

The background is a dark blue gradient. On the left, there is a large, semi-transparent circular image of a circuit board. Overlaid on this and the background are several geometric shapes: a blue parallelogram and a light green parallelogram in the upper left, and a grey, 3D-like circuit pattern in the upper right.

Procesual Hito 2

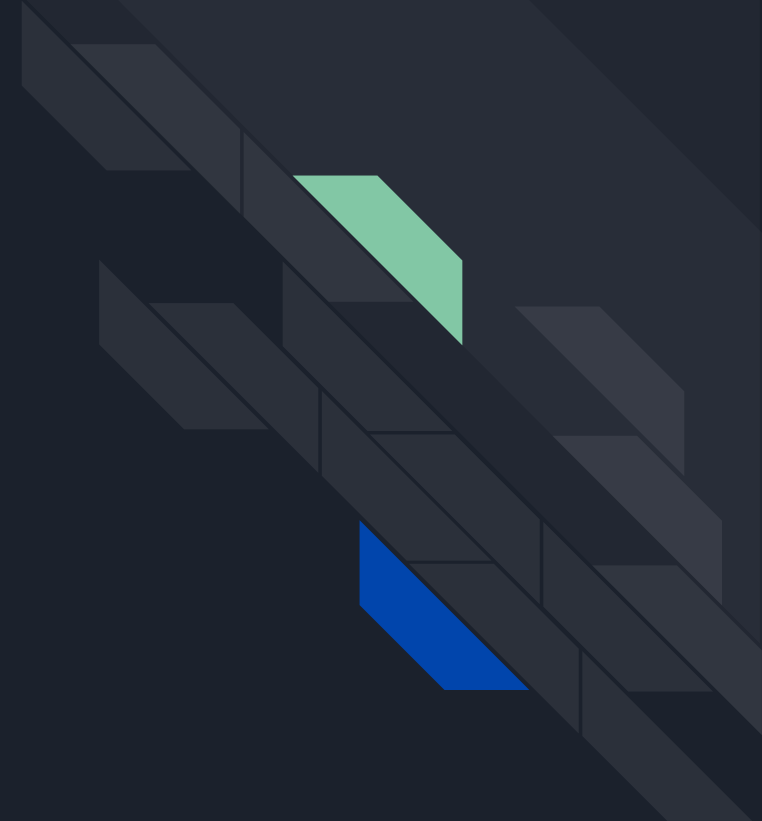
Raquel Terrazas

Contenidos

Resumen

Análisis de los problemas

Desarrollo





Resumen

La presentación actual, mostrará el proyecto propuesto para el hito 2 de la materia de programación de dispositivos móviles, con parte de la explicación para comprender el funcionamiento del mismo.



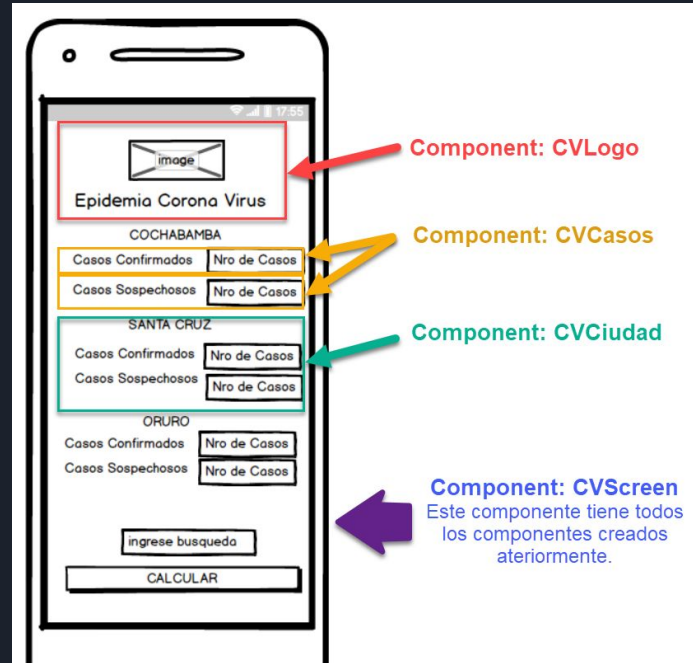
Análisis del Problema

Dado el contexto social de nuestro planeta que va atravesando por este virus viral se toma como caso de uso para la presente defensa correspondiente al hito 2.

