



Single Page Application

Prof. Victor Farias

V 1.3

Introdução



História

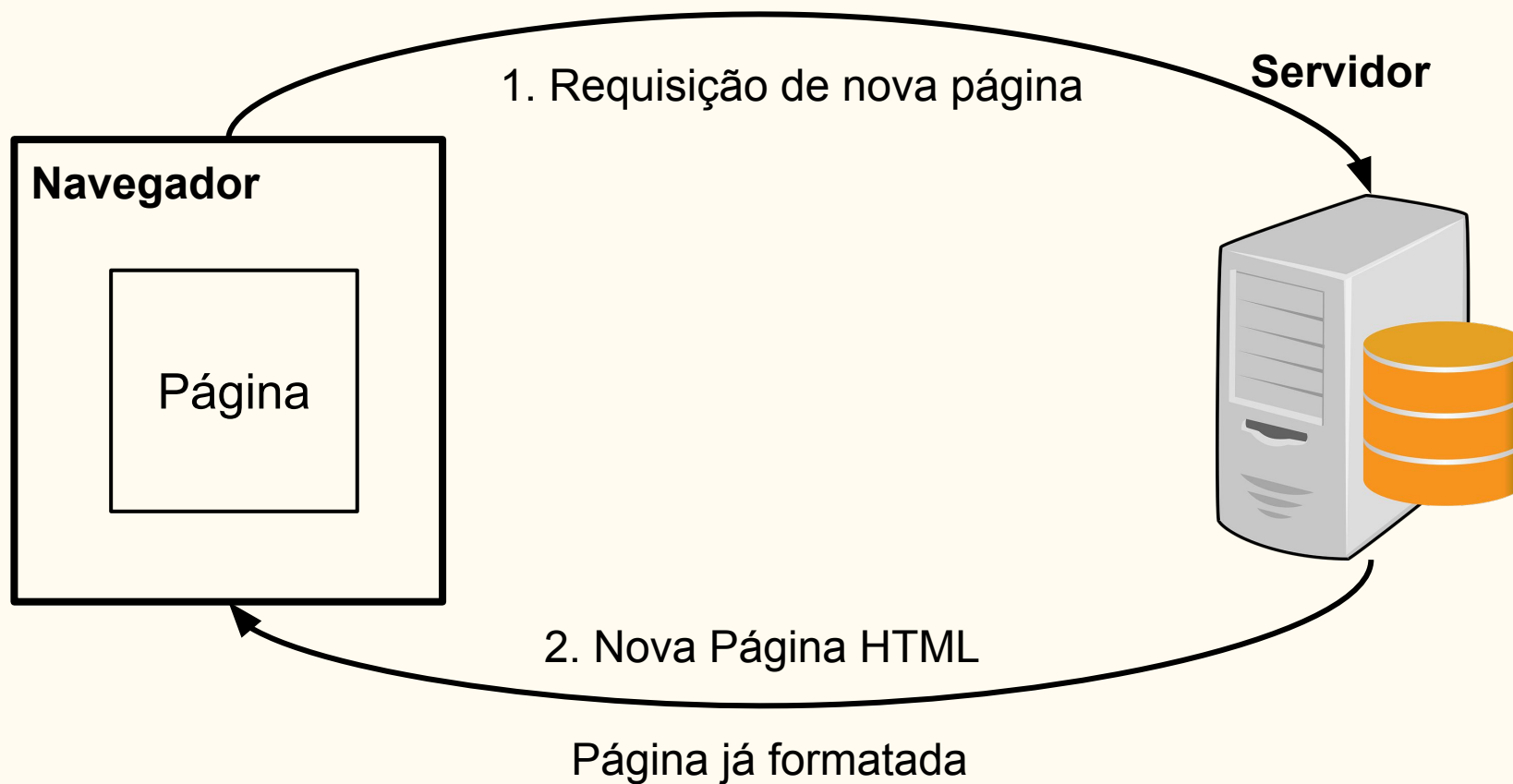
- 1990s
 - JS era usado apenas para auxiliar a criação de elementos de interface
- Era **AJAX** - 1999
 - JavaScript começou a ser usado para fazer requisições HTTP
 - Requisitar dados
 - Validações
- **Engine V8 Google** - 2008
 - Engine capaz de executar JS de forma performática
 - Possibilitou o uso do JS para aplicações reais



