JUNIORDev DIGITALNA DALMACIJA

CRUD operacije > Dohvat podataka

Dohvat podataka

Započinjemo sa ponavljanjem prethodne radionice - slanje zahtjeva za dohvat podataka na poslužitelj i obrada odgovora. Postoje 4 osnovna tipa HTTP zahtjeva - GET, POST, PUT/PATCH i DELETE. Svaki od tih zahtjeva odgovara jednoj od CRUD operacija. Trenutno želimo pročitati (***read***) podatke sa poslužitelja te ćemo koristiti ***GET*** vrstu zahtjeva.

GET zahtjevi su zadana (*default*) vrsta zahtjeva i njih zapravo pozivamo svaki put kada upišemo neku web adresu u web preglednik (on šalje GET zahtjev za dohvat stranice na toj adresi). Zbog toga prilikom GET zahtjeva često nije niti potrebno navoditi da se radi o GET zahtjevu (kao što to nismo radili na prošloj lekciji sa `fetch()` metodom).

Automatski dohvat

Prvi korak nam je prilikom pokretanja aplikacije poslati zahtjev za dohvat svih postojećih rezervacija na poslužitelj. Za tu svrhu možemo iskoristiti *useEffect hook* i dohvaćeni niz podataka spremiti u varijablu stanja komponente.

App.jsx

```
import "./App.css";
import axios from "axios";
import { useState, useEffect } from "react";

function App() {
  const [rezervacije, postaviRezervacije] = useState([]);

  useEffect(() => {
    axios
      .get("http://localhost:3001/rezervacije/")
      .then(res => postaviRezervacije(res.data));
  }, []);
```

```
  return (
    <div className='App'>
      <h2>Popis rezervacija</h2>
    </div>
  );
}

export default App;
```

Sada nam je cijeli niz rezervacija spremljen u varijablu stanja `rezervacije` (podaci se dohvaćaju samo prilikom prvog renderiranja) te možemo početi sa izgradnjom sučelja za prikaz tih podataka.

Prikaz podataka

Iskoristiti ćemo mogućnosti Reacta i "razbiti" našu aplikaciju na komponente. Napraviti ćemo komponentu `<Tablica />` kojoj ćemo proslijediti podatke za prikaz. Na ovaj način nam glavni dio aplikacije ostaje pregledan.

App.jsx

```
import "./App.css";
import axios from "axios";
import { useState, useEffect } from "react";
import Tablica from "./components/Tablica";

function App() {
  const [rezervacije, postaviRezervacije] = useState([]);

  useEffect(() => {
    axios
      .get("http://localhost:3001/rezervacije/")
      .then(res => postaviRezervacije(res.data));
  }, []);

  return (
    <div className='App'>
      <h2>Popis rezervacija</h2>
      <Tablica rezervacije={rezervacije} />
    </div>
```

```
    );
  }

  export default App;
```

Komponenta "Tablica"

Komponentu za tablicu ćemo također podijeliti na dva dijela - glavni izgled tablice i njeno zaglavlje (*header*) ćemo definirati u samoj komponenti, dok ćemo za prikaz pojedinog retka iskoristiti još jednu komponentu (`<RedakTablice />`), čime nam je `map()` izraz ostao jednostavan. Za slanje podataka između komponenti koristimo *props*.

components/Tablica.jsx

```
import RedakTablice from "./RedakTablice";

function Tablica({ rezervacije }) {
  return (
    <table>
      <thead>
        <tr>
          <th>ID</th>
          <th>Ime</th>
          <th>Prezime</th>
          <th>Polazište</th>
          <th>Odredište</th>
          <th>Klasa</th>
        </tr>
      </thead>
      <tbody>
        {rezervacije.map(r => (
          <RedakTablice key={r.id} rez={r} />
        ))}
      </tbody>
    </table>
  );
}

export default Tablica;
```

U prethodnoj komponenti nemamo neku posebnu složenost. Samo definiramo izgled tablice i

pomoću `map()` metode generiramo onoliko redaka tablice koliko imamo podataka. Uočite korištenje ***key*** svojstva kako bi svaki element imao svoj jedinstveni identifikator.

Na kraju nam još ostaje definirati izgled komponente `<RedakTablice />`, što ne bi trebao biti poseban problem. Pripazite na imena *props* vrijednosti i pravilno destrukturiranje.

components/RedakTablice.jsx

```
function RedakTablice({ rez }) {
  return (
    <tr>
      <td>{rez.id}</td>
      <td>{rez.osoba.ime}</td>
      <td>{rez.osoba.prezime}</td>
      <td>{rez.karta.pocetak}</td>
      <td>{rez.karta.kraj}</td>
      <td>{rez.karta.klasa}</td>
    </tr>
  );
}

export default RedakTablice;
```

Sada bi se podaci trebali ispravno dohvaćati i prikazivati na sučelju aplikacije. Po želji možete urediti izgled komponenti koristeći jedan od načina oblikovanja koje smo koristili u prethodnoj lekciji.

‹ Instalacija Slanje podataka ›