



Paradigmas de Programação

React Native

Cliente HTTP Axios
Acessando Web Services e APIs

Gil Eduardo de Andrade





CLIENTE HTTP AXIOS

Introdução:

- O Axios é um cliente (biblioteca) que permite efetuar requisições HTTP;
- Ele permite que aplicações baseadas no node.js sejam capazes de interagir com serviços web e APIs, possibilitando a integração entre diversas plataformas (web, mobile, etc);





CRIAÇÃO DE COMPONENTES

Instalação:

- A instalação do Axios é feita via terminal de comandos com o *npm*.
 - * *\$ npm install axios*
- O comando deve ser executado dentro do diretório da aplicação React native que utilizará a biblioteca;





CRIAÇÃO DE COMPONENTES

Importação:

- Para utilizar a biblioteca basta uma simples importação, como é feito com outros pacotes e componentes dentro do React Native:

** import axios from 'axios';*





CRIAÇÃO DE COMPONENTES

Utilização:

- O Axios permite utilizar diversos tipos (get, post, put, delete, etc) de requisições HTTP;
- Ao efetuarmos uma requisição com o Axios obtemos como retorno um *promiss*. Ele é utilizado para manipular dados recebidos na invocação de eventos assíncronos;



CRIAÇÃO DE COMPONENTES

Utilização (get):

```
axios.get('/user?ID=12345')  
  .then(function (response) {  
    console.log(response);  
  })  
  .catch(function (error) {  
    console.log(error);  
  });
```

Efetua uma requisição GET para URL */user* passando como dado *ID=12345*. A requisição GET tem por objetivo obter dados, nesse caso de um usuário com identificador *12345*.

Assim que os dados solicitados são recebido (caso a requisição e comunicação com a API funcione corretamente) o método *then()* é invocado.

Caso a requisição e comunicação com a API não ocorra como esperado (erro) o método *catch()* é invocado.

CRIAÇÃO DE COMPONENTES

Utilização (post):

```
axios.post('/user', {
  firstName: 'Fred',
  lastName: 'Flintstone'
})
.then(function (response) {
  console.log(response);
})
.catch(function (error) {
  console.log(error);
});
```

Efetua uma requisição POST para URL */user* passando como dado *um objeto contento nome e sobrenome*. A requisição POST tem por objetivo enviar dados, nesse caso *Fred e Flinstones*.

Assim que os dados solicitados são recebido (caso a requisição e comunicação com a API funcione corretamente) o método *then()* é invocado.

Caso a requisição e comunicação com a API não ocorra como esperado (erro) o método *catch()* é invocado.



Axios na Prática

Monitoramento: App

www.gileduardo.com.br/ifpr/pp_rn/downloads/pp_rn_exapp08.zip



IMPORTAÇÃO / STATE: APP EXEMPLO

Monitoramento (App.js)

```
import axios from 'axios';
export default class App extends React.Component{

  constructor(props) {
    super(props);

    this.state = {
      dados: [],
      dados_api : [],
      total: 5,
      temperatura: "",
      umidade: "",
      carregando : false,
      error : false
    };
  }
}
```

Importa o cliente HTTP axios.

Declara os atributos que definem o estado da aplicação. Os atributos *carregando* e *error* são utilizados para controlar o componente `<ActivityIndicator>` que apresenta uma animação de carregamento enquanto os dados requisitados pelo *axios* estão sendo recebidos. Os arrays *dados[]* e *dados_api[]* recebem, respectivamente, o histórico de temperaturas e umidades e as informações básicas sobre a API.

REQUISIÇÃO GET: APP EXEMPLO

Monitoramento (App.js)

```
componentDidMount() {  
  this.setState({ carregando : true });  
  
  const url = 'http://gileduardo.com.br/react/api/rest.php/' + this.state.total;  
  
  setTimeout(() => {  
    // Dados Básicos da API  
    axios.get('http://gileduardo.com.br/react/api/rest.php')  
      .then(response => {  
        this.setState({  
          dados_api : response.data,  
        });  
      }).catch(error => {  
        this.setState({  
          carregando : false,  
          error : true  
        });  
      });  
  });  
};
```

Seta o state *carregando* como *true* para indicar que o componente **<Activity Indicator>** deve ser renderizado.

