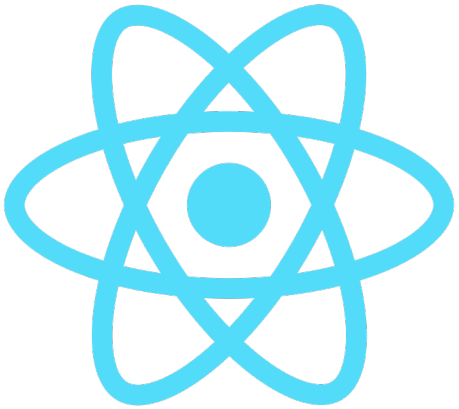
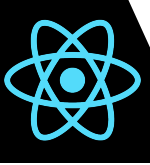


# REACT NATIVE

FRAMEWORK PARA CRIAÇÃO DE APLICAÇÕES

Prof. Guilherme A. Madalozzo
Ph.D. em Ciência da Computação

# Comunicando com Bluetooth



# REACT NATIVE

O aplicativo EXPO não tem acesso ao hardware do Bluetooth para que possamos fazer testes de comunicação

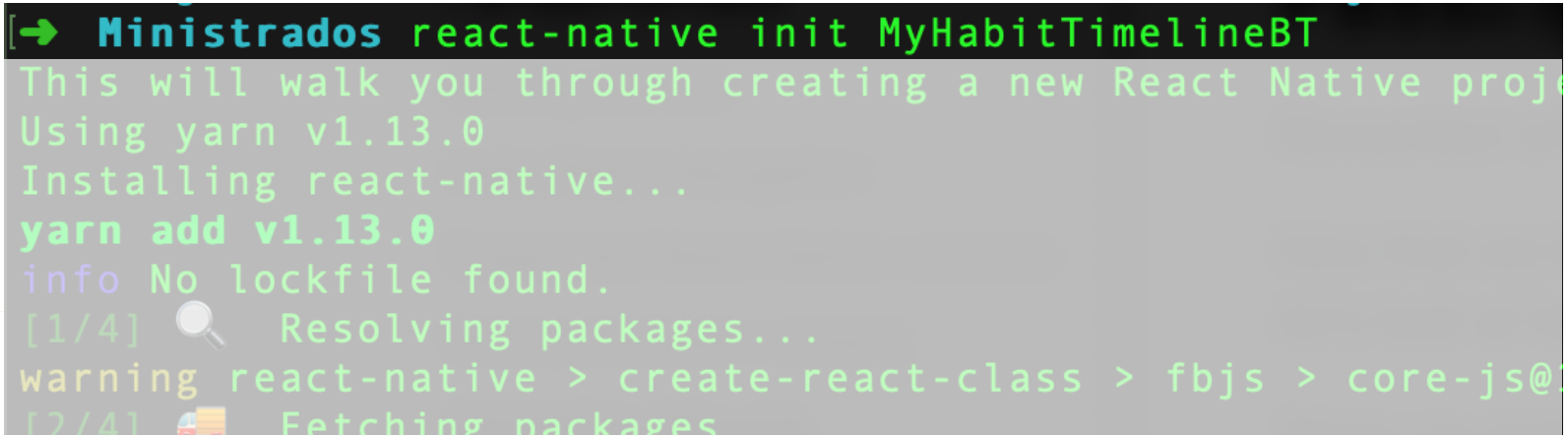
Vamos criar um projeto novo sem o uso do EXPO

# REACT NATIVE

O aplicativo EXPO não tem acesso ao hardware do Bluetooth para que possamos fazer testes de comunicação

Vamos criar um projeto novo sem o uso do EXPO

```
$ react-native init MyHabitTimelineBT
```



# REACT NATIVE

Projetando App.js

Como não vamos usar o EXPO, temos que preparar o ambiente e o celular para podermos testar via USB

