PÓS-GRADUAÇÃO ALFA

UNIALFA
CENTRO UNIVERSITÁRIO
ALVES FARIA

Diego Guedes

PROGRAMAÇÃO COM FRAMEWORKS E COMPONENTES

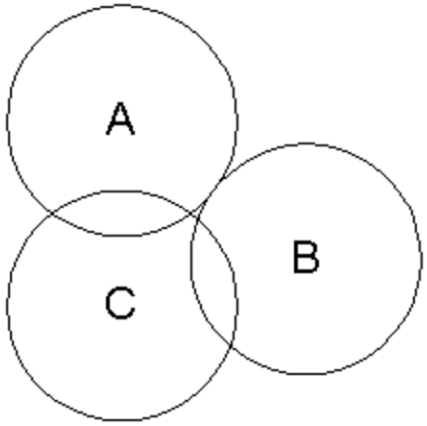
GRUPO JOSÉ ALVES

Apresentação

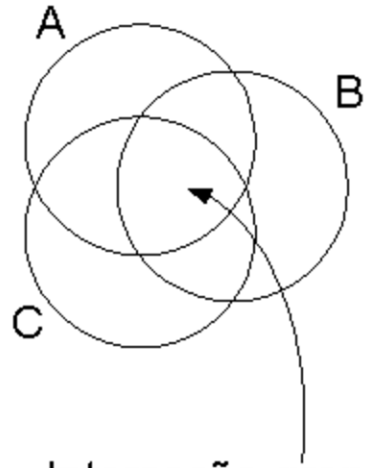
- Bacharel em Ciências da Computação, UFG (2010)
- Mestrado em Ciências da Computação, UFG (2013)
- Campeão das regionais da Maratona de Programação em 2008 e 2009
- Trabalhei em projetos regionais, nacionais e mundiais de pesquisa na área de Ciências da Computação
- Fui professor substituto na UFG por 2 anos (2013-2014)
- Fui professor Adjunto na Faculdade Senac por 2 anos (2015-2016)
- P&D goGeo (2014)
- Analista de Sistemas na Saneago (Desde 2014)

Framework

- Um framework captura a funcionalidade comum a várias aplicações
- As aplicações devem ter algo razoavelmente grande em comum: pertencem a um mesmo domínio de problema



Impossível criar Framework



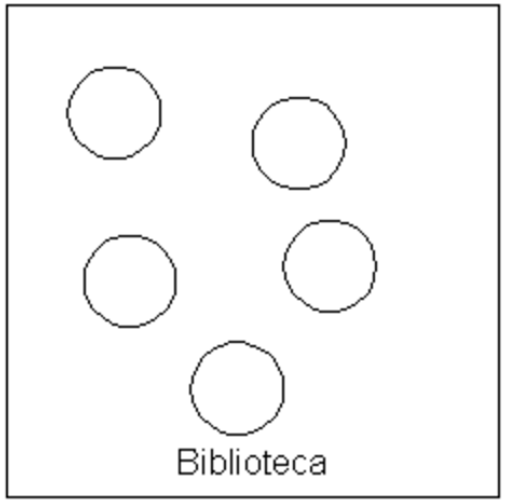
Interseção grande
Possível criar Framework

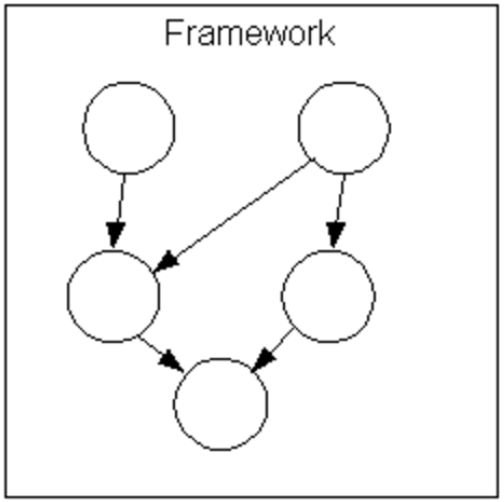
API x Biblioteca x Framework

- API (***Application Programming Interface***)
  - É um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços. Ex. API do Google Maps
- Biblioteca
  - A biblioteca normalmente **é uma implementação real das regras** de uma API. Portanto ela é mais concreta.
- Framework
  - Um *framework* normalmente é um **conjunto de bibliotecas para conseguir executar uma operação maior**