Temporizador de 1 segundo utilizado para que o **<Activity>** seja visualizado.

Requisição GET para API, obtém as informações básicas sobre a API, que são armazenadas no objeto *response* atributo *data* (*response.data*). Armazena os dados recebidos no *state dados_api*.

REQUISIÇÃO GET: APP EXEMPLO

Monitoramento (App.js)

```
// Dados do Monitoramento - Temperatura e Umidade
axios.get(url)
  .then(response => {
    this.setState({
      dados : response.data,
      carregando : false
    });
  }).catch(error => {
    this.setState({
      carregando : false,
      error : true
    });
  });
}, 1000);
```

Requisição GET para API, obtém as informações sobre o histórico de monitoramento da temperatura e umidade. Os dados recebidos são armazenados no *state dados_api*.

Seta o *state carregando* com *false* para que o <Activity> desapareça e a tela principal da aplicação seja renderizada em seu lugar.

Caso aja um erro na requisição seta o *state error* com *true* para que uma mensagem de erro seja renderizada.

REQUISIÇÃO POST: APP EXEMPLO

Monitoramento (App.js)

```
const dados = {
  temp: this.state.temperatura,
  umid: this.state.umidade,
};

axios({
  method: 'post',
  url: 'http://gileduardo.com.br/react/api/rest.php',
  headers: {
    "Content-Type": "application/json"
  },
  data: dados
}).then(response => {
```

Objeto *dados*, criado com as *informações de temperatura e umidade* que serão enviados para API efetuar o armazenamento.

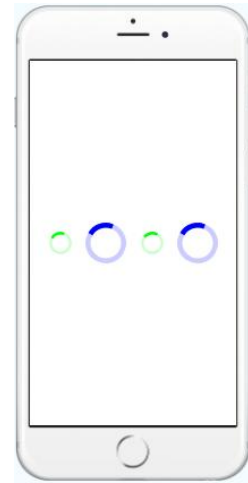
Configura o cliente axios para efetuar uma requisição *POST* contendo (enviando) dados do tipo *JSON*.

Seta o objeto *data* (que armazena os *dados* que serão *enviados pelo axios*) com o objeto *dados*.

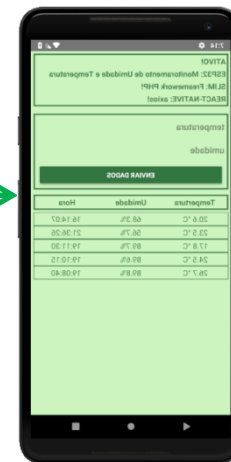
ACTIVITY / CONDIÇÃO: APP EXEMPLO

Monitoramento (App.js)

```
render() {  
  return (  
    <View style={styles.container}>  
      {  
        this.state.carregando ← Testa o state carregando -> (if)  
        ?  
        : <ActivityIndicator size = "large" color="#AA0000"/>  
        ← Caso seja verdade (?) exhibe o <Activity>.  
        this.state.error  
        ?  
        : <Text style={style.error}>Ops!!! Algo deu errado!!! =</Text>  
      }  
    )  
  }  
}
```



Caso não seja verdade o state carregando (:) testa o state error. Sendo verdade exhibe uma mensagem de erro. Caso contrário exhibe a tela principal da aplicação com as informações básicas da API e as informações do monitoramento