**https://facebook.github.io/react-native/docs/running-on-device**

**react-native run-android**

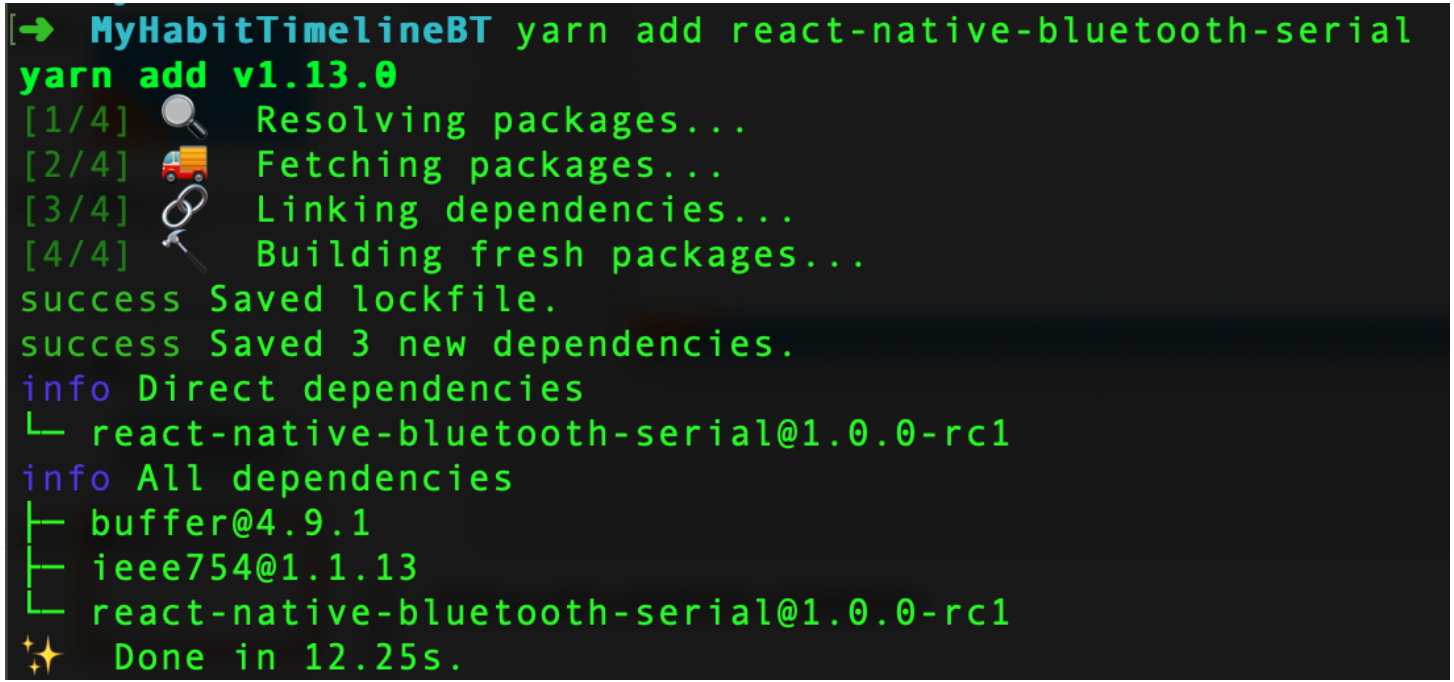
**react-native start**

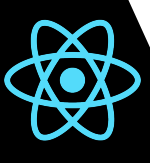
```
export ANDROID_HOME=$HOME/Library/Android/sdk
export PATH=$PATH:$ANDROID_HOME/emulator
export PATH=$PATH:$ANDROID_HOME/tools
export PATH=$PATH:$ANDROID_HOME/tools/bin
export PATH=$PATH:$ANDROID_HOME/platform-tools
```

# REACT NATIVE

Adicionamos ao projeto o uso do componente **`react-native-bluetooth-serial`**

```
$ yarn add react-native-bluetooth-serial
```





# REACT NATIVE

”Linkamos” ao projeto este componente

```
$ react-native link react-native-bluetooth-serial
```

```
➜  MyHabitTimelineBT react-native link react-native-bluetooth-serial
info Linking "react-native-bluetooth-serial" iOS dependency
info iOS module "react-native-bluetooth-serial" has been successfully linked
info Linking "react-native-bluetooth-serial" Android dependency
info Android module "react-native-bluetooth-serial" has been successfully linked
```

