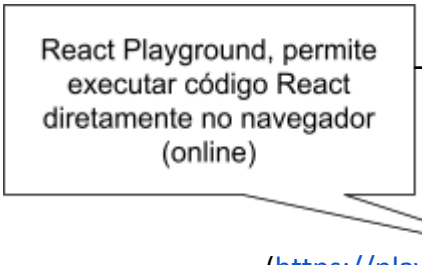
INSTITUTO FEDERAL
Paraná
Campus Paranaguá

Ministério da Educação

ENSINO MÉDIO INTEGRADO - INFORMÁTICA
Disciplina de Desenvolvimento Web
Aula 12: React JS - Ciclo de Vida / Componentes (props e state)



Gil Eduardo de Andrade

Conceitos Preliminares

(https://react.dev/)

(https://playcode.io/react) (https://codesandbox.io/p/sandbox/react-new)

OBS.: *para criação de aplicações com React JS é necessário ter instalado o Node JS, como visto anteriormente na disciplina.*

React JS

O React JS é uma biblioteca JavaScript de código aberto, criada pelo Facebook (agora Meta), utilizada para o desenvolvimento de Interfaces de Usuário (UIs) voltadas a aplicações web. O React JS tem por objetivo facilitar a criação de UIs complexas e interativas, tornando o processo de desenvolvimento mais eficiente e escalável.

Por que utilizar React JS?

- Componentes: utiliza um paradigma de componentes, onde a UI é dividida em partes menores e reutilizáveis, facilitando a organização e manutenção do código, além de dar suporte a criação de interfaces mais consistentes.

- JSX: introduziu o JSX, uma extensão da sintaxe JavaScript que permite escrever HTML diretamente dentro do JavaScript, tornando o código mais intuitivo e fácil de ler.

- Virtual DOM: utiliza uma representação virtual do DOM (*Document Object Model*), que otimiza as atualizações da interface. Ao invés de atualizar o DOM real a cada mudança, o React compara a versão virtual com a real e aplica apenas às alterações necessárias, resultando em um desempenho superior.

- Declarativo: permite que você descreva como a UI deve ser, com o framework se encarregando de atualizar a interface de acordo com as mudanças nos dados, tornando o código mais fácil de entender e depurar.

- Grande comunidade: possui uma comunidade ativa e em constante crescimento, disponibilizando diversos recursos, bibliotecas e ferramentas para auxiliar no desenvolvimento.

Como o React JS funciona?

- Criação de componentes: permite definir os componentes que representam as diferentes partes da interface que está sendo criada.

- Renderização: renderiza (apresenta) os componentes, criando uma representação virtual da UI.

- Atualizações: quando os dados mudam, o React recalcula a representação virtual e identifica as diferenças em relação à versão anterior.

- Atualização do DOM: aplica as mudanças mínimas necessárias no DOM real para refletir as alterações na representação virtual.

Vantagens do React JS

- Desempenho: o Virtual DOM e a otimização das atualizações garantem um bom desempenho, mesmo em aplicações complexas.

- Reutilização de código: os componentes podem ser reutilizados em diferentes partes da aplicação, reduzindo a quantidade de código e facilitando a manutenção.

- Facilidade de aprendizado: a sintaxe do JSX e a abordagem declarativa tornam o React relativamente fácil de aprender, mesmo para desenvolvedores iniciantes.

- Grande ecossistema: possui um ecossistema rico em bibliotecas e ferramentas, como Redux para gerenciamento de estado, React Router para roteamento, entre outros.