Testando a API

POSTMAN

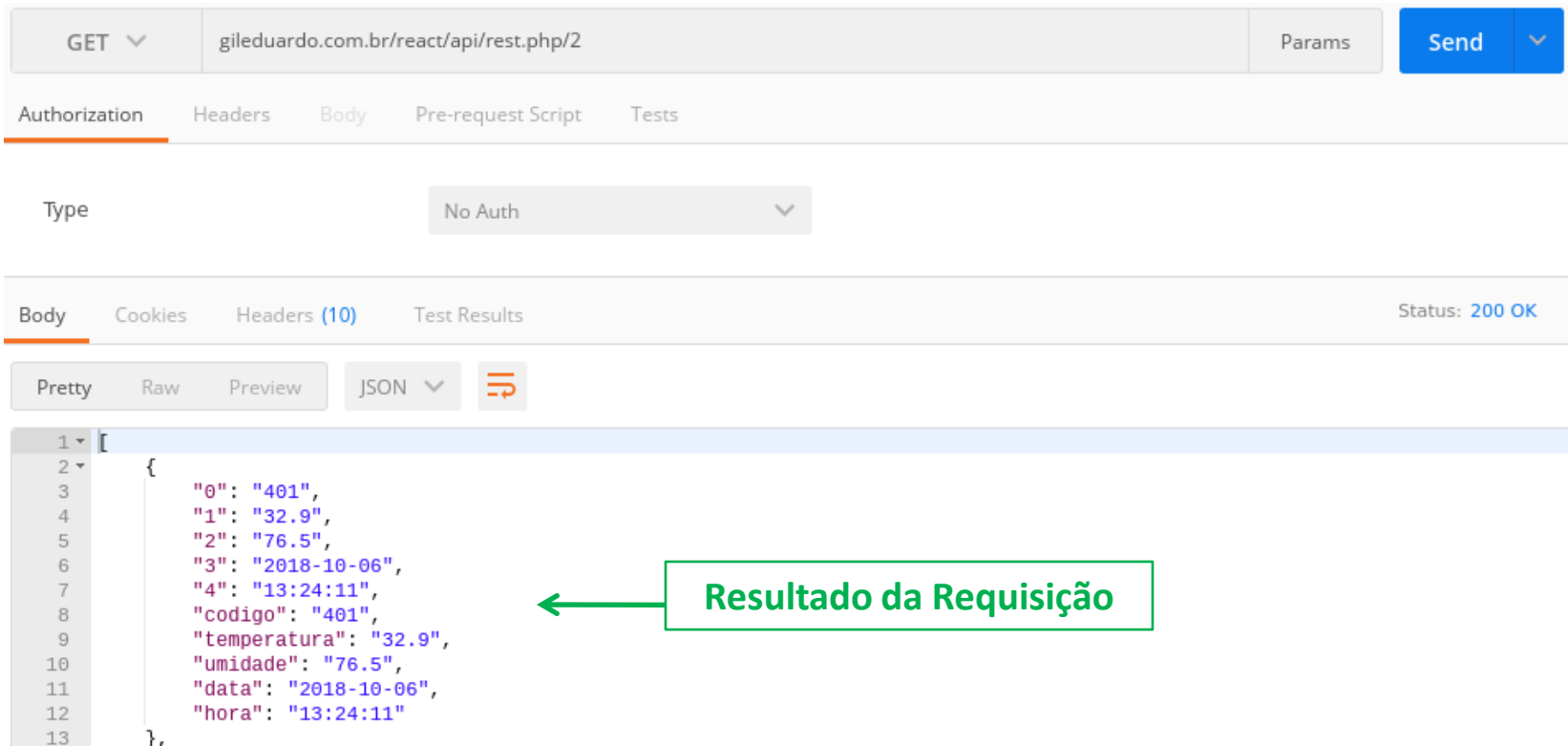
(Mais informações: Web Service / API)

http://www.gileduardo.com.br/ifpr/dwii/downloads/dwii_aula10.pdf



POSTMAN

Testando API – GET



The screenshot shows the Postman interface for a GET request. The URL is `gileduardo.com.br/react/api/rest.php/2`. The request is sent with no authentication. The response status is `200 OK`. The response body is displayed in JSON format, showing a list of weather data points.