# REACT NATIVE

## Projetando embarcado

Como é projeto simples (on/off led)
foi feito com Arduino IDE

EmbeddedBLE §

```
#include "SoftwareSerial.h"

SoftwareSerial bluetooth(2, 3); //TX, RX (do Bluetooth)
const int ledVermelho = 10;
const int ledVerde = 8;

void setup() {
  Serial.begin(38400);
  Serial.println(F("Type the AT commands:"));

  // Polarizacao invertida: 0(liga) / 1(desliga)
  pinMode(ledVermelho, OUTPUT);
  pinMode(ledVerde, OUTPUT);

  digitalWrite(10, 1);
  digitalWrite(ledVerde,  1);
  //Initialize the software serial
  bluetooth.begin(38400);
}

void loop() {

  // Verifica se o monitor serial enviou dados
  if (Serial.available()) {
    char r = Serial.read(); // Lê o dado e salva em registrador
    bluetooth.print(r);  // Envia o dado recebido do serial para o Bluetooth
    Serial.print(r);  // Imprime o dado
  }

  // Verifica se bluetooth recebeu um dado
  if (bluetooth.available()) {
    char r = bluetooth.read(); // Lê o dado e salva em registrador
    Serial.print(r); //Imprime o dado recebido

    if(r=='0') {
      Serial.print("ACESSO NEGADO");
      digitalWrite(ledVermelho, 0);
      digitalWrite(ledVerde,  1);
    }
    else if(r=='1') {
      Serial.print("ACESSO PERMITIDO");
      digitalWrite(ledVermelho, 1);
      digitalWrite(ledVerde,  0);
    }
  }
}
```

# REACT NATIVE

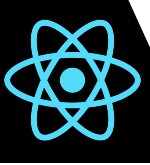
Projetando embarcado

EmbeddedBLE §

```
#include "SoftwareSerial.h"

SoftwareSerial bluetooth(2, 3); //TX, RX (do Bluetooth)
const int ledVermelho = 10;
const int ledVerde = 8;

void setup() {
  Serial.begin(38400);
  Serial.println(F("Type the AT commands:"));
```

Comunicação com SoftwareSerial.h

```
    Serial.print("ACESSO NEGADO");
    digitalWrite(ledVermelho, 0);
    digitalWrite(ledVerde,  1);
  }
  else if(r=='1') {
    Serial.print("ACESSO PERMITIDO");
    digitalWrite(ledVermelho, 1);
    digitalWrite(ledVerde,  0);
  }
 }
}
```



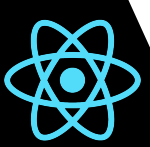
# REACT NATIVE

Projetando embarcado

```
void setup() {
  Serial.begin(38400);
  Serial.println(F("Type the AT commands:"));

  // Polarizacao invertida: 0(liga) / 1(desliga)
  pinMode(ledVermelho, OUTPUT);
  pinMode(ledVerde, OUTPUT);

  digitalWrite(10, 1);
  digitalWrite(ledVerde,  1);
  //Initialize the software serial
  bluetooth.begin(38400);
}
```

EmbeddedBLE §

```
#include "SoftwareSerial.h"

SoftwareSerial bluetooth(2, 3); //TX, RX (do Bluetooth)
const int ledVermelho = 10;
const int ledVerde = 8;

void setup() {
  Serial.begin(38400);
  Serial.println(F("Type the AT commands:"));

  // Polarizacao invertida: 0(liga) / 1(desliga)
  pinMode(ledVermelho, OUTPUT);
  pinMode(ledVerde, OUTPUT);

  digitalWrite(10, 1);
  digitalWrite(ledVerde,  1);
  //Initialize the software serial
  bluetooth.begin(38400);
}

void loop() {

  // Verifica se o monitor serial enviou dados
  if (Serial.available()) {
    char r = Serial.read(); // Lê o dado e salva em registrador
    bluetooth.print(r);  // Envia o dado recebido do serial para o Bluetooth
    Serial.print(r);  // Imprime o dado
  }

  // Verifica se bluetooth recebeu um dado
  if (bluetooth.available()) {
    char r = bluetooth.read(); // Lê o dado e salva em registrador
    Serial.print(r); //Imprime o dado recebido

    if(r=='0') {
      Serial.print("ACESSO NEGADO");
      digitalWrite(ledVermelho, 0);
      digitalWrite(ledVerde,  1);
    }
    else if(r=='1') {
      Serial.print("ACESSO PERMITIDO");
      digitalWrite(ledVermelho, 1);
      digitalWrite(ledVerde,  0);
    }
  }
}
```



