# 42. Навигация сайта в React

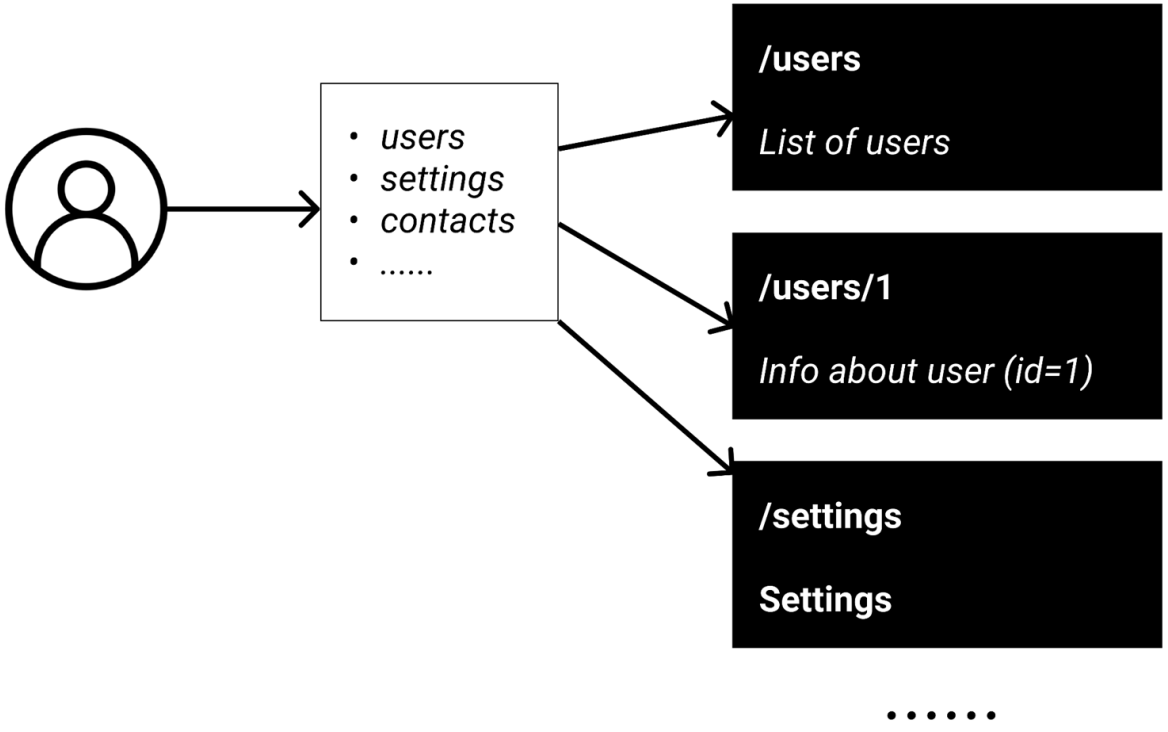
# Цель:

Познакомиться со следующими понятиями и возможностями react:

- маршрутизация сайта
- что такое router

# Маршрутизация:

Маршрутизация необходима, чтобы пользователь видел где он находится в данный момент времени в приложении, а также иметь возможность навигации по истории.