Diferenças entre um Framework e uma Biblioteca de Classes OO

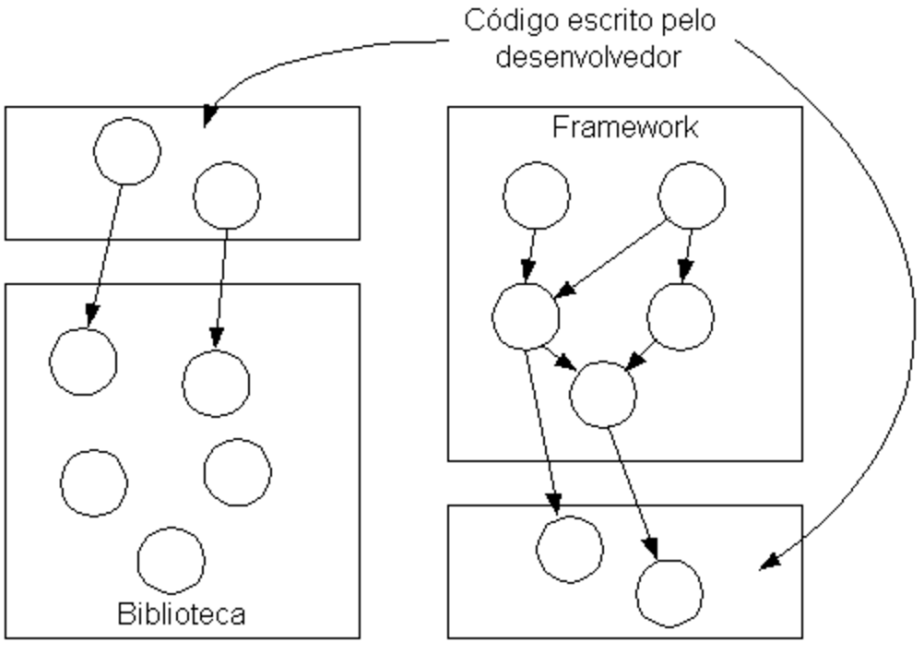
- *Numa biblioteca de classes, cada classe é única e independente das outras*
- *Num framework, as dependências/colaborações estão embutidas (wired-in interconnections*)
- Com biblioteca, as aplicações criam as colaborações