# REACT NATIVE

```
void loop() {

  // Verifica se o monitor serial enviou dados
  if (Serial.available()) {
    char r = Serial.read(); // Lê o dado e salva em registrador
    bluetooth.print(r);  // Envia o dado recebido do serial para o Bluetooth
    Serial.print(r);  // Imprime o dado
  }

  // Verifica se bluetooth recebeu um dado
  if (bluetooth.available()) {
    char r = bluetooth.read(); // Lê o dado e salva em registrador
    Serial.print(r); //Imprime o dado recebido

    if(r=='0') {
      Serial.print("ACESSO NEGADO");
      digitalWrite(ledVermelho, 0);
      digitalWrite(ledVerde,  1);
    }
    else if(r=='1') {
      Serial.print("ACESSO PERMITIDO");
      digitalWrite(ledVermelho, 1);
      digitalWrite(ledVerde,  0);
    }
  }
}
```

EmbeddedBLE §

```
#include "SoftwareSerial.h"

SoftwareSerial bluetooth(2, 3); //TX, RX (do Bluetooth)
const int ledVermelho = 10;
const int ledVerde = 8;

void setup() {
  Serial.begin(38400);
  Serial.println(F("Type the AT commands:"));

  // Polarizacao invertida: 0(liga) / 1(desliga)
  pinMode(ledVermelho, OUTPUT);
  pinMode(ledVerde, OUTPUT);

  digitalWrite(10, 1);
  digitalWrite(ledVerde,  1);
  //Initialize the software serial
  bluetooth.begin(38400);
}
```

```
void loop() {

  // Verifica se o monitor serial enviou dados
  if (Serial.available()) {
    char r = Serial.read(); // Lê o dado e salva em registrador
    bluetooth.print(r);  // Envia o dado recebido do serial para o Bluetooth
    Serial.print(r);  // Imprime o dado
  }

  // Verifica se bluetooth recebeu um dado
  if (bluetooth.available()) {
    char r = bluetooth.read(); // Lê o dado e salva em registrador
    Serial.print(r); //Imprime o dado recebido

    if(r=='0') {
      Serial.print("ACESSO NEGADO");
      digitalWrite(ledVermelho, 0);
      digitalWrite(ledVerde,  1);
    }
    else if(r=='1') {
      Serial.print("ACESSO PERMITIDO");
      digitalWrite(ledVermelho, 1);
      digitalWrite(ledVerde,  0);
    }
  }
}
```

# REACT NATIVE

## Projetando App

Como é projeto simples (on/off led), vamos fazer um tela simples no aplicativo

A tela vai listar os dispositivos bluetooth e permitir conexão

Terá botão para ativer led verde ou vermelho

A sequência de comunicação apresentada foi retirada da documentação do componente instalado

# REACT NATIVE

Projetando App.js

Adicionamos os componentes que vamos utilizar nesta tela

JS App.js ×

MyHabitTimelineBT ▸ JS App.js ▸ styles

```
1   import React from 'react';
2   import { StyleSheet, Text, View, Button, FlatList,
3            Switch, TouchableOpacity, ToastAndroid } from 'react-native';
4   import BluetoothSerial from 'react-native-bluetooth-serial'
5
6   export default class App extends React.Component {
7
8   }
```

# REACT NATIVE

Projetando App.js

Montamos os estilos que serão utilizados

App.js

MyHabitTimelineBT > App.js > ...

```
const styles = StyleSheet.create({
  container: {
      flex: 1,
      backgroundColor: '#F5FCFF',
  },
  toolbar:{
      paddingTop:30,
      paddingBottom:30,
      flexDirection:'row'
  },
  toolbarButton:{
      width: 50,
      marginTop: 8,
  },
  toolbarTitle:{
      textAlign:'center',
      fontWeight:'bold',
      fontSize: 20,
      flex:1,
      marginTop:6
  },
  deviceName: {
      fontSize: 17,
      color: "black"
  },
  deviceNameWrap: {
      margin: 10,
      borderBottomWidth:1
  }
});
```