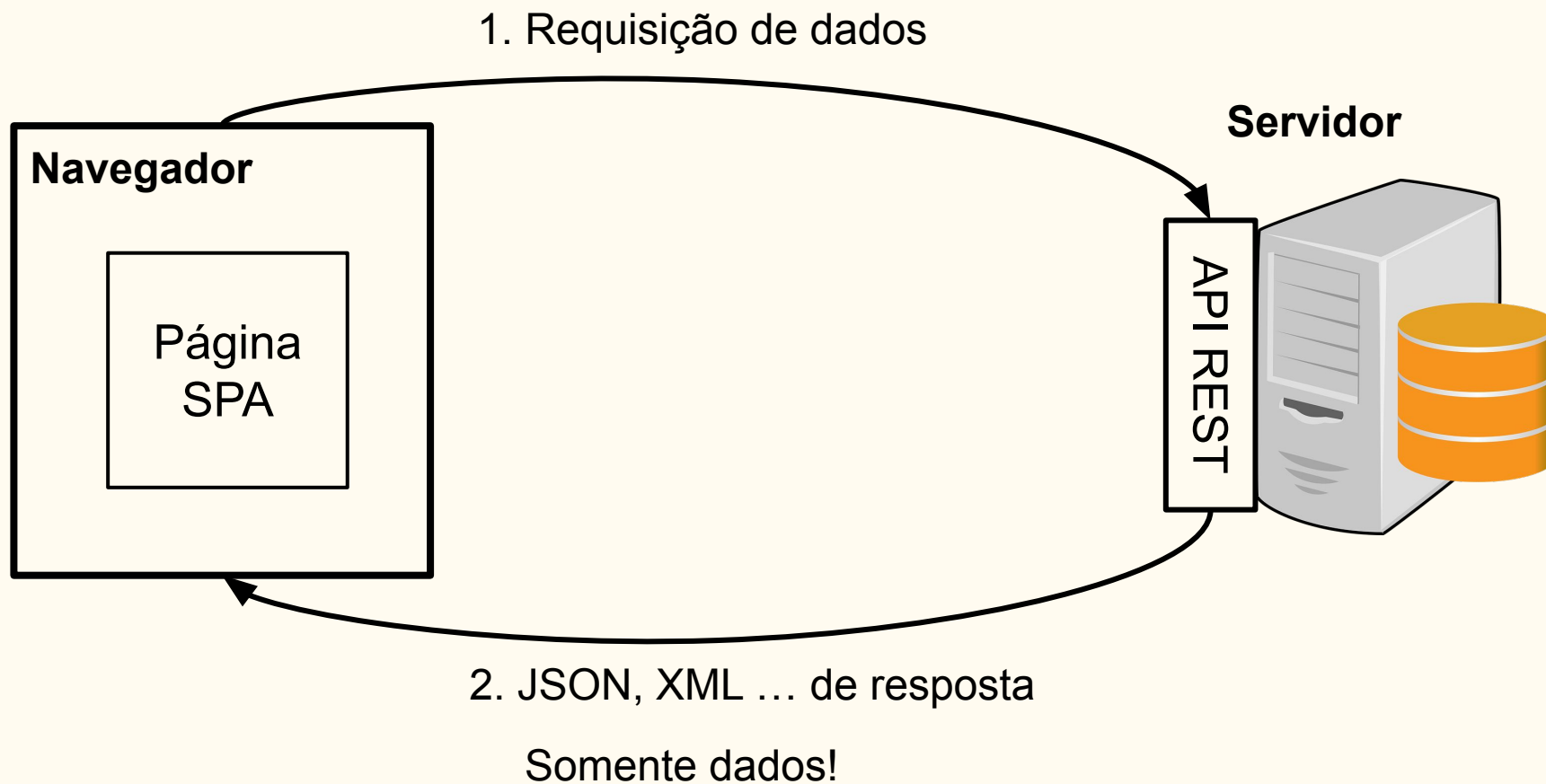
Single Page Application (SPA)