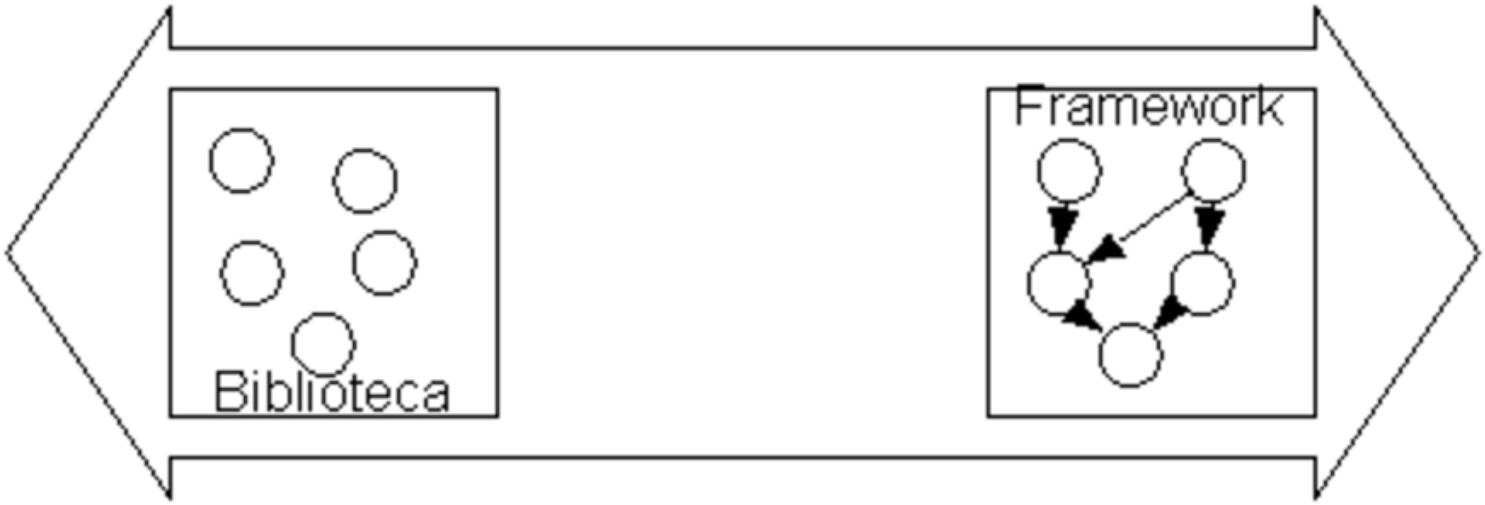


Diferenças entre um Framework e uma Biblioteca de Classes OO

- O framework é usado de acordo com o Hollywood Principle ("*Don't call us, we'll call you*")
  - É o framework que chama o código da aplicação (que trata das particularidades dessa aplicação)
  - Framework = *Upside-down library* (Biblioteca de cabeça para baixo)



Diferenças entre um Framework e uma Biblioteca de Classes OO



- Classes instanciadas pelo cliente
- Cliente chama funções
- Não tem fluxo de controle predefinido
- Não tem interação predefinida
- Não tem comportamento default

- Customização com subclasse ou composição
- Chama funções da "aplicação"
- Controla o fluxo de execução
- Define interação entre objetos
- Provê comportamento default

Características Básicas de Frameworks

- Um framework deve ser reusável
  - É o propósito final!
  - Para ser reusável, deve primeiro ser *usável*
    - Bem documentado
    - Fácil de usar
- Deve ser extensível
  - O framework contém funcionalidade abstrata (sem implementação) que deve ser completada
- Deve ser de uso seguro
  - O desenvolvedor de aplicações não pode destruir o framework
- Deve ser eficiente
  - Devido a seu uso em muitas situações, algumas das quais poderão necessitar de eficiência
- Deve ser completo
  - Para endereçar o domínio do problema pretendido

Framework Baseado em Componentes

Prof. Me. Diego Guedes

Framework baseado em componentes

- São *frameworks* baseados em componentes de software.
- A extensão da arquitetura é feita a partir de interfaces definidas para componentes.
- Os recursos existentes são reutilizados e estendidos por meio de: definição de um componente adequado a uma interface específica e integração de componentes [Gamma 94]
- Exemplos:
  - Framework ZK
  - PrimeFaces
  - AngularJS
  - React?

React

- React é uma biblioteca JavaScript para construção de interfaces de usuário
- React faz com que a criação de UIs interativas seja uma tarefa fácil.
- O React trabalha com componentes encapsulados que gerenciam seu próprio estado e então, combina-os para criar UIs complexas.
- Como a lógica do componente é escrita em JavaScript e não em templates, pode-se facilmente passar diversos tipos de dados ao longo da sua aplicação e ainda manter o estado fora do DOM.
- Há “framework” dentro do react (Redux, por exemplo)

React

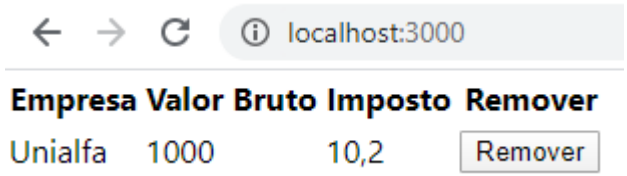
- npm install create-react-app
  - uma ferramenta do Facebook que gerará automaticamente um projeto inicial do React
- npx create-react-app react-unialfa
- cd react-unialfa
- npm start

React – Criando a primeira tabela

```
import React from 'react';

function App() {
  return (
    <div className="App">
      <table>
        <thead>
          <tr>
            <th>Empresa</th>
            <th>Valor Bruto</th>
            <th>Imposto</th>
            <th>Remover</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>Unialfa</td>
            <td>1000</td>
            <td>10,2</td>
            <td><button>Remover</button></td>
          </tr>
        </tbody>
      </table>
    </div>
  );
}

export default App;
```



React – Primeiro Componente

- Todo class componente deve ter o método *render()*
- Vamos criar o componente Tabela no arquivo Tabela.js

React – Primeiro Componente

```
import React, { Component } from 'react';

class Tabela extends Component {
    render() {
        return (
            <table>
                <thead>
                    <tr>
                        <th>Empresa</th>
                        <th>Valor Bruto</th>
                        <th>Imposto</th>
                        <th>Remover</th>
                    </tr>
                </thead>
                <tbody>
                    <tr>
                        <td>Unialfa</td>
                        <td>1000</td>
                        <td>10,2</td>
                        <td><button>Remover</button></td>
                    </tr>
                </tbody>
            </table>
        );
    }
}
export default Tabela;
```

