# Routing

adres URL na naszej stronie - jak i po co React Router

# Co to jest routing (w aplikacjach)

Routing - proces obsługi zmian lokalizacji (ścieżki/adresu) i wynikających stąd zmian w aplikacji.

Routing - definiowanie endpointów (URL-ów) pod którymi aplikacja będzie zdolna wyświetlić określony widok.

Ważne jest to, że routing musi posiadać dwie cechy. Móc rozpoznać oraz zmodyfikować adres, a także móc zdefiniować widok dla danego adresu.

Co to jest Router? Jest to mechanizm/narzędzie/technologia (biblioteka w przypadku React Router), która zajmuje się routingiem.

# Jak zbudowany jest adres URL

https://www.jakasdomena.net/example/id/190?user=pl

`https` - protokół

`www.jakasdomena.net` - nazwa hosta/adres serwera

`example/id/190?user=pl` - ścieżka dostępu (z parametrami)

# Tradycyjne podejście (routing po stronie serwera)

1. Wysłanie żądania do serwera (adres: https://strona/category/29).
2. Serwer identyfikuje URL i przetwarza dane.
3. Serwer renderuje i odsyła plik HTML (i zasoby).
4. Klient przetwarza odpowiedź i wyświetla ją jako DOM w przeglądarce.
5. Zmiana URL - wysyłana jest kolejne żądanie do serwera.
6. Serwer identyfikuje... itd.

# Założenia SPA i routingu w SPA

Jeden plik html dla całej aplikacji niezależnie od URL.

Brak wysyłania żądania do serwera za każdym razem gdy zmienia się widok czy zmienia się URL. Żądanie  do serwera jest wysyłane tylko raz, za pierwszym razem.

Rózne widoki w zależności od zachowania użytkownika/stanu aplikacji/lokalizacji(URL)

SPA oczywiście dopuszcza (i jest to korzystne), że strona ma strukturę opartą na URL-ach i generuje różne widoki. Ważne, że musi to robić bez wysyłania żądania do serwera i odświeżania strony przy każdym nowym URL. SPA potrzebuje więc narzędzia/rozwiązania do routingu.

Ps. UX i wydajność w SPA najczęściej bardzo na plus.

# Czym jest routing w aplikacji React (po stronie klienta)

1. Zapytanie na serwer. Niezależnie od URL aplikacji otrzymujemy jeden plik HTML.
2. Aplikacja jest montowana (tworzy się DOM).
3. Zmiana URL - strona nie jest odświeżana, żądanie do serwera nie jest wysyłane. Zamiast tego aplikacja przechwytuje zmianę URL i w oparciu o konfiguracje routera renderuje nowy widok zdefiniowany dla danego URL.

# Czym jest React Router

React nie ma wbudwanych rozwiązań do routingu.

Biblioteka React Router nie jest oficjalną częścia React, ale jest tak popularna, że może się wydawać, że jest.

React Router dostarcza nam komponenty (w React wszystko jest komponentem), które obsługują nasz routing. Rozpoznają zmiany w URL, zmieniają URL, renderują aplikację przy zmianie URL, wyświetlają różne komponenty w zależności od URL.

React Router, to pełna obsługa routingu po stronie przeglądarki.

Dzięki React Router możemy też wykorzystać interfejs przeglądarki (np. cofanie do poprzednie "strony" (URL)  bez odświeżania)

# Co robi React Router - 3 podstawowe funkcje

- modyfikuje URL
- po wykonananie modyfikacji ponownie renderuje aplikację
- rozpoznaje URL i określa jakie komponenty mają być (a jakie nie) wyświetlane dla danej lokalizacji

# Jak instalujemy

Dostępna poprzez NPM

Wykonujemy komendę w CLI będąc w folderze projektu aplikacji.

```
npm install react-router-dom
```

Mamy też react-router-native (do React Native)

Potem importujemy w modułach, których używamy, np.

```
import {BrowserRouter, Route, NavLink} from 'react-router-dom';
```

# Trzy typy komponentów w React Router

Wszystko jest komponentem - biblioteka React Router dostarcza nam różne komponenty który mogą być użyte w procesie routingu.

- Router - komponent będący rdzeniem routingu w naszej aplikacji. W przypadku aplikacji webowych najczęściej będzie to <BrowserRouter>

- Komponenty dobierające (dopasowujące). Określają kiedy i co wyświetlić. Najczęściej używanym komponentem tego typu jest <Route>

- Komponenty nawigacyjne. Najczęściej używane to <Link> i <NavLink>

# Komponenty - przykłady

Główny komponent (czy też pierwszy Element React) zawiniemy w komponent dostarczony przez React Router

```
<BrowserRouter>
        <App/>
</BrowserRouter>
```

```
return (
    <BrowserRouter>
        <div>...</div>
    </BrowserRouter>
)
```

Route określa co ma się stać i kiedy. Jeśli ścieżka zgadza się z URL, renderowany jest komponent.

```
<Route path="/contact" component={Contact} />
```

Zamiast `<a href="/contact">Kontakt</a>` użyjemy (na przykład):

```
<Link to="/contact">Kontakt</Link>
```

# Przejdżmy do kodu