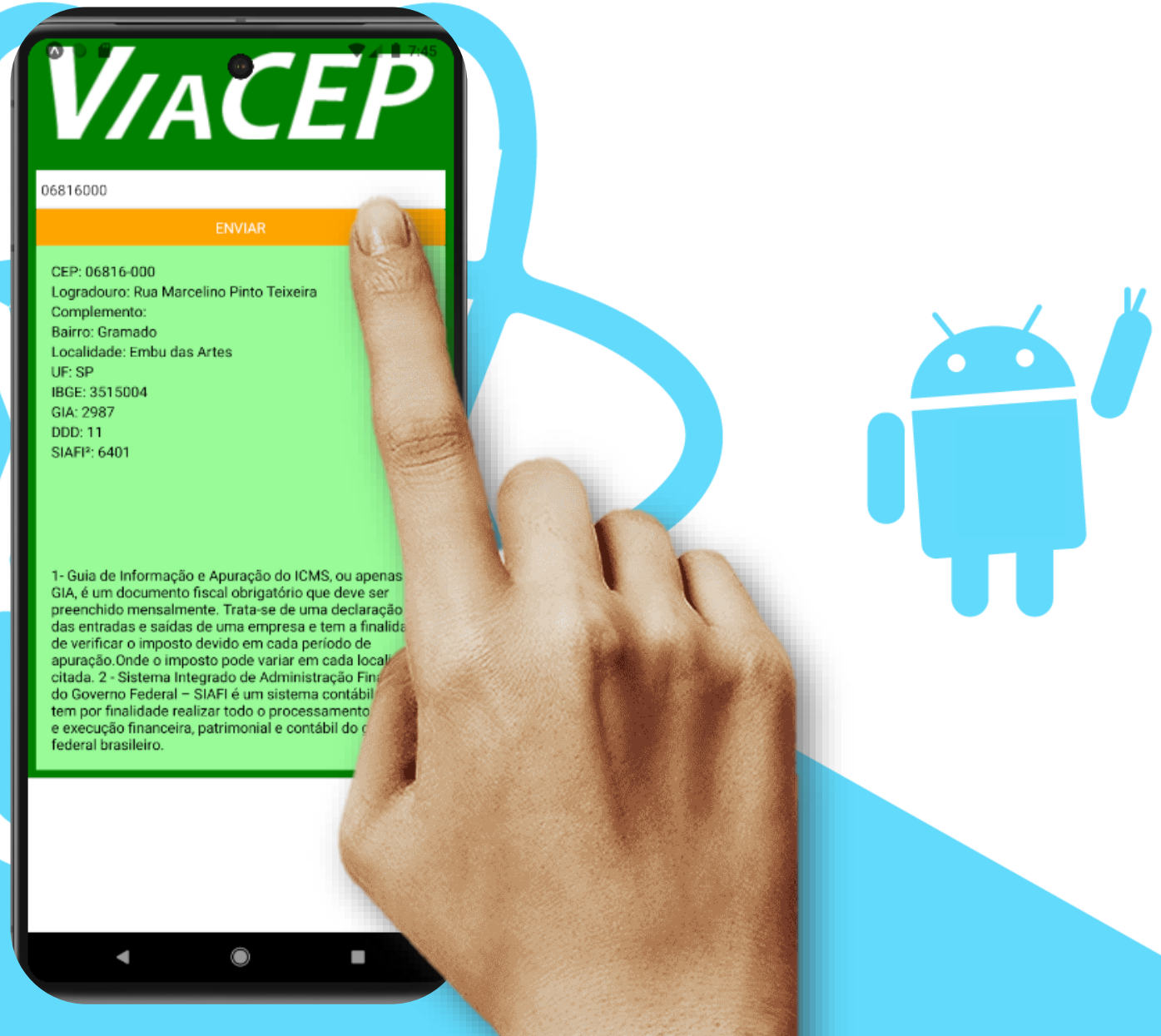
Consumindo API do ViaCEP



O que iremos aprender?

React Native

- Consumindo API Rest;
- JSOn
- Api ViaCEP
- Uso da biblioteca Axios;
- Uso de funções;
- useState;
- atributo onChangeText no TextInput.

O que é uma API?

React Native

APIs são mecanismos que permitem que dois componentes de software se comuniquem usando um conjunto de definições e protocolos. Por exemplo, o sistema de software do instituto meteorológico contém dados meteorológicos diários. A aplicação para a previsão do tempo em seu APP troca dados com esse sistema por meio de APIs e mostra atualizações meteorológicas diárias no telefone.