- Aplicações completas no navegador
 - **Página não recarrega!**
 - Componentes são criados dinamicamente no cliente
 - Lógica da aplicação no navegador
 - Todo código é carregado no primeiro acesso
- **Exemplos:**
 - Gmail
 - Facebook
 - Netflix

Web Tradicional



SPA



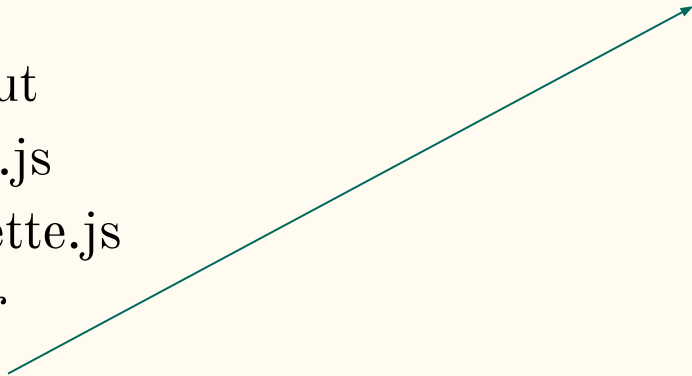
Single Page Application (SPA)

- Fazer SPA em JavaScript puro é problemático
 - Baixa **manutenibilidade**
 - Muito **verboso**
- Miško Hevery - Criador do **Angular**
 - “**80% do código é somente manipulação de DOM e apenas 20% é lógica da aplicação**”
- Assim, surgem os frameworks para SPA

Single Page Application (SPA)

- **Frameworks SPA**

- Backbone.js
- Ember
- Knockout
- Batman.js
- Marionette.js
- Angular
- **React**
- Vue.js



- **React**

- Componentes
- Redux
- **Roteadores!**
- HTTP

Vantagens

- Desenvolvimento WEB com framework se torna mais parecido desenvolvimento **Desktop** ou **Móvel**
 - Facilita manutenção
- Processamento é movido para o **Front**
 - Melhor experiência do usuário
 - Melhor performance
- Desacopla o **Back** do **Front**
 - Agnóstico de back-end
 - Back pode ser programado em qualquer linguagem/framework
 - Back pode, inclusive, ser acessado de outras aplicações

Desvantagens

- Aprendizagem de um novo framework
- Complexidade desnecessária para aplicações pequenas
- Bibliotecas podem ser muito pesadas
 - Problemático para dispositivos com conexão limitada
- **SEO** - Search Engine Optimization
- Fragmentação de regras de negócio
- Primeiro acesso pode ser lento

Roteamento



Roteamento

- Módulo mais importante para SPA!
- Permite atualizar parte das páginas dinamicamente sem carregar a página inteira
- Baseia-se na url para mostrar ou esconder componentes
 - parâmetros da url

<https://www.sistemamatricula.com/alunos/42344/edit>



Parâmetros

Padrão Parâmetros

- **/alunos**
 - Listar alunos
- **/alunos/233543**
 - Exibir detalhes de aluno com id (matrícula) 233543
- **/alunos/233543/edit**
 - Editar dados do aluno com id 233543
- **/alunos/create**
 - Criar novo aluno

/recurso/:id/ação

Instalando pacote

npm install --save react-router-dom

(execute na pasta do projeto)

Criando Roteamento

React Router

- Dois components principais para o roteamento
 - **BrowserRouter:** Tag para habilitar roteamento
 - **Route:** Tag para declarar rotas

React Router

```
<BrowserRouter>
  <Routes>
    <Route path="/" element={<PaginaPrincipal></PaginaPrincipal>}>
    </Route>
    <Route path="/disciplinas"
element={<PaginaDisciplinas></PaginaDisciplinas>}></Route>
  </Routes>
</BrowserRouter>
```