# REACT NATIVE

Projetando App.js

Variáveis de controle
do bluetooth
do celular

JS App.js

MyHabitTimelineBT ▸ JS App.js ▸ styles

```
6   export default class App extends React.Component {
7     constructor (props) {
8       super(props)
9
10      this.state = {
11        isEnabled: false,
12        discovering: false,
13        devices: [],
14        unpairedDevices: [],
15        connected: false,
16      }
17    }
18  }
```

# REACT NATIVE

Projetando App.js

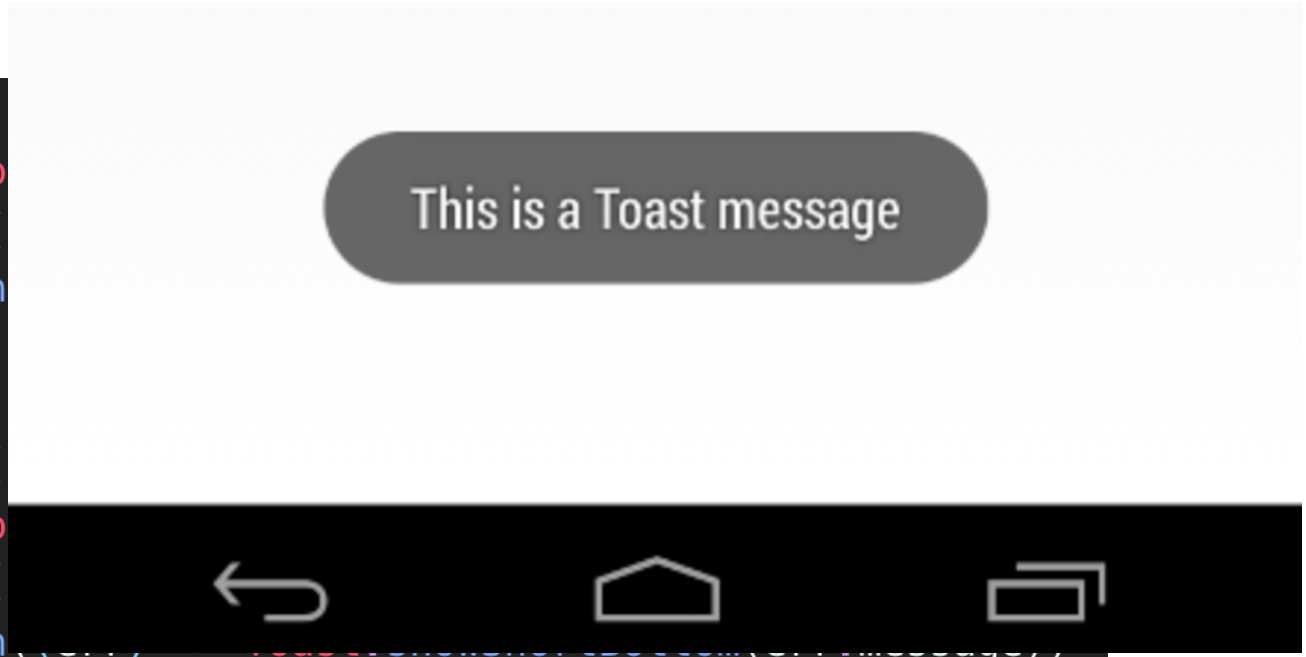
```
19    enable() {
20      BluetoothSerial.enable()
21      .then((res) => this.setState({ isEnabled: true }))
22      .catch((err) => Toast.showShortBottom(err.message))
23    }
24
25    disable() {
26      BluetoothSerial.disable()
27      .then((res) => this.setState({ isEnabled: false }))
28      .catch((err) => Toast.showShortBottom(err.message))
29    }
```

Criamos 2 novos métodos para controlar Bluetooth

# REACT NATIVE

Projetando App.js

```
19    enable()
20      Blueto
21      .then(
22      .catch
23    }
24
25    disable(
26      Blueto
27      .then(
28      .catch
29    }
```



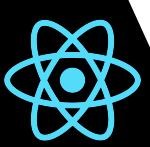

Criamos 2 novos métodos para controlar Bluetooth

# REACT NATIVE

Projetando App.js

```
19    toggleBluetooth(value) {
20      if (value === true) {
21        this.enable()
22      } else {
23        this.disable()
24      }
25    }
```