Consumindo API

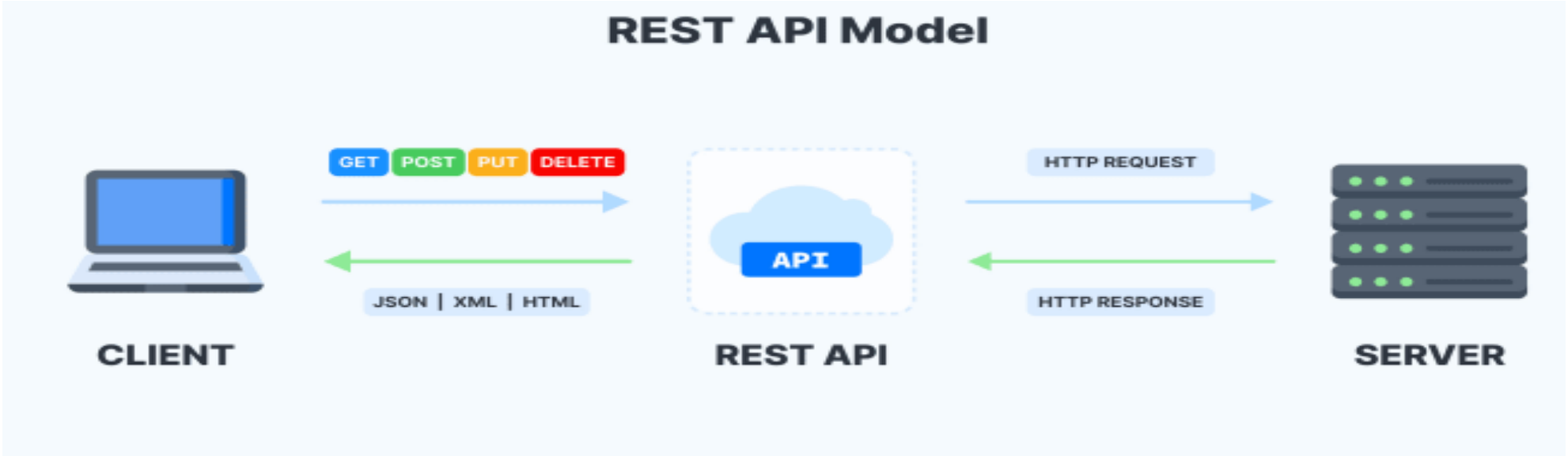
React Native

Quando usamos os recursos produzidos por uma API, dizemos que estamos consumindo uma API, existem vários site que disponibilizam recursos através de uma API, como site de correlação de endereço via CEP, mapas on-line, previsão do tempo, meteorologia, cotações, portais de pagamento, verificações de crédito, informações diversas ou específicas, rádios on-line, streamers, etc.

REST API

React Native

Quando fazemos uma requisição de API, temos como restorno um arquivo com os dados requisitado que podem serem um JSON, XML ou HTML. Isso é solicitado por um servidor de API, que através de um back-end faz a tratativa da consulta e através de um comando request buscam esse dados num banco de dados. Quando esse retorna para o servidor de API ele retorna o arquivo com os dados solicitados.



Consumindo API do ViaCEP

Consumindo API do Via CEP

Um site que presta serviço API de gr~ça para podermos consultar é o ViaCEP em https://viacep.com.br/

Para acessar o webservice, um CEP no formato de {8} dígitos deve ser fornecido, exemplo: “**06816000**".
Após o CEP, deve ser fornecido o tipo de retorno desejado, que deve ser do tipo JSON.

Exemplo de consulta de CEP:
viacep.com.br/ws/**06816000**/json/

Mandamos a requisição usando a url acima lembrando que o texto em laranla é o que queremos consultar.