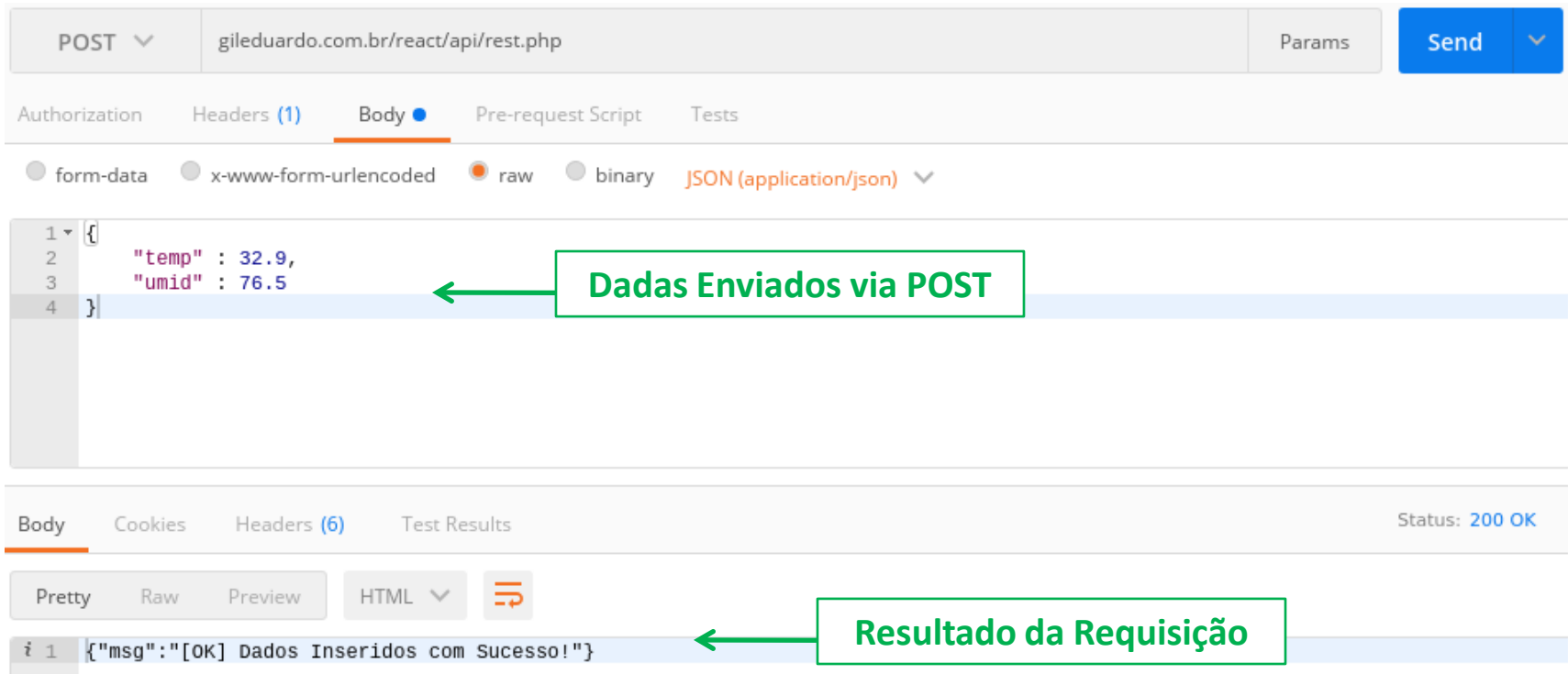
```
1 [
2   {
3     "0": "401",
4     "1": "32.9",
5     "2": "76.5",
6     "3": "2018-10-06",
7     "4": "13:24:11",
8     "codigo": "401",
9     "temperatura": "32.9",
10    "umidade": "76.5",
11    "data": "2018-10-06",
12    "hora": "13:24:11"
13  },
```

← Resultado da Requisição

URL: gileduardo.com.br/react/api/rest.php/2

POSTMAN

Testando API – POST



The screenshot displays the Postman interface for a POST request. The URL is `gileduardo.com.br/react/api/rest.php`. The request body is a JSON object: `{ "temp": 32.9, "umid": 76.5 }`. The response status is `200 OK` and the body is `{"msg":"[OK] Dados Inseridos com Sucesso!"}`. Green callout boxes with arrows point to the request body and the response body.

Dadas Enviados via POST

```
1 {  
2   "temp" : 32.9,  
3   "umid" : 76.5  
4 }
```

Resultado da Requisição

```
i 1 {"msg":"[OK] Dados Inseridos com Sucesso!"}
```

URL: gileduardo.com.br/react/api/rest.php



CRIAÇÃO DE COMPONENTES

Exemplos Utilizados no Documento

Código-fonte do App Exemplo: Calculadora

http://www.gileduardo.com.br/ifpr/pp_rn/downloads/pp_rn_exapp08zip

Exercício sobre o Conteúdo

http://www.gileduardo.com.br/ifpr/pp_rn/downloads/pratica08.pdf

