer prikaza povijesti dodavanja proizvoda u košaricu

Zadatak ne propisuje samo jedno točno rješenje, kao pomoć možemo navesti elemente koje bi se moglo izmijeniti/dodati u cilju postizanja željene funkcionalnosti:

- dodati objekt u sjednicu koji čuva povijest dodavanja do pet zadnjih proizvoda u košaricu
- dodati metodu u model košarice koji vraća podatke o proizvodu
- stvoriti definiciju parcijalnog predloška koji se brine o prikazu povijesti
- dodati parcijalni predložak u Cart i Checkout predloške

Procijenjena složenost rješenja je: 20-tak redaka JavaScript kôda, do 15 redaka u ejs predlošcima.