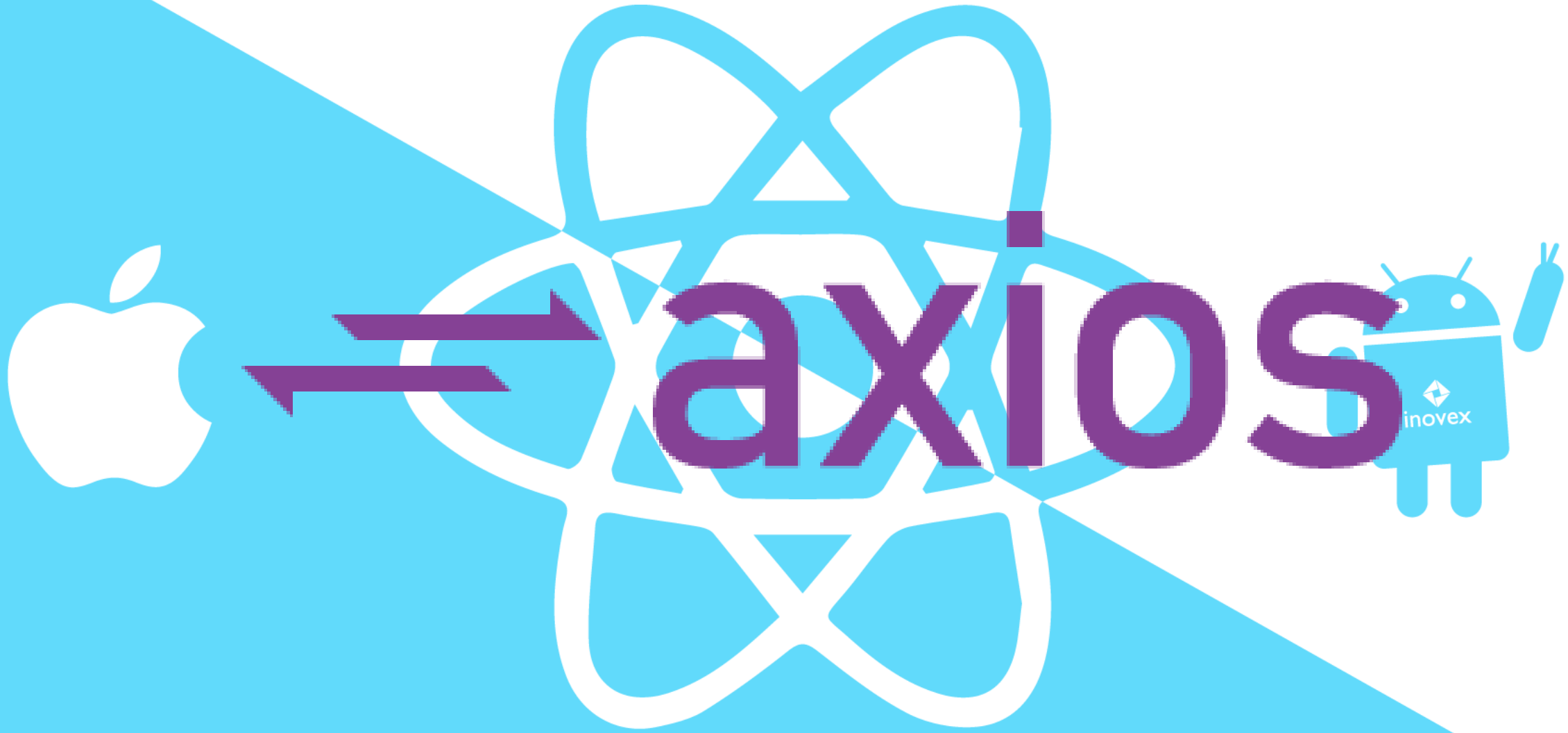
Consumindo API do Via CEP

O retorno é um arquivo JSON como vemos ao lado e podemos usar esses dados para usarmos em um novo APP, sem ter de reinventar a roda:

```
// 20240613184346
// https://viacep.com.br/ws/06816000/json/

{
  "cep": "06816-000",
  "logradouro": "Rua Marcelino Pinto Teixeira",
  "complemento": "",
  "bairro": "Gramado",
  "localidade": "Embu das Artes",
  "uf": "SP",
  "ibge": "3515004",
  "gia": "2987",
  "ddd": "11",
  "siafi": "6401"
}
```

API com uso da biblioteca AXIOS



O que é AXIOS?

Axios é um cliente HTTP baseado em Promises para fazer requisições. Pode ser utilizado tanto no navegador quanto no Node.js ou qualquer serviço de API. Neste artigo criaremos um projeto em React que realiza requisições HTTP a API do GitHub usando o Axios.

Características do Axios

- Usando o Ajax como uma camada abaixo, faz requisições através do browser via XMLHttpRequests;
- Faz requisições HTTP com o Node.js;
- Suporta a Promises;
- Todas as respostas são transformadas e retornadas em JSON;
- Tem suporte a falsificação de solicitações entre sites, conhecido como XRSF.

