JUNIORDev DIGITALNA DALMACIJA

CRUD operacije > Promjena podataka

Promjena podataka

Na kraju nam je još preostao još jedan dio CRUD funkcionalnosti - ***Update*** tj. promjena ili osvježavanje podataka.

HTTP protokol podržava dvije metode za osvježavanje podataka - PUT i PATCH. Ove metode su na prvi pogled identične ali postoji bitna razlika pa ćemo se kratko osvrnuti na obje metode.

PUT metoda

PUT metoda služi za osvježavanje već postojećeg resursa na poslužitelju. Glavna karakteristika ove metode je u tome što zamjenjuje **cijeli** resurs sa podatkom kojeg smo poslali u tijelu zahtjeva. Čak i ako želimo promijeniti samo dio podatka, u tijelu zahtjeva moramo poslati potpuni podatak.

Na primjer ako imamo strukturu podatka sa tri polja: "ime", "mail" i "godine", PUT zahtjev mora u svom tijelu (*body*) sadržavati podataka u obliku:

```
{
  "ime": "Ana Anić",
  "mail": "anaanic@mail.com",
  "godine": 24
}
```

Zahtjev šaljemo na adresu već postojećeg resursa, najčešće po ID-u. Na primjer u gornjem slučaju bi adresi bila u obliku `"/osobe/1"` i umjesto postojećih podataka za osobu sa `ID: 1` bi se spremili gore navedeni podaci. Čak i kada bi npr. htjeli samo promijeniti godine, PUT zahtjev mora sadržavati i ostale podatke kako bi ostali spremljeni na poslužitelju.

PATCH metoda

PATCH metoda se također koristi za modifikaciju postojećih resursa na poslužitelju ali umjesto zamjene cijelog podatka osvježava se samo dio resursa, onaj koji se nalazi u tijelu samog zahtjeva.

Korištenjem ove metode možemo raditi parcijalnu zamjenu podataka što može doprinijeti boljoj efikasnosti aplikacije i smanjenom mrežnom prometu. Uzmemo li u obzir prethodni primjer i situaciju kada želimo samo promijeniti godine, dovoljno bi bilo uz PATCH zahtjev poslati podatak u obliku:

```
{
  "godine": 25
}
```

U ovom slučaju će se odabranom podatku (pretpostavimo da je adresa ponovno `"/osobe/1"`) promijeniti samo svojstvo "godine" na novu vrijednost, dok će sve ostale vrijednosti ostati nepromijenjene.

Iz ovih primjera je očito i kada izabrati koju metodu. Ukoliko želite zamijeniti cijeli podatak na poslužitelju, koristite PUT metodu. S druge strane, djelomične promjene podataka je bolje raditi uz pomoć PATCH metode. Naravno, to sve ovisi i o implementaciji poslužitelja i koje metode podržava.

Axios primjeri

Pogledajmo sada jednostavni primjer kako koristiti ove metode pomoću Axios biblioteke na našoj trenutnoj aplikaciji. Ukoliko otvorite datoteku koja nam služi kao "baza podataka", vidjeti ćemo strukturu podatka za klase rezervacija koje izgleda ovako:

```
"klase": [
```

```
  {
    "id": 1,
    "ime": "Ekonomska",
    "oznaka": "E"
  },
  {
    "id": 2,
    "ime": "Poslovna",
    "oznaka": "B"
  }
]
```

Recimo da želimo promijeniti oznaku poslovne klase na način da umjesto prvog slova engleske riječi (B za *business*) bude slovo "P". S obzirom da se radi o djelomičnoj promjeni, dovoljno nam je koristiti PATCH zahtjev.

PATCH zahtjev

Pisanje PATCH zahtjeva uz *Axios* je gotovo identično kao i POST zahtjev. Pozivamo metodu `axios.patch()` kojoj kao prvi argument šaljemo adresu resursa kojeg modificiramo dok je drugi argument tijelo (*body*) zahtjeva koje sadrži dio resursa kojeg mijenjamo. Kako bi pojednostavili primjer, zahtjev šaljemo pritiskom na tipku, bez unosa korisnika.

components/Promjena.jsx

```
import axios from "axios";

function Promjena() {
  function saljiZahtjev() {
    axios.patch("http://localhost:3001/klase/2", {
      oznaka: "P",
    })
    .then(rez => console.log(rez))
  }
  return <button onClick={saljiZahtjev}>Promjena kategorije</button>;
}

export default Promjena;
```

Nakon što uključimo ovu komponentu u glavni dio aplikacije i napravimo novu rezervaciju sa

poslovnom klasom, vidjeti ćemo kako se promijenila oznaka. Naravno, stare rezervacije se neće promijeniti (ne koristimo relacijsku bazu).