Instalando Comando: *create-react-app*
(no terminal de comando)

```
None
npm i -g create-react-app
```

Criando Aplicação
(no terminal de comando)

```
None
npx create-react-app project-name
```

Executando Aplicação
(no terminal de comando)

```
None
npm run start
```

Preparando Aplicação
(no terminal de comando)

```
None
npm run build
```

OBS.: *o comando "**npm run build**" prepara a aplicação para ser colocada em produção. Em resumo, otimiza e compacta todo o código React, transformando-o num conjunto de arquivos estáticos, prontos para serem hospedados em um servidor web.*

Criando Primeiro Projeto
(no terminal de comando)

```
None
npx create-react-app start-project
```

Arquivo: package.json

```
JavaScript
{
  "name": "start-project",  // Nome do projeto
  "version": "0.1.0",       // Versão do projeto
  "private": true,          // Não publicado em repos públicos
  "dependencies": {         // Lista de dependências do projeto
    "cra-template": "1.2.0",     // Template inicial projeto
    "react": "^19.0.0",          // Biblio principal do React
    "react-dom": "^19.0.0",      // Biblio renderiza componente
```

```
    "react-scripts": "5.0.1"     // Conjunto de scripts
  },
  "scripts": {  // Scripts que podem ser executado com npm run
    "start": "react-scripts start", // Inicia servidor desenv.
    "build": "react-scripts build", // Cria versão produção
    "test": "react-scripts test",   // Executa testes
    "eject": "react-scripts eject"  // Sai config. padrão
  },
...
```

OBS.: caso seja necessária instalação do pacote ***web-vitals***:

```
None
                    npm install web-vital
```

Arquivo: src/index.js

```
JavaScript
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

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

Arquivo: ./src/App.js

JavaScript

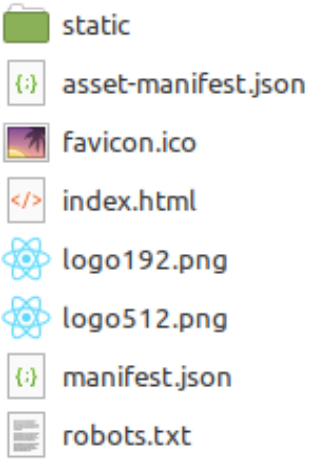
```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

Executando Aplicação
(no terminal de comando - dentro da pasta do projeto)

```
npm run start
```

Preparando Aplicação
(no terminal de comando - dentro da pasta do projeto)

```
None
npm run build
```



Componentes: Classe e Funcional
No React existem duas formas principais de criação de componentes: classe e funcional. Apesar de ambos servirem para construir a interface do usuário, eles possuem características e abordagens distintas.

Componentes de Classe
São baseados em classes (ES6), sendo criados através da extensão da classe React.Component. Possuem um estado interno (state) que pode ser modificado para causar re-renderização.

OSB.: *tornaram-se defasados após a adoção do recurso de Hooks, que será abordado posteriormente na disciplina.*

Exemplo: Componente de Classe

```javascript
class Welcome extends React.Component {

  constructor(props) {
    super(props);
    this.state = {
      message: 'Hello'
    };
  }

  render() {
    return <h1>{this.state.message}</h1>;
  }
}
export default Welcome;
```

Componentes Funcionais

São baseados em função JavaScript, recebendo `props` como argumento e retornam um elemento JSX.

Exemplo: Componente Funcional

```javascript
function Welcome(props) {
  return <h1>Olá, {props.name}!</h1>;
}
export default Welcome;
```

Estrutura Básica de Diretórios - Aplicação React

Quando o assunto é a estrutura de diretórios de uma aplicação React, temos algumas discussões e discordâncias, até porque ela pode variar de acordo com o tamanho e complexidade do projeto, das preferências da equipe e de convenções estabelecidas no mercado. No entanto, existem algumas práticas comuns e bem estabelecidas que podem servir como base para tal organização.

Uma estrutura básica para um projeto React poderia incluir os seguintes diretórios:

src:

- components: contém os componentes reutilizáveis da aplicação.

- pages: contém os componentes que representam as diferentes páginas da aplicação.
- utils: contém funções auxiliares e utilitárias.
- api: contém as chamadas para APIs externas.
- styles: contém os estilos globais ou modulares da aplicação.

public: contém arquivos estáticos como o index.html e outros assets.