Métodos de requisição HTTP

O protocolo HTTP define um conjunto de métodos de requisição responsáveis por indicar a ação a ser executada para um dado recurso. Embora esses métodos possam ser descritos como substantivos, eles também são comumente referenciados como HTTP Verbs (Verbos HTTP). Cada um deles implementa uma semântica diferente;

GET => Solicita dados de requisição através da URL, os dados são conhecidos e apresentados na URL;

POST => Submete dados esse não são mostrados na URL o em lugar nenhum, sendo recomendado para postar dados ou qualquer métodos que envolva segurança;

PUT => Faz uma requisição solicitando a substituição de um determinado dado;

DELETE => Solicita a eliminação ou exclusão de determinado dado, especificado.

O que é JSON?

JSON é um arquivo de armazenamento de dados do tipo texto, diferente do XML não possui tags de marcações, os dados são representados como dados salvos em estrutura de dados como ARRAY de dados.

Métodos de requisição HTTP

Também não deve ser confundido banco de dados NO SQL, ele foi desenvolvido inicialmente como meio de transmissão de dados entre APIs e programas ou sistemas de terceiros.

A biblioteca AJAX foi a primeira a permitir a conexão e leitura de dados JSON. Hoje em dia existem várias biblioteca das mais diversas linguagem que fazem a mesma função do AJAX, no nosso caso o pacote npm AXIOS conecta APIs com o NodeJS, React, React Native e Electron.

O que é JSON?

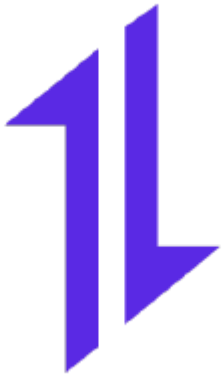
React Native

```
[
    {
      “titulo”: “JSON x XML”,
      “resumo”: “o duelo de dois modelos de representação de informações”,
      “ano”: 2012,
      “genero”: [“aventura”, “ação”, “ficção”]
    },
    {
      “titulo”: “JSON James”,
      “resumo”: “a história de uma lenda do velho oeste”,
      “ano”: 2012,
      “genero”: [“western”]
    }
]
```

Saiba mais...

Vamos conectar nossos projetos

Mas antes para saber mais acesse a documentação em: https://axios-http.com/docs/intro



Iniciando o projeto

`npx create-expo-app --template blank apiViaCEP`
Criamos o nosso APP

`cd apiViaCEP`
Acessamos a pasta de nosso projeto

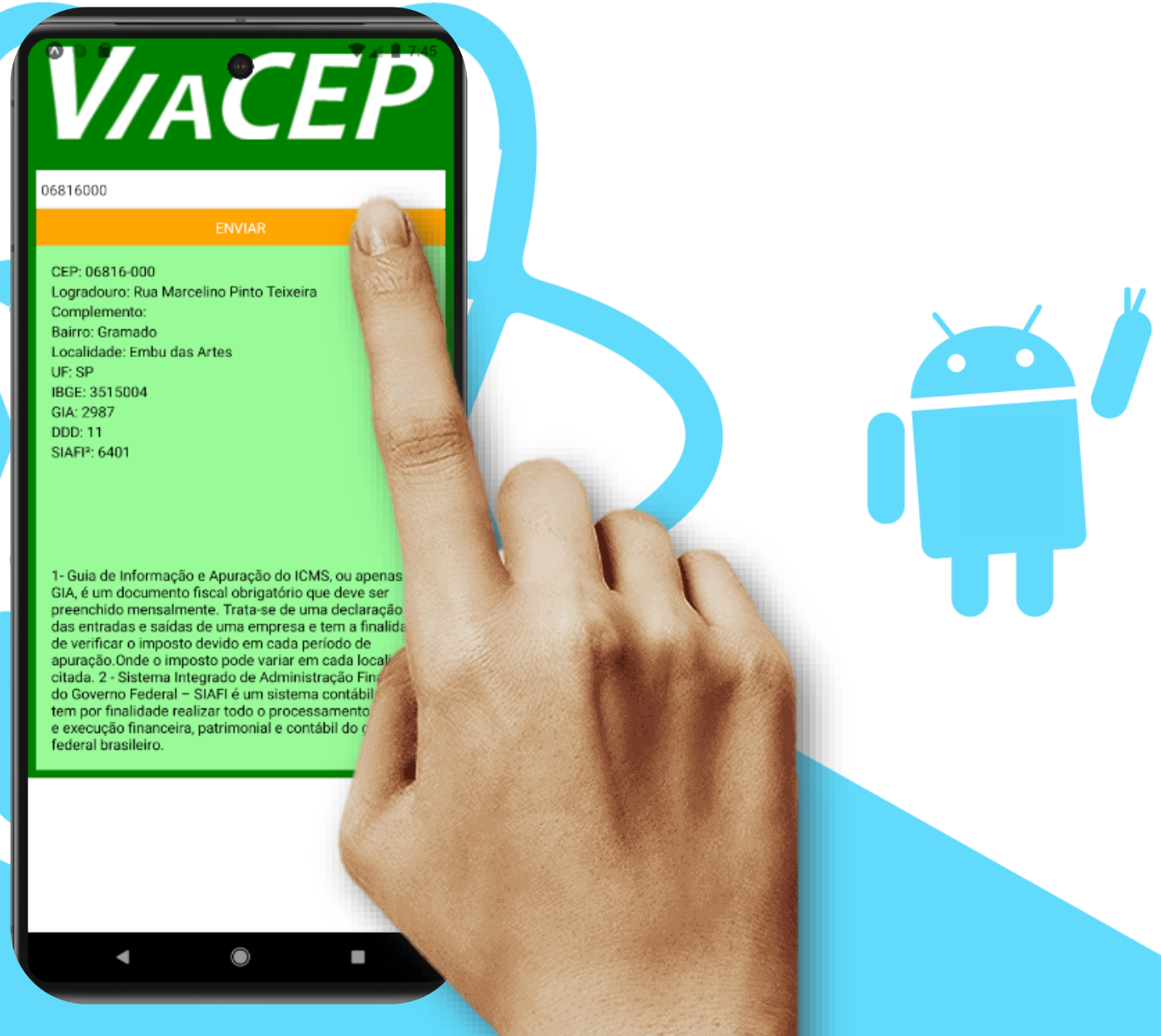
`npm install axios`
Para instalar a biblioteca Axios