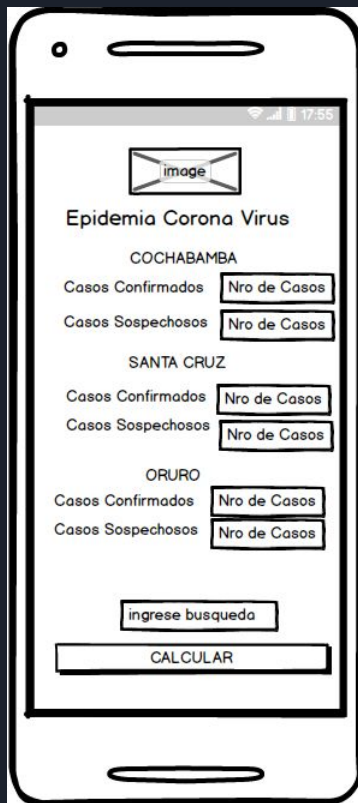
Debe generar un aplicación que permita saber en qué ciudad se tiene más casos confirmados o la cantidad de más casos sospechosos.

Para poder dar solución a este problema se debe de utilizar la tecnología REACT NATIVE utilizando el manejo de componentes, eventos y las buenas prácticas de desarrollo.

Análisis del Problema (Explicación técnica)



Análisis del Problema (Explicación técnica)



image

Epidemia Corona Virus

COCHABAMBA

Casos Confirmados

Casos Sospechosos

SANTA CRUZ

Casos Confirmados

Casos Sospechosos

ORURO

Casos Confirmados

Casos Sospechosos



Defensa Hito 2 - Caso de uso: Epidemia Corona Virus

- Diseñar e implementar el componente CVLogo.
- Diseñar e implementar el componente CVCasos.
- Diseñar e implementar el componente CVCiudad.
- Diseñar e implementar el componente CVSecreen.
- Generar la clase App.JS.



Desarrollo (CVLogo)

```
import React, {Component} from 'react'
import {
  StyleSheet,
  View,
  Text,
  Image
} from 'react-native';
import Logo from '../../images/coronavirus.png'
```

Se importan los
componentes necesarios

Se importa la imagen para el
logo

Desarrollo (CVLogo)



Se especifica que se puede exportar

```
export default class CVLogo extends Component{  
  constructor(props) {  
    super(props);  
  }  
  render() {  
    return(  
      <View style={styles.container}>  
        <Image source={Logo} style={styles.image}/>  
        <Text style={styles.text}>  
          Epidemia Corona Virus  
        </Text>  
      </View>  
    );  
  }  
}
```



Se declara el constructor



Se crea una vista y se le asigna estilos



Se crea una imagen y se asigna el logo, también se le asignan estilos



Se crea un texto se le asignan estilos y se escribe el texto que debe mostrar

Desarrollo (CVLogo)

```
const styles = StyleSheet.create({  
  container:{  
    flex:1,  
    alignItems:'center',  
    justifyContent:'center',  
    backgroundColor:'transparent',  
  },  
  image:{  
    width:80,  
    height:80,  
  },  
  text:{  
    color:'white',  
    fontWeight:'bold',  
    backgroundColor:'transparent',  
    marginTop:10,  
  }  
});
```

Se crean constantes para
asignar estilos

Se crea la constante style de
la colección stylesheet

Desarrollo (CVLogo) LOGRADO





Desarrollo (CVCasos)

```
import React, { Component } from 'react';  
import { StyleSheet, View, TextInput, Text, Image }  
from 'react-native';  
import PropTypes from 'prop-types';  
import Colors from '../..../Config/Colors';
```



Se importan los
componentes necesarios



Se importa los colores

Desarrollo (CVCasos)

```
class CVCasos extends Component {  
  constructor(props) {  
    super(props);  
  }  
  render() {  
    return (  
      <View style={stylesCVCasos.container}>  
        <Text styles={stylesCVCasos.text}>  
          {this.props.titleCasos}  
        </Text>  
      </View>  
    );  
  }  
}
```



Se declara el constructor



Se crea una vista y se le asigna estilos



Se crea un texto y se le asignan estilos



Se le asigna una propiedad variable al
que se puede llamar

Desarrollo (CVCasos)

```
<TextInput
```

→ Se crea un TextInput

```
    onChangeText={this.props.onChangeText}
```

```
    style={stylesCVCasos.textInput}
```

```
    placeholder={this.props.placeholder}
```

```
    secureTextEntry={this.props.secureTextEntry}
```

```
    autoComplete={this.props.autoCorrect}
```

```
    placeholderTextColor={Colors.white}
```

```
    underlineColorAndroid="transparent"
```

```
  />
```

```
</View>
```

```
);
```

```
}
```

```
}
```

→ Se asigna las propiedades variables

Desarrollo (CVCasos)

```
const stylesCVCasos = StyleSheet.create({
  container: {
    flexDirection: 'row',
    flex: 3,
    backgroundColor: Colors.blue,
    alignItems: 'baseline',
    justifyContent: 'space-between',
  },
  textInput: {
    backgroundColor: 'rgba(255, 255, 255, 0.4)',
    alignItems: 'flex-end',
    height: 40,
    borderColor: Colors.silver,
    paddingLeft: 40,
    borderRadius: 15,
    borderBottomWidth: StyleSheet.hairlineWidth,
    marginBottom: 5,
  },
});
```