Criando Componente de Classe: ./src/components/welcome/index.js

```JavaScript
import React from "react";

class Welcome extends React.Component {

    constructor(props) {
      super(props);
      this.state = {
        message: 'Hello'
      };
    }

    render() {
      return <h1>{this.state.message}</h1>;
    }
  }
  export default Welcome;
```

Utilizando o Componente de Classe: ./src/App.js

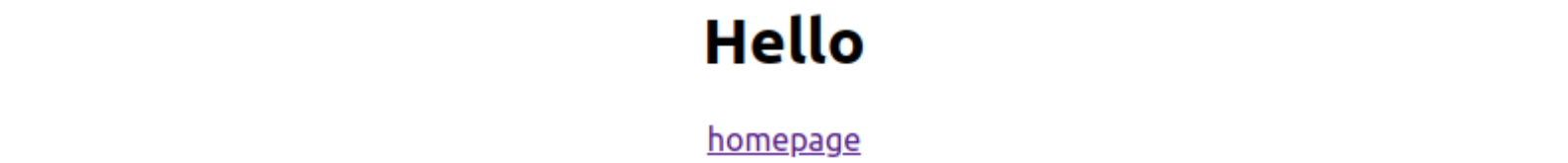
```JavaScript
...
import Welcome from './components/welcome';
...
return (
    <div className="App">
      <Welcome />
...
```

Criando Componente Funcional - props: ./src/components/link/index.js

```javascript
function Link(props) {
    return <a href={props.url} target="_blank">{props.name}</a>;
}
export default Link;
```

Utilizando o Componente Funcional: ./src/App.js

```javascript
...
import Welcome from './components/welcome';
import Link from './components/link';
...
return (
    <div className="App">
      <Welcome />
      <Link url="https://www.gileduardo.com.br" name="homepage"/>
...
```



Componente de Classe - "state": ./src/components/counter/index.js

```javascript
import React from "react";

class Counter extends React.Component {

    constructor(props) {
      super(props);
      this.state = {
        value: props.value,
      };
    }

    addValue = (v) => {
```

```
        this.setState({
            value: parseInt(this.state.value) + v,
        });
    }

    render() {
      return (
        <div>
            <button onClick={() => this.addValue(1)}>+</button>
            <h1 style={{display: 'inline', margin: '10px'}}>
                {this.state.value}
            </h1>
            <button onClick={() => this.addValue(-1)}>-</button>
        </div>
      );
    }
  }
  export default Counter;
```

OSB.: *no React, ao atualizarmos / modificarmos o valor de um "state", ou seja, um dos estados internos do componente, o método "render()" é automaticamente invocado, redesenhando o componente novamente.*

Utilizando o Componente: ./src/App.js

JavaScript

```
...
import Welcome from './components/welcome';
import Link from './components/link';
import Counter from './components/counter';
...
return (
    <div className="App">
      <Welcome />
      <Link url="https://www.gileduardo.com.br" name="homepage"/>
      <Counter value="10"/>
...
```

Hello

homepage

+ **10** -

States com Array e Objetos: ./src/components/list/index.js

JavaScript

```
import {Component} from "react";

export default class List extends Component {
    state = {
        movies: [
            {
                id: 1,
                name: "It - Chapter One",
                director: "Andy Muschietti",
            },
            {
                id: 2,
                name: "X-Men",
                director: "Bryan Singer",
            },
            {
                id: 3,
                name: "Green Mile",
                director: "Frank Darabont",
            },
        ]
    };

    render() {
      return (
        <div style={{marginTop: "10px"}}>
            {
                this.state.movies.map(mov =>
                    <div key={mov.id}>
                        <h2>{mov.name}</h2>
```

```
                    <h5>{mov.director}</h5>
                </div>
            )
        }
      </div>
    );
  }
}
```

OSB.: *podemos criar o "state" do componente sem a utilização do método construtor, considerando que o mesmo não seja necessário. Perceba que também podemos utilizar o comando "export default" na mesma linha em que declaração a classe que representa o componente. Ainda, ao utilizarmos o "map" para percorrer o Array state, criamos vários itens , cada qual contendo a descrição de um filme. Neste contexto torna-se adequado a utilização da propriedade "key", com valor único para cada item. Isso ocorre porque para otimizar a renderização do componente, quando um item é modificado, o React identifica através dessa "key" qual parte do componente deve ser atualizada considerando o identificador do mesmo.*