`code .`
Abrimos nosso projeto no VisualCode

`npx expo start`
Colocamos nosso projeto para rodar no seu computador

`npx expo start --tunnel`
Colocamos nosso projeto para rodar no seu smartphone

Consumindo API do ViaCEP



Iniciando o projeto

`npx create-expo-app --template blank apiViaCEP`
Criamos o nosso APP

`cd  quiz`
Acessamos a pasta de nosso projeto

`npm install axios`
Para instalar a biblioteca Axios

`code .`
Abrimos nosso projeto no VisualCode

`npx expo start`
Colocamos nosso projeto para rodar no seu computador

`npx expo start --tunnel`
Colocamos nosso projeto para rodar no seu smartphone

App.js função consumirAPI

JS App.js > App

```
1  // Importando os pacotes necessários
2  import React, { useState } from 'react';
3  import { View, Text, TextInput, Button, Image, ScrollView} from 'react-native';
4  import axios from 'axios';
5  import styles from './Styles';
6
7  // Definindo o componente principal do aplicativo
8  export default function App() {
9    // Usando o Hook useState para criar duas variáveis de estado
10   const [cep, setCep] = useState('');
11   const [data, setData] = useState(null);
12
13   // Função assíncrona que é acionada quando o usuário pressiona o botão "Enviar"
14   const consumirAPI = async () => {
15     // Fazendo uma solicitação GET para a API do ViaCEP com o CEP fornecido pelo usuário
16     const response = await axios.get(`https://viacep.com.br/ws/${cep}/json/`);
17     // Armazenando a resposta na variável de estado `data`
18     setData(response.data);
19   };
20
```

App.js render 1ª parte

React Native

```
20
21  // Renderizando o JSX
22  return (
23    <ScrollView>
24    <View style={styles.container}>
25      <View style={styles.logoArea}>
26      <Image source={{uri:'https://viacep.com.br/estatico/images/viacep.png.pagespeed.ce.I80LiA6qpr.png'}} style={styles.logo}
27      </View>
28      <TextInput
29        value={cep}
30        onChangeText={setCep}
31        placeholder="Digite o CEP"
32        style={styles.inputCep}
33      />
34      <Button title="Enviar" onPress={consumirAPI} color={'orange'} style={styles.btn}/>
35
```

App.js render 2ª parte

React Native

```
35
36        {/* Se a variável de estado `data` não for nula, o código renderizará um `View` que exibe as informações do endereço retornadas pela API
37        {data && (
38          <View style={styles.viewArea}>
39            <Text>CEP: {data.cep}</Text>
40            <Text>Logradouro: {data.logradouro}</Text>
41            <Text>Complemento: {data.complemento}</Text>
42            <Text>Bairro: {data.bairro}</Text>
43            <Text>Localidade: {data.localidade}</Text>
44            <Text>UF: {data.uf}</Text>
45            <Text>IBGE: {data.ibge}</Text>
46            <Text>GIA: {data.gia}</Text>
47            <Text>DDD: {data.ddd}</Text>
48            <Text>SIAFI²: {data.siafi}</Text>
49            <Text style={styles.subText}>1- Guia de Informação e Apuração do ICMS, ou apenas GIA, é um documento fiscal obrigatório que deve ser
50            2 - Sistema Integrado de Administração Financeira do Governo Federal – SIAFI é um sistema contábil que tem por finalidade realizar t
51
52          </View>
53        )}
54      </View>
55      </ScrollView>
56    );
57  }
```