React – Primeiro Componente

- App.js fica somente assim:

```
import React from 'react';
import './App.css';
import Tabela from './Tabela';

function App() {
  return (
    <div className="App">
      <Tabela />
    </div>
  );
}

export default App;
```

React – Primeiro Componente

- Criar o *Header* e *Body* da tabela fora da classe por meio de um *arrow function*

React – Primeiro Componente

```
import React, { Component } from 'react';

const TableHead = () => {
    return (
        <thead>
            <tr>
                <th>Empresa</th>
                <th>Valor Bruto</th>
                <th>Imposto</th>
                <th>Remover</th>
            </tr>
        </thead>
    );
}
const TableBody = () => {
    return (
        <tbody>
            <tr>
                <td>Unialfa</td>
                <td>1000</td>
                <td>10,2</td>
                <td><button>Remover</button></td>
            </tr>
        </tbody>
    );
}
class Tabela extends Component {
    render() {
        return (
            <table>
                < TableHead />
                < TableBody />
            </table>
        );
    }
}
export default Tabela;
```

React – Props

- Passa para uma informação de um componente para outro componente, em React, utilizamos o Props que têm uma notação semelhante às propriedades do HTML
- No exemplo ao lado, a Props é notasfiscais

```
<div className="App">
   <Tabela notasfiscais = { nf }/>
</div>
```

- Ademais, em App.js, criaremos um array de json

React – Props

```
import React from 'react';
import Tabela from './Tabela';
function App() {
  const nf = [
    {empresa: 'Unialfa',
    valorBruto: 1000,
    imposto: 10.2}
  ];


  return (
    <div className="App">
      <Tabela notasFiscais={nf}/>
    </div>
  );
}

export default App;
```

React – Props

- Faremos uso das Props em Tabela.js

```
import React, { Component } from 'react';

const TableHead = () => {
    return (
        <thead>
            <tr>
                <th>Empresa</th>
                <th>Valor Bruto</th>
                <th>Imposto</th>
                <th>Remover</th>
            </tr>
        </thead>
    );
}
const TableBody = props => {
    const linhas = props.nf.map((linha, index) => {
        return (
            <tr key={index}>
                <td>{linha.empresa}</td>
                <td>{linha.valorBruto}</td>
                <td>{linha.imposto}</td>
                <td><button>Remover</button></td>
            </tr>

        )

    });
    return (
        <tbody>
            {linhas}
        </tbody>
    );
}
class Tabela extends Component {
    render() {
        const { notasFiscais } = this.props;
        return (
            <table>
                < TableHead />
                < TableBody nf={notasFiscais} />
            </table>
        );
    }
}
export default Tabela;
```

React – State

- *state* é o estado da aplicação
- Vamos passar App.js para uma classe Component

```
import React, { Component } from 'react';
import Tabela from './Tabela';

class App extends Component {
  state = {
    nf: [
      {
        empresa: 'Unialfa',
        valorBruto: 1000,
        imposto: 10.2
      }
    ]
  };
  render() {
    return (
      <div className="App">
        <Tabela notasFiscais={this.state.nf} />
      </div>
    );
  }
}
export default App;
```

React – Removendo NF

- Precisamos implementar a funcionalidade de remover uma nota
  - Para isso, devemos mudar o state
- Como não é possível alterar um *state* diretamente, teremos que utilizar um método específico do React para efetuar essa alteração, chamado setState()
- Criaremos um método que remove as notas fiscais em App.js

```
removeNotaFiscal = index => {
    const { nf } = this.state;

    this.setState(
      {
        nf: nf.filter((nota, posAtual) => {
          return posAtual!==index;
        }),
      }
    );
  }
```

React – Removendo NF

- Como precisamos acessar esse método em Tabela.js , passaremos por meio da adição de um prop removeNF={this.removeNotaFiscal} no método render()
- Em Tabela.js , recebemos o método removeNotaFiscal (por meio da prop removeNF) e o passaremos também como prop, para TableBody