Criamos o método que permitirá ligar e desligar o Bluetooth



# REACT NATIVE

```
39  discoverAvailableDevices() {
40    if (this.state.discovering) {
41      return false;
42    }
43    else {
44      this.setState({ discovering: true });
45
46      BluetoothSerial.discoverUnpairedDevices()
47      .then((unpairedDevices) => {
48        const uniqueDevices = _.uniqBy(unpairedDevices, 'id');
49        console.log(uniqueDevices);
50        this.setState({ unpairedDevices: uniqueDevices, discovering: false })
51      })
52      .catch((err) => console.log(err.message))
53    }
54  }
```

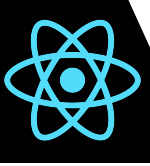
Criamos o método que buscará os dispositivos para conexão

# REACT NATIVE

Projetando App.js

Criamos os métodos que irão enviar os comandos (0 ou 1) para o dispositivo conectado

```
activeGreenLed() {
  BluetoothSerial.write("0")
  .then((res) => {
    console.log(res);
    console.log('Successfuly wrote to device')
  this.setState({ connected: true })
  })
  .catch((err) => console.log(err.message))
}

activeRedLed() {
  BluetoothSerial.write("1")
  .then((res) => {
    console.log(res);
    console.log('Successfuly wrote to device')
  this.setState({ connected: true })
  })
  .catch((err) => console.log(err.message))
}
```



# REACT NATIVE

Projetando App.js

```
76  connect(device) {
77    this.setState({ connecting: true });
78
79    BluetoothSerial.connect(device.id)
80    .then((res) => {
81      console.log(`Connected to device ${device.name}`);
82      ToastAndroid.show(`Connected to device ${device.name}`, ToastAndroid.SHORT);
83    })
84    .catch((err) => console.log((err.message)))
85  }
```

Criamos o método para conexão do app ao dispositivo escolhido

```
19  componentWillMount() {
20    Promise.all([
21      BluetoothSerial.isEnabled(),
22      BluetoothSerial.list()
23    ])
24    .then((values) => {
25      const [isEnabled, devices] = values
26      this.setState({ isEnabled, devices })
27    });
28
29    BluetoothSerial.on('bluetoothEnabled', () => {
30      Promise.all([
31        BluetoothSerial.isEnabled(),
32        BluetoothSerial.list()
33      ])
34      .then((values) => {
35        const [ devices ] = values
36        this.setState({  devices })
37      });
38
39      BluetoothSerial.on('bluetoothDisabled', () => {
40        this.setState({ devices: [] })
41      });
42
43      BluetoothSerial.on('error', (err) => console.log(`Error: ${err.message}`))
44    });
45  }
```

Executamos WillMount antes de montar todos os componentes em tela

Preparamos listeners do bluetooth para quando ações forem executadas nele

# REACT NATIVE

Projetando App.js

```
115   renderDevices(item) {
116     return (
117       <TouchableOpacity onPress={() => this.connect(item.item)}>
118         <View style={styles.deviceNameWrap}>
119           <Text style={styles.deviceName}>
120             { item.item.name ? item.item.name : item.item.id }
121           </Text>
122         </View>
123       </TouchableOpacity>
124     )
125   }
```

Criamos método para renderizar a listagem de dispositivos encontrados para conexão

# REACT NATIVE

Projetando App.js

Monta tela com:

- Botão on/off BT
- Listagem dispositivos
- Botão para ligar Led Verde
- Botão para ligar Led Vermelho

```
render() {
  return (
    <View style={styles.container}>
      <View style={styles.toolbar}>
        <Text style={styles.toolbarTitle}>Bluetooth Device List</Text>
        <View style={styles.toolbarButton}>
          <Switch
            value={this.state.isEnabled}
            onValueChange={(val) => this.toggleBluetooth(val)}
          />
        </View>
      </View>

      <Button
        onPress={this.discoverAvailableDevices.bind(this)}
        title="Scan for Devices"
        color="#841584"
      />

      <FlatList
        style={{flex:1}}
        data={this.state.devices}
        keyExtractor={item => item.id}
        renderItem={(item) => this.renderDevices(item)}
      />

      <Button
        onPress={this.activeGreenLed.bind(this)}
        title="Green"
        color="#287731"
      />

      <Button
        onPress={this.activeRedLed.bind(this)}
        title="Red"
        color="#aa3443"
      />
    </View>
  )
}
}
```