Utilizando o Componente: ./src/App.js

JavaScript

```
...
import Welcome from './components/welcome';
import Link from './components/link';
import Counter from './components/counter';
import List from './components/list';
...
return (
    <div className="App">
      <Welcome />
      <Link url="https://www.gileduardo.com.br" name="homepage"/>
      <Counter value="10"/>
      <List/>
...
```

INSTITUTO FEDERAL
Paraná
Campus Paranaguá

Ministério da Educação

It - Chapter One

Andy Muschietti

X-Men

Bryan Singer

Ciclo de Vida do Componente
O ciclo de vida de um componente React descreve as diferentes fases pelas quais um componente passa desde a sua criação até a sua remoção da DOM. Conhecer esse ciclo é fundamental para criar componentes eficientes e com comportamentos complexos.
Principais Fases e Métodos

1. *Montagem*:

- *constructor()*: primeiro método invocado quando uma instância de um componente é criada.
- *static getDerivedStateFromProps(props, state)*: permite que um componente leia as novas props e calcule o novo estado, sendo chamado antes do método render.
- *render()*: método mais importante, retorna a representação JSX do componente.
- *componentDidMount()*: método invocado após o componente ser renderizado pela primeira vez na DOM. É um bom lugar para fazer requisições de dados ou iniciar timers.

2. *Atualização*:

- *static getDerivedStateFromProps(props, state)*: invocado antes de render quando as props são atualizadas.
- *shouldComponentUpdate(nextProps, nextState)*: permite que um componente otimize a renderização, retornando false para evitar que o componente seja renderizado desnecessariamente.
- *render()*: renderiza o componente com as novas props ou estado.
- *componentDidUpdate(prevProps, prevState)*: invocado após o componente ser atualizado na DOM, sendo utilizado para realizar ações após a atualização, como efetuar mais requisições de dados.

3. *Desmontagem*:

- *componentWillUnmount()*: invocado imediatamente antes que um componente seja removido da DOM, sendo utilizado para limpar timers, cancelar requisições de rede ou remover event listeners.

Utilizando DidMount() e DidUpdate(): ./src/components/mega/index.js

JavaScript

```
import {Component} from "react";

export default class Mega extends Component {

    state = {
        title: "Sorteando Números...",
        numbers:[],
    };

    generateRandNumbers() {
        let arr = [];

        for(let i=0; i<6; i++) {
            arr[i] = {
                id: i,
                value: (Math.random()*60).toFixed(0),
            }
        }

        return arr;
    }

    componentDidUpdate(prevProps, prevState) {
        // Verfica repetidos
        for(let i=0; i<6; i++) {
            for(let j=i+1; j<6; j++) {
                if(this.state.numbers[i] == this.state.numbers[j]) {
                    this.setState({
                        numbers: this.generateRandNumbers(),
                    });
```

```
                    i=j=7;
                }
            }
        }
    }

    componentDidMount() {
        setTimeout(() => {
            this.setState({
                title: "Números Sorteados:",
                numbers: this.generateRandNumbers(),
            });
        }, 3000);
    }

    render() {
      return (
        <div style={{marginTop: "10px"}}>
            <h1>{this.state.title}</h1>
            {
                this.state.numbers.map(num =>
                  <h5 key={num.id} style={{display: 'inline'}}>
                        {num.value}  
                  </h5>
                )
            }
        </div>
      );
    }
  }
```

Utilizando o Componente: ./src/App.js

JavaScript

```
...
import Welcome from './components/welcome';
import Link from './components/link';
import Counter from './components/counter';
```

```
import List from './components/list';
import Mega from './components/mega';
...
return (
    <div className="App">
      <Welcome />
      <Link url="https://www.gileduardo.com.br" name="homepage"/>
      <Counter value="10"/>
      <List/>
      <Mega/>
...
```

Números Sorteados:

13 4 8 26 17 50

Fetch API

O fetch é uma API de busca do Javascript que permite realizar requisições HTTP assíncronas entre uma aplicação e recursos externos (obtenção de dados), hospedados em servidores web, sendo amplamente utilizado no React.