React – Removendo NF

```
import React, { Component } from 'react';
const TableHead = () => {
    return (
        <thead>
            <tr>
                <th>Empresa</th>
                <th>Valor Bruto</th>
                <th>Imposto</th>
                <th>Remover</th>
            </tr>
        </thead>
    );
}
const TableBody = props => {
    const linhas = props.nf.map((linha, index) => {
        return (
            <tr key={{index}>
                <td>{linha.empresa}</td>
                <td>{linha.valorBruto}</td>
                <td>{linha.imposto}</td>
                <td><button onClick={() => { props.removeNotaFiscal(index) }}>Remover</button></td>
            </tr>
        )
    });
    return (
        <tbody>
            {linhas}
        </tbody>
    );
}
class Tabela extends Component {
    render() {
        const { notasFiscais, removeNF } = this.props;
        return (
            <table>
                < TableHead />
                < TableBody nf={notasFiscais} removeNotaFiscal={removeNF} />
            </table>
        );
    }
}
export default Tabela;
```

React – Criando Formulário

- Vamos criar um Formulario.js

```
import React, { Component } from 'react'

class Formulario extends Component {
    constructor(props){
        super(props);
        this.stateInicial = {
            empresa:'',
            valorBruto:'',
            imposto:'',
        }
        this.state = this.stateInicial;
    }
    render() {
        const {empresa, valorBruto, imposto} = this.state;
        return (
            <form>
                <label for="empresa">Empresa</label>
                <input
                    id="empresa"
                    type="text"
                    name="empresa"
                    value={empresa}
                />
                <label for="valorBruto">Valor Bruto</label>
                <input
                    id="valorBruto"
                    type="text"
                    name="valorBruto"
                    value={valorBruto}
                />
                <label for="imposto">Imposto</label>
                <input
                    id="imposto"
                    type="text"
                    name="imposto"
                    value={imposto}
                />
                <button type="button">Salvar
                </button>
            </form>
        )
    }
}
export default Formulario;
```

React – Criando Formulário

- Vamos incluir o Formulario no App.js
- Note que ainda não é possível preencher o formulário

```
render() {
    return (
      <div className="App">
        <Fragment>
          <Tabela notasFiscais={this.state.nf} removeNF={this.removeNotaFiscal} />
          <Form />
        </Fragment>
      </div>
    );
  }
```

React – Criando Formulário

- Vamos criar um “Escutador de eventos”

```
inputListener = event =>{
        const { name, value } = event.target;
        this.setState({
            [name] : value
        });
    }
```

React – Criando Formulário

```
import React, { Component } from 'react'

class Formulario extends Component {
    constructor(props){
        super(props);
        this.stateInicial = {
            empresa:'',
            valorBruto:'',
            imposto:'',
        }
        this.state = this.stateInicial;
    }
    inputListener = event =>{
        const { name, value } = event.target;
         this.setState({
            [name] : value
        });
    }
    render() {
        const {empresa, valorBruto, imposto} = this.state;
        return (
            <form>
                <label for="empresa">Empresa</label>
                <input
                    id="empresa"
                    type="text"
                    name="empresa"
                    value={empresa}
                    onChange={this.inputListener}
                />
                <label for="valorBruto">Valor Bruto</label>
                <input
                    id="valorBruto"
                    type="text"
                    name="valorBruto"
                    value={valorBruto}
                    onChange={this.inputListener}
                />
                <label for="imposto">Imposto</label>
                <input
                    id="imposto"
                    type="text"
                    name="imposto"
                    value={imposto}
                    onChange={this.inputListener}
                />
                <button type="button">Salvar
                </button>
            </form>
        )
    }
}
export default Formulario;
```

React – Estilizando a Aplicação

- Vamos estilizar nosso projeto com o materialize (há outros como MaterialUI e PrimeReact)
  - npm install materialize-css
- Em App. js, ao invés de importar com o App.css, vamos usar o materialize-css/dist/css/materialize.min.css

React – Estilizando a Aplicação

- Estilizando o botão da Tabela.js
  - className="btn waves-effect waves-light red darken-4"
- Colocando os dados do formulário na mesma linha

React – Estilizando a Aplicação

```
<form>
        <div className="row">
          <div className="input-field col s4">
            <label className="input-field" htmlFor="empresa">Empresa</label>
            <input
              id="empresa"
              type="text"
              name="empresa"
              value={empresa}
              onChange={this.inputListener} />
          </div>
          <div className="input-field col s4">
            <label className="input-field" htmlFor="valorBruto">Valor Bruto</label>
            <input
              id="valorBruto"
              type="text"
              name="valorBruto"
              value={valorBruto}
              onChange={this.inputListener} />

          </div>
          <div className="input-field col s4">
            <label className="input-field col s4" htmlFor="imposto">Imposto</label>
            <input
              id="imposto"
              type="text"
              name="imposto"
              value={imposto}
              onChange={this.inputListener} />
          </div>
        </div>
        <button type="button" className="btn waves-effect waves-light red darken-4">Salvar
        </button>
      </form>
```

