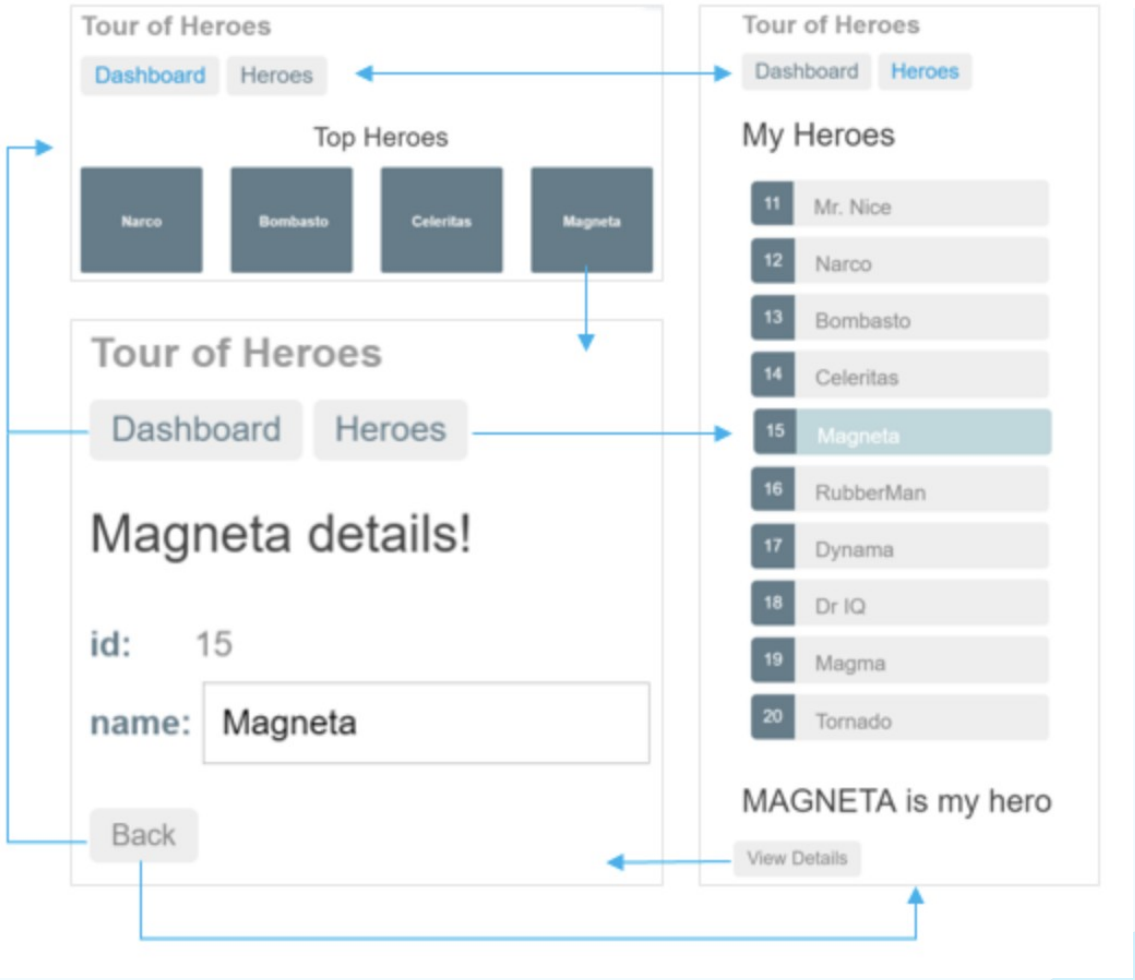
React – Rutiranje

Slađana Turudić - sladjanaturudic@uns.ac.rs
Teodora Ruvčeski - teodoraruvceski@uns.ac.rs

Routing


Tour of Heroes
Dashboard
Heroes
Top Heroes
Narco
Bombasto
Celeritas
Magneta
Tour of Heroes
Dashboard
Heroes
Magneta details!
id: 15
name: Magneta
Back
Tour of Heroes
Dashboard
Heroes
My Heroes
11 Mr. Nice
12 Narco
13 Bombasto
14 Celeritas
15 Magneta
16 RubberMan
17 Dynama
18 Dr IQ
19 Magma
20 Tornado
MAGNETA is my hero
View Details

Routing

- Omogućuje:
  - Navigaciju između različitih prikaza
  - Razmenu parametara između komponenti koje upravljaju prikazima
- S obzirom na činjenicu da je aplikacija stablo komponenti, rutiranje se svodi na zadavanje komponenti za rute
- Rutiranje se postiže pravilima za rutiranje
- Kako bi omogućili rutiranje u Reactu potrebno je:
  - instalirati odgovarajuću biblioteku
  - podesiti ruter
  - definisati rute
  - rukovati navigacijom

React Router

- Najpopularnija biblioteka za rutiranje u Reactu.
- Omogućava navigaciju bez osvježavanja stranice, korišćenje funkcije istorije pretraživača uz očuvanje pravog prikaza aplikacije itd.
- Koristi arhitekturu baziranu na komponentama. Nudi različite komponente za usmjeravanje prema zahtjevima aplikacije.
- Komanda za instaliranje React Routera:
  - `npm i react-router-dom`