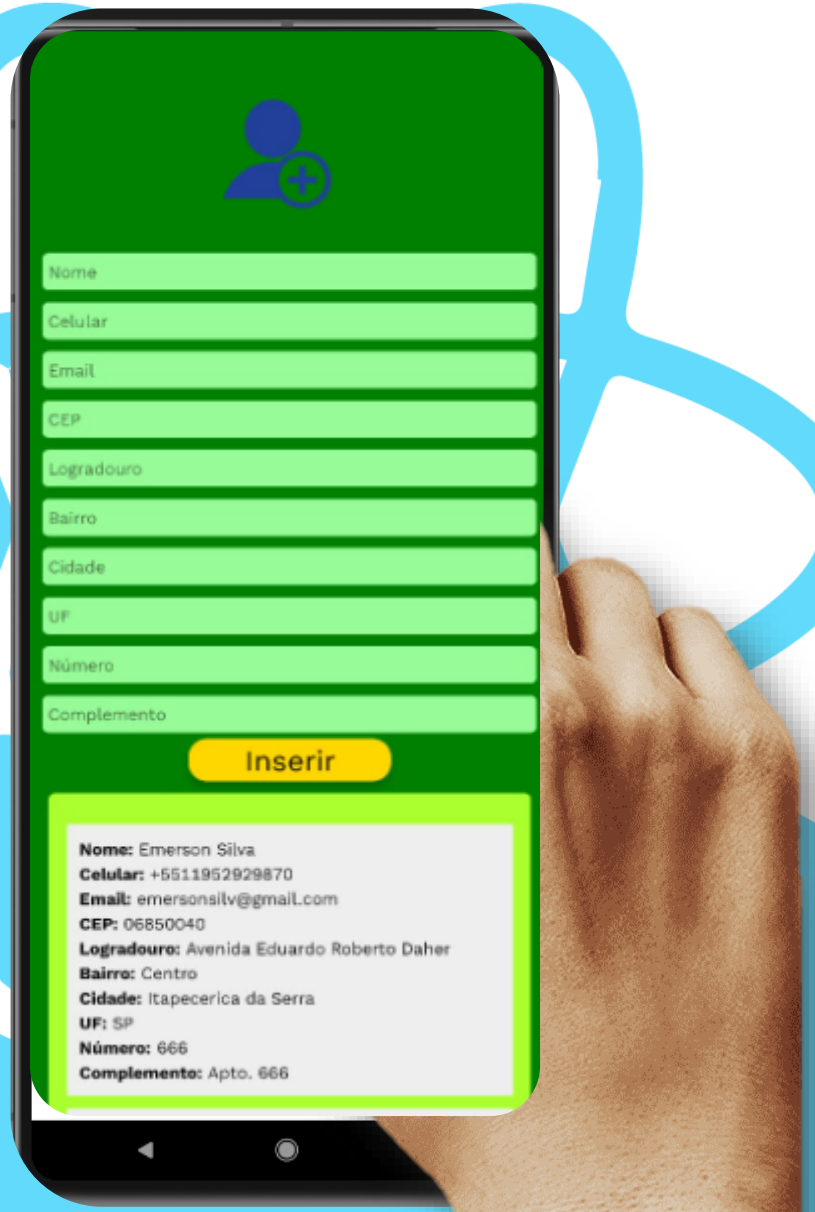
Styles.js

JS Styles.js > default

```
1   import { StyleSheet } from "react-native-web";
2
3   const styles = StyleSheet.create({
4       container: {
5         flex: 1,
6         justifyContent: 'center',
7         padding: 8,
8         backgroundColor: 'green',
9       },
10      logo: {
11          imageResize: 'cover',
12          width: '95%',
13          height: 90,
14          marginBotton: 50,
15          marginLeft: 10,
16      },
17      logoArea: {
18          marginBotton: 50,
19      },
20      inputCep: {
21          marginTop: 30,
22          marginBotton: 20,
23          padding: 5,
24          backgroundColor: '#FFF',
25      },
26      viewArea: {
27          backgroundColor: '#98FB98',
28          padding: 15,
29          fontSize: 16,
30      },
```

```
31      subText: {
32          fontSize: 14,
33          marginTop: 100,
34      },
35      btn: {
36          color: '#000',
37      }
38  });
39
40  export default styles;
```