React – Estilizando a Aplicação

- Por fim, em App.js, vamos colocar nossa aplicação no class container e colocar um cabeçalho

```
<Fragment>
        <div className="container">
          <h1>Programação com Frameworks - Unialfa</h1>
          <Tabela notasFiscais={this.state.notasFiscais}
removeNotaFiscal={this.removeNotaFiscal} />
          <Formulario escutadorDeSubmit={this.escutadorDeSubmit} />
        </div>
      </Fragment>
```

React – Notificações na Aplicação

- Vamos criar um componente para notificações PopUp.js

```
import M from 'materialize-css';


const PopUp = {
  exibeMensagem: (status, msg) => {
    if (status === "success")
      M.toast({ html: msg, classes: "green", displayLength: 2000 })


    if (status === "error")
      M.toast({ html: msg, classes: "red", displayLength: 2000 })


  }
}
export default PopUp;
```

React – Consumindo de uma API

- Vamos agora consumir dados de um API
- Teremos que mudar a aplicação de NotaFiscal da aula anterior
- O código está no github
- Além disso, temos que mudar o nome da coluna “valor” para “valor_bruto” do banco de dados notafiscal
- Iremos utilizar o fetch(), uma funcionalidade do ECMAScript 6, para realizarmos requisições.
- Nossa API terá 3 serviços
  - ListaNotasFiscais
  - CriaNotaFiscal
  - RemoveNotaFiscal

React – Consumindo de uma API (ApiService.js)

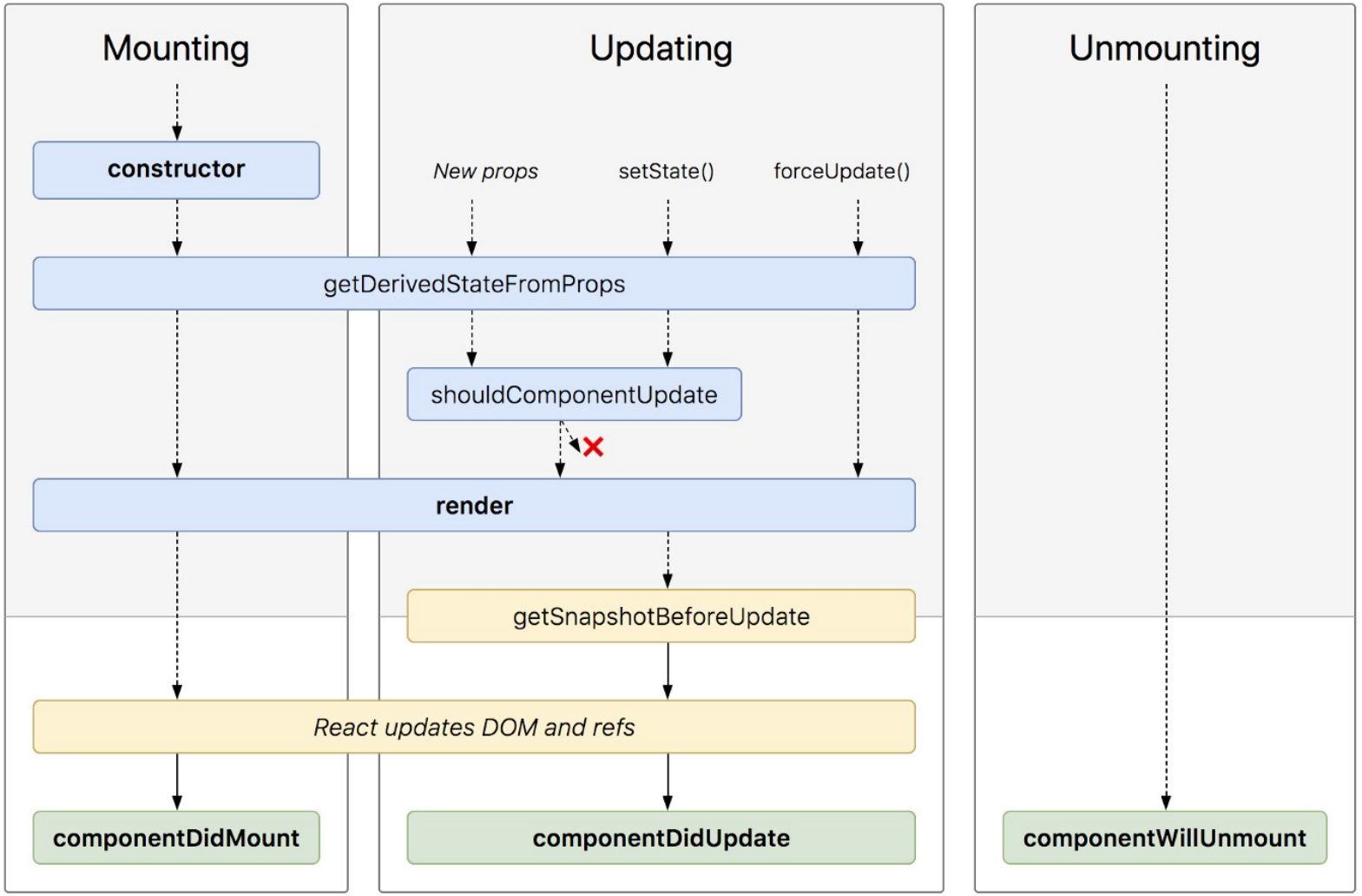
```
const ApiService = {
    ListaNotasFiscais: () => {
        return fetch('http://localhost:8080/api/notafiscal')
            .then(resposta => resposta.json())
    },
    CriaNotaFiscal : notaFiscal => {
        return fetch('http://localhost:8080/api/notafiscal',
{method: 'POST', headers: {'content-type': 'application/json'}, body: notaFiscal})
        .then(res => res.json());
    },RemoveNotaFiscal: id => {
        return fetch(`http://localhost:8080/api/notafiscal/${id}`,
{method: 'DELETE', headers: { 'content-type' : 'application/json'},})
        .then(res => res.json());;
    }
}
export default ApiService;
```