PUT zahtjev

Možemo sada pogledati i pisanje PUT zahtjeva. Vratiti ćemo oznaku na englesku verziju. Za početak možete samo u prethodnom zahtjevu promijeniti vrijednost oznake na "B" i u pozivu zamijeniti metodu sa `axios.put()` verzijom. Ako pošaljete taj zahtjev i osvježite stranicu, uočiti će da je nestala oznaka uz *radio button* za poslovnu klasu. To je zato jer smo poslali samo djelomični podatak i sada na poslužitelju uz podatak "oznaka" nedostaje podatak "ime" za resurs `"/klase/2"`

Ispravna verzija PUT zahtjeva bi izgledala ovako:

components/Promjena.jsx

```
import axios from "axios";

function Promjena() {
  function saljiZahtjev() {
    axios.put("http://localhost:3001/klase/2", {
      ime: "Poslovna",
      oznaka: "B",
    })
    .then(rez => console.log(rez))
  }
  return <button onClick={saljiZahtjev}>Promjena kategorije</button>;
}

export default Promjena;
```

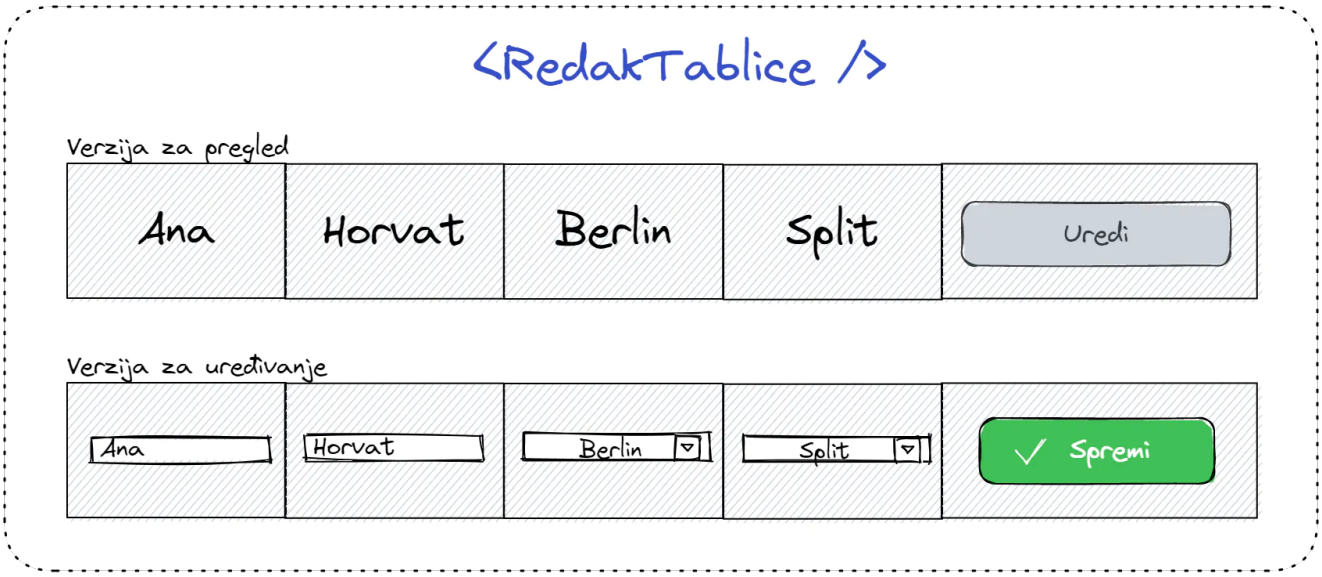
Zadatak

Sada kada znamo koristiti PUT i PATCH metode, modificirajte aplikaciju na način da možete po želji promijeniti dio rezervacije (npr. odredište ili ime i prezime putnika). Pripazite kod promjene rezervacija - ako je vrijednost nekog svojstva objekt, onda i kod PATCH metode morate poslati

cijeli objekt (ne radi višerazinsko djelomično osvježavanje). Npr. vrijednost svojstva "osoba" je objekt sa poljima "ime", "prezime" i "godine" pa ako želite promijeniti samo jedan od ta tri podataka, svejedno morate poslati cijeli objekt (ali ne morate slati svojstvo "karta").

Jednostavniji način je napraviti novu komponentu sa novim poljima za unos ID-a i novim podacima - slično kao DELETE primjer. U tom slučaju dodajte novu komponentu i omogućite korisniku da upiše ID podatka kojeg želi modificirati te dodajte odgovarajuća polja za unos novih podataka. S obzirom da mijenjate samo dio podatka, morati ćete dohvatiti podatak sa poslužitelja (vidi napomenu u prethodnom odlomku.)

Složeniji način (ali sa boljim korisničkim iskustvom) bi bio modifikacija komponente za prikaz retka tablice. Dodajte novi stupac tablice - u svakom retku bi se trebala prikazati tipka "Uredi". Pritiskom na tu tipku, promijenio bi se izgled komponente u tom retku na način da se umjesto "običnog" teksta sada prikazuju polja za unos sa trenutnim podacima - za ime, prezime i godine bi se prikazala obična polja za unos teksta, dok za gradove i klasu možete prikazati *select* elemente kako bi se odabrala nova vrijednost. Automatski se i tipka "Uredi" mijenja u tipku "Spremi" koja će nakon pritiska slati zahtjev sa novim podacima. Ovaj pristup zahtijeva malo više truda i korištenje uvjetnog renderiranja - imamo dvije verzije iste komponente ("*RedakTablice*"), jedan kada komponenta služi za prikaz, a druga kada komponenta služi za uređivanje. Prikaz komponente može ovisiti o internom stanju komponente.



Kod ovog pristupa nije potrebno raditi dodatne zahtjeve za prikaz podataka jer kroz *props* imamo sve potrebne podatke o elementu kojeg prikazujemo/uređujemo, čak i njegovu ID

vrijednost.

Last updated on March 27, 2023

Brisanje podataka

Praktični dio