Indica qual url essa rota casa.
Ou seja, qual caminho ativa essa rota

Aqui dentro vem os componentes que
são renderizados quando esse rota está
ativa

App.js

```
...
import { BrowserRouter, Routes, Route } from 'react-router-dom';
...
function App() {
  return (
    <div className="App">
      <BrowserRouter>
        <Routes>
          <Route path="/" element={<PaginaPrincipal></PaginaPrincipal>}>
          </Route>
          <Route path="/disciplinas" element={<PaginaDisciplinas></PaginaDisciplinas>}></Route>
        </Routes>
      </BrowserRouter>
    </div>
  );
}
```

Navegando pelas rotas



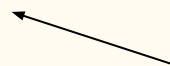
Navegando pelas rotas

- Componente Link permite a navegação entre as rotas sem recarregar a página

```
import { Link } from 'react-router-dom';
```

```
...
```

```
<Link to="/pagina2">Link 2</Link>
```



Indica qual rota será redirecionado

Renderiza um `<a>` no html

Estilizando Links ativos



NavLink

- Componente NavLink

```
import { NavLink } from 'react-router-dom';  
...  
<NavLink to="/pagina2">Link 2</NavLink>
```

Quando rota está ativa, tag `<a>` recebe classe “active”

Recebendo Parâmetros via URL

Parâmetros URL

Parâmetros podem ser passados pela URL.

- **/alunos/233543**
 - Exibir detalhes de aluno com id (matrícula) 233543
- **/alunos/233543/edit**
 - Editar dados do aluno com id 233543

/recurso/:id/ação

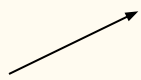
Parâmetros URL

Como receber parâmetros em um componente?


- 1) Preparar rota para receber parâmetro. É necessário usar outra sintaxe.

```
<Route path="/alunos/:id" element={<PaginaAluno></PaginaAluno>}></Route>
```

Parâmetro a ser recebido

An arrow points from the text 'Parâmetro a ser recebido' to the ':id' part of the path string in the code above.

Componente a ser renderizado

An arrow points from the text 'Componente a ser renderizado' to the element prop value in the code above.

Parâmetros URL

2) Acessar parâmetro via useParams()

```
import { useParams } from "react-router-dom";  
...  
export function PaginaAluno(){  
  let params = useParams();  
  return(  
    <div>  
      <h2>Página detalhe aluno com id {params.id}</h2>  
    </div>  
  )  
}
```

Objeto que contém os parâmetros passados pela url

Redirecionando para Rotas com Parâmetros



Redirecionamento com Parâmetros

- Basta concatenar o id no final da rota

```
import { Link } from 'react-router-dom';  
...  
<Link to={"/alunos/" + id}>Link 2</Link>
```

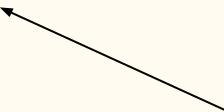
Aqui há uma variável **id** contendo um inteiro representando o identificador do aluno

Ativando somente um rota

—

Switch

```
// /src/app.js
import { BrowserRouter, Route, Switch } from 'react-router-dom';
...
<BrowserRouter>
  <Switch>
    <Route exact path="/alunos" component={ListarAlunos}></Route>
    <Route path="/alunos/create" component={CadastroAluno}></Route>
    <Route path="/alunos/:matricula" component={Alunos}></Route>
    <Route path="*">
      <h1>Erro 404</h1>
    </Route>
  </Switch>
</BrowserRouter>
```



Se não casa com nenhum, cai na última rota

Trocando rotas
programaticamente

Objeto history

```
export function Cabecalho({pagina}){  
  let navigate = useNavigate();  
  return (  
    <header>  
      <span onClick={()=>{navigate("/")}}>  
        SigaaPIW  
      </span>  
    </header>  
  )  
}
```

Função para trocar de página programaticamente

Arrow function para navegar para página

Perguntas?

Prof. Victor Farias