Obtendo Dados Externos - fetch():

JavaScript

```
import {Component} from "react";

export default class Clients extends Component {

    state = {
        data: [],
        load: true,
        error: null,
    };

    componentDidMount() {
        this.loadDataAPI();
    }
```

```
    async loadDataAPI() {

        try {
                                    const   apiResponse   =   await
fetch('https://jsonplaceholder.typicode.com/users');

            if (!apiResponse.ok) {
                throw new Error('ERRO - response!!');
            }
            const json = await apiResponse.json();
            this.setState({ data: json });
            console.log(this.state.data);

        } catch (e) {
            this.setState = { error: e };
        } finally {
            this.setState({ load: false });
        }
    }

    render() {
      return (
        <div style={{marginTop: "10px"}}>
            {
                this.state.load ?
                    <p>Carregando...</p>
                :
                this.state.error ?
                    <p>Erro: { this.error.message }</p>
                :
                    <div>
                        <hr/>
                        {
                            this.state.data.map((item) => (
                                    <p key={item.id}>{item.name} -
({item.email})</p>
                            ))
```

```
                    }
                    <hr/>
                </div>
            }
        </div>
    );
  }
}
```

Leanne Graham - (Sincere@april.biz)

Ervin Howell - (Shanna@melissa.tv)

Clementine Bauch - (Nathan@yesenia.net)

Patricia Lebsack - (Julianne.OConner@kory.org)

Criando uma Aplicação Simples

A seguir será criada uma aplicação simples utilizando os conceitos do React abordados anteriormente. Contudo, serão aplicados estilos aos componentes criados, dando uma aparência mais profissional à aplicação, ao mesmo tempo em que abordamos boas práticas organizacionais para os arquivos css.

None

```
npx create-react-app my-shadcn-app
npm install web-vitals
```

Estrutura Básica de Diretórios Criada

- ***components***: contém os componentes reutilizáveis da aplicação.
- ***pages***: contém os componentes que representam as diferentes páginas da aplicação.
- ***utils***: contém funções auxiliares e utilitárias.
- ***api***: contém as chamadas para APIs externas.
- ***styles***: contém os estilos globais ou modulares da aplicação.
- **models**: contém a definição dos tipos utilizados na aplicação;

Novo Diretório Criado

- ./src/pages/Home

Arquivos Movidos e Renomeados

- ./src/App.js ⇒ ./src/pages/Home/index.js
- ./src/App.css ⇒ ./src/pages/Home/styles.js
- ./src/App.test.js ⇒ ./src/pages/Home/test.js
- ./src/logo.svg ⇒ ./src/pages/Home/logo.svg

API Utilizada

https://randomuser.me/
https://randomuser.me/documentation
https://randomuser.me/api/?results=10

Adaptações da Codificação

Arquivo: "./src/index.js"

JavaScript

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './styles/index.css';     // linha modificada
import Home from './pages/Home'; // linha modificada
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Home />     // linha modificada
  </React.StrictMode>
);
```

Arquivo: "./src/pages/Home/index.js"

JavaScript

```
import logo from './logo.svg';
import './styles.css';     // linha modificada

function Home() {     // linha modificada
  return (
    <div className="App">
```

```
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          ./src/page/Home        // linha modificada
        </p>
      </header>
    </div>
  );
}

export default Home;  // linha modificada
```

Arquivo: "./src/pages/Home/test.js"

JavaScript

```
import { render, screen } from '@testing-library/react';
import Home from '.';       // linha modificada

test('renders learn react link', () => {
  render(<Home />);         // linha modificada
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});
```

Executando Aplicação

None

```
npm run start
```

Configurando API Externa

Arquivo: "./src/api/index.ts"

```typescript
const USERSAPI = "https://randomuser.me/api/";

export {
    USERSAPI,
};
```

Definindo os Modelos (User)

Arquivo: "./src/models/user.model.ts"

```typescript
export interface RandomUser {
  users: User[];
  load: boolean;
  error: string | null;
}

export interface User {
  gender: 'male' | 'female';
  name: {
    title: string;
    first: string;
    last: string;
  };
  location: {
    street: {
      number: number;
      name: string;
    };
    city: string;
    state: string;
    country: string;
    postcode: number | string;
```