Konfigurisanje rutera

- Uvesi ruter koji vam je potreban
  - BrowserRouter za web
  - NativeRouter za mobilne uređaje

```jsx
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";
import reportWebVitals from "./reportWebVitals";
import { BrowserRouter } from "react-router-dom";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

Definisanje ruta

- Ovo se obično radi u okviru App komponente, ali nije pravilo.
- React Route najčešće ima dva parametra:
  - path - string unutar URL address bara pretraživača
  - element - komponenta koja će se pozvati prilikom pogađanja URL-a

```jsx
import "./App.css";
import { Route, Routes } from "react-router-dom";
import { Home } from "./components/Home";
import { About } from "./components/About";
import { Contact } from "./components/Contact";

function App() {
  return (
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/about" element={<About />} />
      <Route path="/contact" element={<Contact />} />
    </Routes>
  );
}

export default App;
```

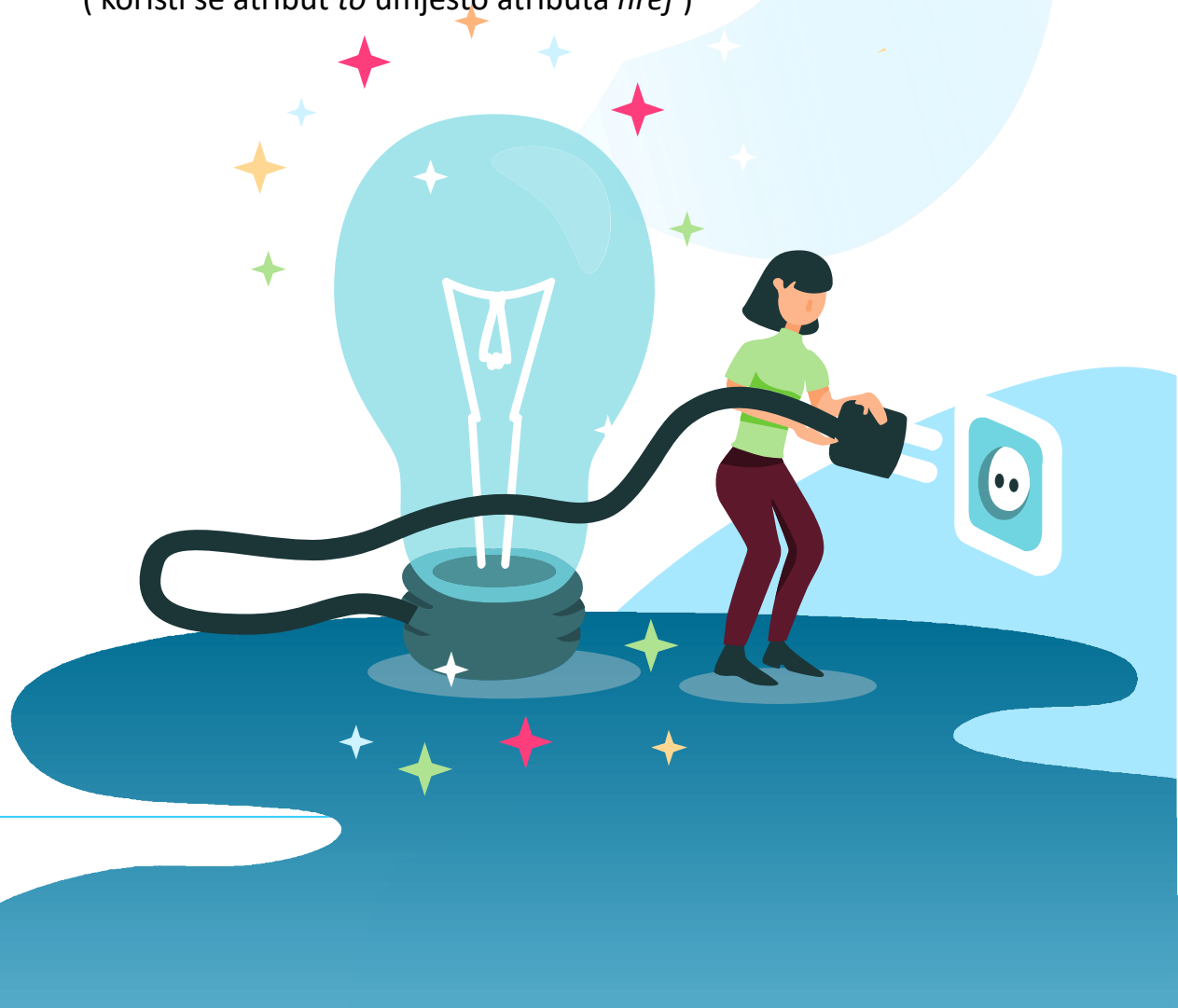
Rukovanje navigacijom