Consumindo API do ViaCEP no Cadastro



Iniciando o projeto

`npx create-expo-app --template blank cadastroViaCEP`
Criamos o nosso APP

`cd cadastroViaCEP`
Acessamos a pasta de nosso projeto

`npm install axios`
Para instalar a biblioteca Axios

`code .`
Abrimos nosso projeto no VisualCode

`npx expo start`
Colocamos nosso projeto para rodar no seu computador

`npx expo start --tunnel`
Colocamos nosso projeto para rodar no seu smartphone

App.js função cadastroViaCEP parte 1

JS App.js > App > inserirCadastro

```
1   // Importa os pacotes necessários
2   import React, { useState } from 'react';
3   import { View, Text, TextInput, TouchableOpacity, Image, ScrollView, Alert } from 'react-native';
4   import axios from 'axios';
5   import styles from './Styles';
6
7   // Define o componente principal do aplicativo
8   export default function App() {
9
10    // Estado inicial do formulário de cadastro
11    const [formData, setFormData] = useState({
12      nome: '',
13      celular: '',
14      email: '',
15      cep: '',
16      logradouro: '',
17      bairro: '',
18      cidade: '',
19      uf: '',
20      numero: '',
21      complemento: '',
22    });
23
24    // Estado para armazenar cadastros em um array
25    const [cadastros, setCadastros] = useState([]);
26
```

App.js função cadastroViaCEP parte 2


```
27    // Função para buscar informações do CEP na API ViaCEP
28    const buscarCep = async () => {
29      try {
30        // Faz requisição GET para obter dados do endereço pelo CEP digitado
31        const response = await axios.get(`https://viacep.com.br/ws/${formData.cep}/json/`);
32
33        // Se o CEP não for encontrado, exibe um alerta e limpa os campos relacionados
34        if (response.data.erro) {
35          Alert.alert('Erro', 'O CEP informado não retornou dados. Insira os dados manualmente.');
36          setFormData({ ...formData, logradouro: '', bairro: '', cidade: '', uf: '' });
37        } else {
38          // Se encontrado, preenche os campos com os dados retornados pela API
39          setFormData({
40            ...formData,
41            logradouro: response.data.logradouro,
42            bairro: response.data.bairro,
43            cidade: response.data.localidade,
44            uf: response.data.uf,
45          });
46        }
47      } catch (error) {
48        // Em caso de erro na requisição, exibe um alerta
49        Alert.alert('Erro', 'Falha ao buscar o CEP.');
50      }
51    };
52
```

App.js função cadastroViaCEP parte 3


```
52
53    // Função para validar e inserir um novo cadastro
54    const inserirCadastro = () => {
55      // Verifica se todos os campos obrigatórios foram preenchidos
56      if (!formData.nome || !formData.celular || !formData.email || !formData.cep || !formData.numero) {
57        Alert.alert('Erro', 'Todos os campos obrigatórios devem ser preenchidos.');
58        return;
59      }
60
61      // Adiciona o novo cadastro ao array de cadastros
62      setCadastros([...cadastros, formData]);
63
64      // Limpa o formulário para permitir um novo cadastro
65      setFormData({
66        nome: '',
67        celular: '',
68        email: '',
69        cep: '',
70        logradouro: '',
71        bairro: '',
72        cidade: '',
73        uf: '',
74        numero: '',
75        complemento: '',
76      });
77    };
78
```