```
    coordinates: {
      latitude: string;
      longitude: string;
    };
    timezone: {
      offset: string;
      description: string;
    };
  };
  email: string;
  login: {
    uuid: string;
    username: string;
  };
  dob: {
    date: string;
    age: number;
  };
  registered: {
    date: string;
    age: number;
  };
  phone: string;
  cell: string;
  id: {
    name: string;
    value: string | null;
  };
  picture: {
    large: string;
    medium: string;
    thumbnail: string;
  };
  nat: string;
}
```

Definindo os Estilos - Home

Arquivo: "./src/page/Home/styles.css"

```css
.App {
  margin: 10px;
}

.card-box {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(380px, 1fr));
  padding: 0px 10px;
}

.card {
  display: grid;
  grid-template-columns: auto auto;
  margin:7px;
  padding: 5px;
  border: 2px solid gray;
  border-radius: 10px;
  box-shadow: 5px 5px 3px 2px gray;
}

.card:hover{
  opacity: 0.9;
  transform: scale(1.02);
}

.card-photo {
  display: flex;
  grid-row-start: 1;
  grid-row-end: 6;
  border-right: 2px solid gray;
  width: min(160px);
  justify-content: center;
  align-items: center;
}

.card-item {
```

```
  display: flex;
  justify-content: center;
  height: 35px;
}

.image {
  border-radius: 100px;
}

.name {
  font-weight: bold;
}

.mail {
  font-size: 14px;
  font-style:italic;
}
```

Buscando Dados Externos - Home

Arquivo: "./src/page/Home/index.tsx"

JavaScript

```
import { Component } from "react";
import { USERSAPI } from "../../api/index.ts";
import './styles.css';
import type { RandomUser } from "../../models/user.model.ts";

export default class Home extends Component<RandomUser> {

    state: RandomUser = {
        users: [],
        load: true,
        error: null,
    }

    componentDidMount() {
        this.loadUsers()
```

```
    }

    async loadUsers() {

        try {
            const apiResponse = await fetch(USERSAPI + "/?results=10");

            if (!apiResponse.ok) {
                throw new Error('ERRO - response!!');
            }

            const json = await apiResponse.json();
            this.setState({ users: json.results });
            console.log(this.state.users);

        } catch (e) {
            this.setState = { error: 'Não foi possível carregar os usuários!' };
            console.log(e);
        } finally {
            this.setState({ load: false });
        }
    }
```

Renderizando a Home

Arquivo: "./src/page/Home/index.js"

TypeScript

```
render() {
        return (
            <div className="App">
            {
            this.state.load ?
                <p>Carregando...</p>
            :
                this.state.error ?
                    <p>Erro: { this.state.error }</p>
                :
```

```
                    <div className="card-box">
                    {
                        this.state.users.map((item) => (
                                        <div  className="card"
key={item.email}>
                                <div className="card-photo">
                                        <img className="image"
src={item.picture.large} />
                                </div>
                                <div className="card-item name">
                                        <p>{item.name.title}.
{item.name.last.toUpperCase() } {item.name.first}</p>
                                </div>
                                <div className="card-item mail">
                                    <p>{item.dob.age} anos</p>
                                </div>
                                <div className="card-item">

<p>{item.location.state}({item.location.country})</p>
                                </div>
                                <div className="card-item">

<p>{item.location.street.name}, {item.location.street.number}</p>
                                </div>
                            </div>
                        ))
                    }
                    </div>
            }
            </div>
        )
    }
}
```

INSTITUTO FEDERAL
Paraná
Campus Paranaguá

Ministério da Educação

Mr. COLE Ron

42 anos

Cork (Ireland)

North Road, 7570

Mr. VELA Porfirio

25 anos

Hidalgo (Mexico)

Periférico Norte Tijerina, 5409

Mr. AKMAN Gökhan

73 anos

Ankara (Turkey)

Anafartalar Cd, 7760

Ms. TEIXEIRA Rosalba

68 anos

Rondônia (Brazil)

Rua São Pedro , 3640