React – Consumindo de uma API

- Como agora iremos consumir de uma API, o construtor de App.js começa vazio, ou seja, o state começa vazio

```
constructor(props) {
    super(props);
    this.state = {
      nf: [],
    };
  }
```

React – Consumindo de uma API



React – Consumindo de uma API

- Vamos criar o método componentDidMount em App.js

```
componentDidMount() {
    ApiService.ListaNotasFiscais()
      .then(res => {
        this.setState({ nf: [...this.state.nf, ...res] })

      })
  }
```

- Temos que também modificar a arrow function de removeNotaFiscal de App.js
  - Não esquecer de mudar index em Tabela.js

```
removeNotaFiscal = id => {
    const { nf } = this.state;
    this.setState(
      {
        nf: nf.filter(nota => {
          return nota.id !== id;
        }),
      }
    );
    PopUp.exibeMensagem("error", "Nota Fiscal removida com sucesso");
    ApiService.RemoveNotaFiscal(id);
  }
```

React – Consumindo de uma API

- Falta criar as notas fiscais
- Vamos utilizar um método em App.js um método escutadorDeSubmit

```
escutadorDeSubmit = notaFiscal => {
    ApiService.CriaNotaFiscal(JSON.stringify(notaFiscal))
      .then(notaFiscal)
      .then(notaFiscal => {
        this.setState({ nf: notaFiscal });
        PopUp.exibeMensagem("success", "Nota Fiscal adicionado com sucesso");
      })

  }
```

- Em Formulario.js, iremos criar uma ação para o clique do botão salvar

```
onClick={this.submitFormulario}
```

- SubmitFormulario em Formulario.js fica assim

```
submitFormulario = () => {
        this.props.escutadorDeSubmit(this.state);
        this.setState(this.stateInicial);

      }
```

- Por fim, colocar na propriedade do Formulario em App.js

```
<Form escutadorDeSubmit={this.escutadorDeSubmit} />
```

React – Aplicação Final

GRUPO JOSÉ ALVES

Programação com Frameworks - Unialfa

Empresa	Valor Bruto	Imposto	Remover
Saneago	1560	230	REMOVER
Unialfa	3125	1200	REMOVER

Empresa | Valor Bruto | Imposto

SALVAR

UNIALFA
CENTRO UNIVERSITÁRIO
ALVES FARIA

- Obrigado!
- Dúvidas?

Diego Américo Guedes
www.facebook.com/professordiegoguedes
diegoamericoguedes@gmail.com