App.js função cadastroViaCEP parte 4

React Native

```
78
79  // Renderiza a interface do aplicativo
80  return (
81    <ScrollView style={styles.scroll}>
82      <View style={styles.container}>
83        {/* Área do logo */}
84        <View style={styles.logoArea}>
85          <Image source={{ uri: 'https://lojavirtual.compesa.com.br:8443/gsan/imagens/portal/icons/icone_novo_cliente.png' }} style={styles.logo} />
86        </View>
87
88        {/* Campos do formulário */}
89        <TextInput placeholder="Nome" value={formData.nome} onChangeText={(text) => setFormData({ ...formData, nome: text })} style={styles.input} />
90        <TextInput placeholder="Celular" value={formData.celular} onChangeText={(text) => setFormData({ ...formData, celular: text })} style={styles.input} keyboardType="phone-pad" />
91        <TextInput placeholder="Email" value={formData.email} onChangeText={(text) => setFormData({ ...formData, email: text })} style={styles.input} keyboardType="email-address" />
92        <TextInput placeholder="CEP" value={formData.cep} onChangeText={(text) => setFormData({ ...formData, cep: text })} style={styles.input} keyboardType="numeric" onBlur={buscarCep} />
93        <TextInput placeholder="Logradouro" value={formData.logradouro} onChangeText={(text) => setFormData({ ...formData, logradouro: text })} style={styles.input} />
94        <TextInput placeholder="Bairro" value={formData.bairro} onChangeText={(text) => setFormData({ ...formData, bairro: text })} style={styles.input} />
95        <TextInput placeholder="Cidade" value={formData.cidade} onChangeText={(text) => setFormData({ ...formData, cidade: text })} style={styles.input} />
96        <TextInput placeholder="UF" value={formData.uf} onChangeText={(text) => setFormData({ ...formData, uf: text })} style={styles.input} />
97        <TextInput placeholder="Número" value={formData.numero} onChangeText={(text) => setFormData({ ...formData, numero: text })} style={styles.input} keyboardType="numeric" />
98        <TextInput placeholder="Complemento" value={formData.complemento} onChangeText={(text) => setFormData({ ...formData, complemento: text })} style={styles.input} />
99
100         {/* Botão com bordas arredondadas e sombra externa */}
101         <TouchableOpacity style={styles.button} onPress={inserirCadastro}>
102           <Text style={styles.buttonText}>Inserir</Text>
103         </TouchableOpacity>
104
```

App.js função cadastroViaCEP parte 5

React Native

```
104
105
106   <View style={styles.viewArea}>
107     {cadastros.map((cadastro, index) => (
108       <View key={index} style={styles.card}>
109         <Text><Text style={{ fontWeight: 'bold' }}>Nome:</Text> {cadastro.nome}</Text>
110         <Text><Text style={{ fontWeight: 'bold' }}>Celular:</Text> {cadastro.celular}</Text>
111         <Text><Text style={{ fontWeight: 'bold' }}>Email:</Text> {cadastro.email}</Text>
112         <Text><Text style={{ fontWeight: 'bold' }}>CEP:</Text> {cadastro.cep}</Text>
113         <Text><Text style={{ fontWeight: 'bold' }}>Logradouro:</Text> {cadastro.logradouro}</Text>
114         <Text><Text style={{ fontWeight: 'bold' }}>Bairro:</Text> {cadastro.bairro}</Text>
115         <Text><Text style={{ fontWeight: 'bold' }}>Cidade:</Text> {cadastro.cidade}</Text>
116         <Text><Text style={{ fontWeight: 'bold' }}>UF:</Text> {cadastro.uf}</Text>
117         <Text><Text style={{ fontWeight: 'bold' }}>Número:</Text> {cadastro.numero}</Text>
118         <Text><Text style={{ fontWeight: 'bold' }}>Complemento:</Text> {cadastro.complemento}</Text>
119       </View>
120     ))}
121   </View>
122         </View>
123       </ScrollView>
124     );
125   }
```

App.js função cadastroViaCEP parte 6

React Native

JS Styles.js > styles > input

```
import { StyleSheet } from "react-native";

const styles = StyleSheet.create({
    scroll: {
        backgroundColor: 'green',
    },
    container: {
      flex: 1,
      justifyContent: 'center',
      padding: 8,
      backgroundColor: 'green',
    },
   logo: {
        resizeMode: 'contain',
        width: '95%',
        height: 150,
    },
    logoArea: {
        marginTop:20,
    },
    inputCep: {
        marginTop: 10,
        marginBotton: 20,
        padding: 5,
        backgroundColor: '#FFF',
    },
    input: {
        margin: 5,
        padding: 5,
        backgroundColor: '#98FB98',
        borderRadius: 5,
    },
    viewArea: {
        backgroundColor: '#ADFF2F',
        padding: 15,
        fontSize: 16,
        margin: 10,
        borderRadius: 5,
    },
    subText: {
        fontSize: 14,
        marginTop: 100,
    },
    button: {
        color: '#000',
        backgroundColor: "#FFD700",
        borderRadius: 15,
        width: '40%',
        marginLeft: 'auto',
        marginRight: 'auto',
        elevation: 5,
    },
    buttonText: {
        textAlign: 'center',
        fontSize: 24,
    },
    card: {
        backgroundColor: '#EEE',
        padding: 10,
        marginTop: 10,
    }
});

export default styles;
```

Style.js