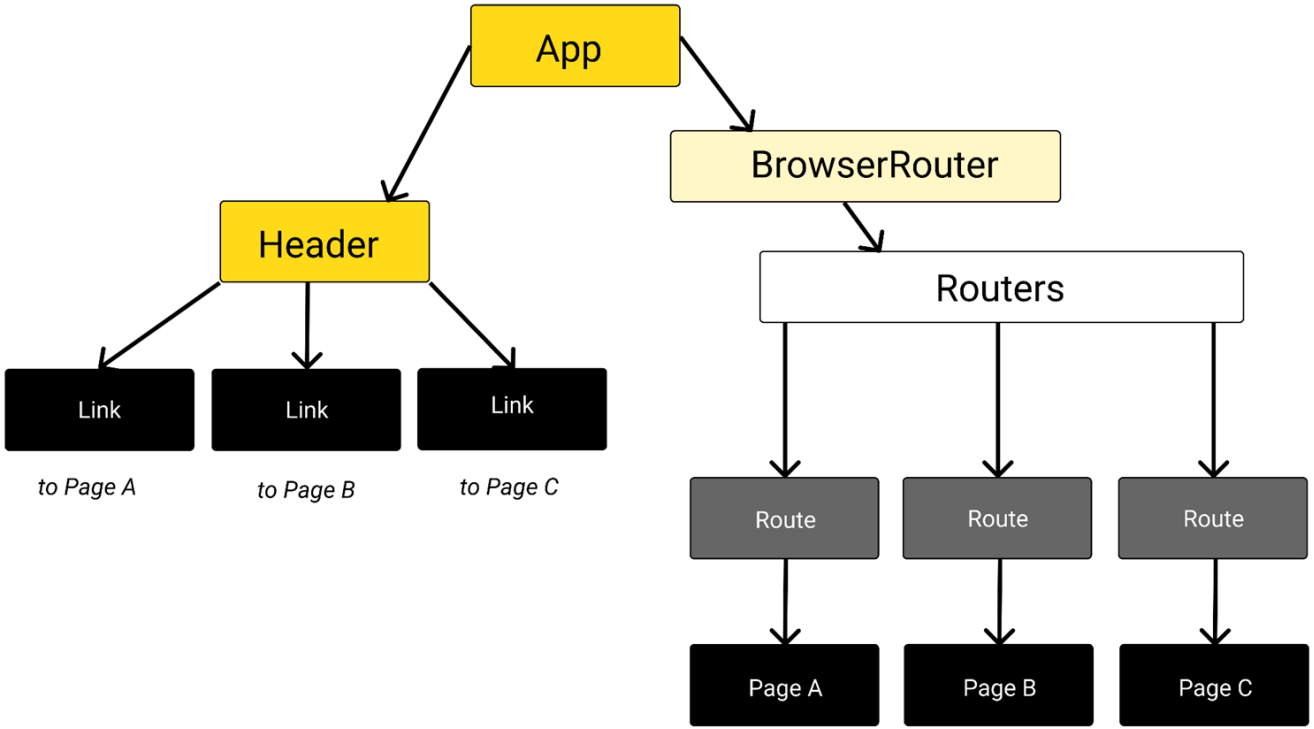


# Маршрутизация:

Так как сам реакт не дает инструментов для маршрутизации, это можно решить за счет дополнительных библиотек.

**react-router-dom**

# react-router-dom:



# Роутер:

**BrowserRouter** определяет набор маршрутов и, когда к приложению, приходит запрос, то Router выполняет сопоставление запроса с маршрутами. И если какой-то из маршрутов совпадает с URL запроса, то этот маршрут выбирается для обработки запроса.

**Routes** - объект, содержащий в себе набор маршрутов. Он позволяет выбрать первый попавшийся правильный маршрут и использовать его для обработки.

**Route** определяет зависимостью между URL и компонентом.

# Конфигурация Routers:

```
import { render } from "react-dom";
import {
  BrowserRouter,
  Routes,
  Route
} from "react-router-dom";
// import your route components too

render(
  <BrowserRouter>
    <Routes>
      <Route path="/" element={<App />}>
        <Route index element={<Home />} />
        <Route path="teams" element={<Teams />}>
          <Route path=":teamId" element={<Team />} />
          <Route path="new" element={<NewTeamForm />} />
          <Route index element={<LeagueStandings />} />
        </Route>
      </Route>
    </Routes>
  </BrowserRouter>,
  document.getElementById("root")
);
```

# Особенности Route:

- **element:** тот компонент, который отвечает за обработку запроса по этому маршруту
- **path:** шаблон адреса, с которым будет сопоставляться запрошенный адрес URL element
- **path="*"**: Путь в виде звездочки - "*" в **path** указывает, что этот маршрут будет сопоставляться со всеми адресами URL, которые не соответствуют предыдущим маршрутам.
- **exact:** Атрибут exact используется в <Route>, чтобы сказать данный <Route> работает только если URL на браузере абсолютно подходит значению атрибута его path.

# Вложенные роуты:

"/invoices"

"/invoices/sent"

"/invoices/:invoiceId"

```
function App() {
  return (
    <Routes>
      <Route path="invoices" element={<Invoices />}>
        <Route path=":invoiceId" element={<Invoice />} />
        <Route path="sent" element={<SentInvoices />} />
      </Route>
    </Routes>
  );
}
```

# Ссылки: Link & NavLink:

**Link и NavLink** применяются для создания ссылок.

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

function Home() {
  return (
    <div>
      <h1>Home</h1>
      <nav>
        <Link to="/">Home</Link> |{" "}
        <Link to="about">About</Link>
      </nav>
    </div>
  );
}
```

# History (useHistory):

**History** - это объект, где хранится наша история навигации по сайту (стек маршрутов), а также методы для удобной работы с историей.

Хук **useHistory** позволяет нам получить доступ к объекту history.

```
import { useHistory } from "react-router-dom";

function HomeButton() {
  let history = useHistory();

  function handleClick() {
    history.push("/home");
  }

  return (
    <button type="button" onClick={handleClick}>
      Go home
    </button>
  );
}
```

# Методы History:

- **goBack** позволяет перенаправить пользователя к предыдущему маршруту в стеке истории. Например, если пользователь перейдет со страницы Home на страницу Students, а затем нажмет кнопку для возврата назад (“Go Back”), он снова будет перенаправлен на страницу Home.
- **push** позволяет добавлять новые записи в стек истории и заставляет пользователя перейти на этот маршрут
- **location** - объект, содержащий в себе информацию о текущем месторасположении. Может включать в себя следующие свойства: pathname, search, hash, state