```jsx
import "./App.css";
import { Link, Route, Routes } from "react-router-dom";
import { Home } from "./components/Home";
import { About } from "./components/About";
import { Contact } from "./components/Contact";

function App() {
  return (
    <>
      <nav>
        <ul>
          <li>
            <Link to="/">Home</Link>
          </li>
          <li>
            <Link to="/about">About</Link>
          </li>
          <li>
            <Link to="/contact">Home</Link>
          </li>
        </ul>
      </nav>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/contact" element={<Contact />} />
      </Routes>
    </>
  );
}

export default App;
```

- Korisnici se kreću po aplikaciji klikom na navigacione linkove
- React Router koristi vlastitu prilagođenu Link komponentu za upravljanje navig
  (koristi se atribut *to* umjesto atributa *href*)



Dinamičko rutiranje

- U React Routeru dinamičke rute definišete tako što stavite : ispred onoga što treba da bude dinamički dio rute. Na primjer, /products/1, products/productName ili products/nesto će odgovarati našoj dinamičkoj ruti.

```
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="/about" element={<About />} />
  <Route path="/contact" element={<Contact />} />
  <Route path="/products" element={<ProductList />} />
  <Route path="/products/:id" element={<Product />} />
</Routes>
```

- Tada koristimo *useParams* koji će vratiti proizvod sa identifikatorom koji odgovara dinamičkom parametru u ruti.

```
1  import { useParams } from "react-router-dom";
2
3  export function Product() {
4    const { id } = useParams();
5    return <h1>Product {id}</h1>;
6  }
```

Nested Routes

- Kada su rute dostupne i vidljive u okviru drugih ruta, potrebno je kreirati ih kao *nested routes*
- Sve što treba uraditi je napraviti Route (roditelja) koji ima atribut path postavljen na dijeljenu putanju
- Tada unutar roditeljskog Routa možemo dodati sve ugniježdene rute (njihov atribut path neće uključivati /products)
- Ruta za /products je zamijenjena komponentom Route koja nema atribut path, ali umjesto toga ima atribut index

```
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="/about" element={<About />} />
  <Route path="/contact" element={<Contact />} />
  <Route path="/products">
    <Route index element={<ProductList />} />
    <Route path=":id" element={<Product />} />
    <Route path="new" element={<NewProduct />} />
  </Route>
</Routes>
```

Shared Layouts

- Prikazujemo navigacioni meni i sadržaj izabrane komponente
- Outlet je komponenta koja “čuva mjesto” za prikaz izabrane komponente

```jsx
import { Link, Outlet } from "react-router-dom";

export function ProductLayout() {
  return (
    <>
      <Link to="/products/1">Product 1</Link>
      <br />
      <Link to="/products/2">Product 2</Link>
      <Outlet />
    </>
  );
}
```

```jsx
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="/about" element={<About />} />
  <Route path="/contact" element={<Contact />} />
  <Route path="/products" element={<ProductLayout />}>
    <Route index element={<ProductList />} />
    <Route path=":id" element={<Product />} />
    <Route path="new" element={<NewProduct />} />
  </Route>
</Routes>
```

Outlet Context

- Outlet komponente imaju context atribut
- Proslijeđujemo vrijednost contexa, a zatim u našoj podređenoj komponenti koristimo useOutletContext za pristup toj vrijednosti.

```jsx
import { Link, Outlet } from "react-router-dom";

export function ProductLayout() {
  return (
    <>
      <Link to="/products/1">Product 1</Link>
      <br />
      <Link to="/products/2">Product 2</Link>
      <Outlet  context={{opis: "Neki opis"}}/>
    </>
  );
}
```

```jsx
import { useOutletContext, useParams } from "react-router-dom";

export function Product() {
  const { id } = useParams();
  const obj = useOutletContext();
  return (
    <h1>
      Product {id} {obj.opis}
    </h1>
  );
}
```

useRoutes Hook

```jsx
function App() {
  let element = useRoutes([
    {
      path: "/",
      element: <Home />,
    },
    {
      path: "/products",
      children: [
        { index: true, element: <ProductList /> },
        { path: ":id", element: <Product /> },
      ],
    },
  ]);

  return (
    <>
      <nav>
        <ul>
          <li>
            <Link to="/">Home</Link>
          </li>
          <li>
            <Link to="/products">Products</Link>
          </li>
        </ul>
      </nav>
      {element}
    </>
  );
}
```

Guards, Protected routes

Protected routes

- React router guards predstavljaju interfejse koji govore Router-u da li da dozvoli prelazak na određenu rutu

- Kad i zašto zabranjivati rute:
  - korisnik nije autorizovan da pristupi
  - korisnik nije autentifikovan (potreban je login)
  - podaci potrebni za prikaz stranice još uvek nisu dostupni

- Povratna vrednost guard-a definiše ponašanje rutera:
  - ukoliko je True dozvoljen je prelazak na željenu rutu
  - ukoliko je False zabranjen je prelazak na željenu rutu
  - ukoliko je UrlTree trenutna navigacija se otkazuje i pokreće se nova definisana Url-om

THANK YOU