Se crea la constante style de la colección stylesheet

Se crean constantes para asignar estilos

```
text: {
  alignItems: 'flex-start',
  height: 40,
},
});
```

Desarrollo (CVCasos)

```
CVCasos.propTypes = {  
  placeholder: PropTypes.string.isRequired,  
  autoCorrect: PropTypes.bool,  
  secureTextEntry: PropTypes.bool  
};
```

Se asigna las propiedades a la clase

Se asigna las propiedades a
exportar de la clase

```
export default CVCasos;
```

Se especifica que se puede
exportar



Desarrollo (CVCasos) LOGRADO



Desarrollo (CVCiudad)

```
import React, { Component } from 'react';  
import PropTypes from 'prop-types';  
import { StyleSheet, View, TouchableOpacity, Text }  
from 'react-native';
```

```
import CVCasos from  
'../../Components/login/CVCasos';
```

```
import Constants from '../../Config/Constants';  
import Colors from '../../Config/Colors';
```

Se importan los
componentes necesarios

Se importa la clase que se va
a utilizar

Se importan las constantes

Se importa los colores

Desarrollo (CVCiudad)

Se especifica que se puede exportar

```
export default class CVCiudad extends Component {
```

```
  constructor(props) {
```

```
    super(props);
```

```
  }
```

```
  render() {
```

```
    return (
```

```
      <View style={stylesCVCiudad.container}>
```

```
        <Text style={stylesCVCiudad.Text}>
```

```
          {this.props.titleCiudad}
```

```
        </Text>
```

```
        <CVCasos
```

```
          titleCasos={this.props.titleConfirmados}
```

```
          onChangeText={this.props.onChangeTextConfirmados}
```

```
          placeholder={this.props.confirmados}
```

```
          secureTextEntry={this.props.secureTextEntry}
```

```
          autoCorrect={this.props.autoCorrect}>
```

```
        </CVCasos>
```

Se declara el constructor

Se crea una vista y se le asigna estilos

Se crea un texto y se le asignan estilos

Se le asigna una propiedad variable al que se puede llamar

Se crea la variable exportada anteriormente en la clase CVCasos

Se asigna las propiedades variables exportadas de CVCasos

Desarrollo (CVCiudad)

```
        <CVCasos
            titleCasos={this.props.titleSospechosos}
            onChangeText={this.props.onChangeTextSospechosos}
            placeholder={this.props.sospechosos}
            secureTextEntry={this.props.secureTextEntry}
            autoCorrect={this.props.autoCorrect}>
        </CVCasos>
    </View>
    );
}
```

Se crea otra variable exportada anteriormente en la clase CVCasos

Se asigna las propiedades variables

Desarrollo (CVCiudad)

```
const stylesCVCiudad = StyleSheet.create({
  container: {
    flex: 4,
    backgroundColor: Colors.blue,
    alignItems: 'baseline',
    justifyContent: 'space-between',
  },
  text: {
    alignItems: 'flex-start',
    height: 40,
  },
  form: {
    flex: 4,
    justifyContent: 'center',
    width: '100%',
  },
});
```


Se crea la constante style de la colección stylesheet

Se crean constantes para asignar estilos




Desarrollo (CVCiudad)

```
CVCiudad.propTypes = {  
  titleSospechosos: PropTypes.string.isRequired,  
  titleConfirmados: PropTypes.string.isRequired,  
  titleCiudad: PropTypes.string.isRequired,  
  confirmados: PropTypes.string.isRequired,  
  sospechosos: PropTypes.string.isRequired,  
  autoCorrect: PropTypes.bool,  
  secureTextEntry: PropTypes.bool  
};
```



Se asigna las propiedades exportables a la clase



Se asigna las propiedades a exportar de la clase

Desarrollo (CVCiudad) LOGRADO

COCHABAMBA

Confirmados

4

Sospechosos

5

SANTA CRUZ

Confirmados

3

Sospechosos

6

ORURO

Confirmados

8

Sospechosos

2

Desarrollo (CVScreen)

Se importan las
clase que se van a
utilizar

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import { StyleSheet, View, TouchableOpacity,
TextInput } from 'react-native';
import ButtonLogin from
'../../Components/login/button';
import CVCiudad from
'../../Components/login/CVCiudad';
import CVLogo from '../../Components/login/CVLogo';

import Constants from '../../Config/Constants';
import Colors from '../../Config/Colors';
```

Se importan los
componentes
necesarios

Se importan las constantes

Se importa los colores

Desarrollo (CVScreen)

Se especifica que se puede exportar

```
export default class CVScreen extends Component {
```

```
  constructor(props) {
```

```
    super(props);
```

```
    this.state = {
```

```
      busqueda: '',
```

```
      sospechososCocha: '',
```

```
      confirmadosCocha: '',
```

```
      sospechososSanta: '',
```

```
      confirmadosSanta: '',
```

```
      sospechososOruro: '',
```

```
      confirmadosOruro: ''
```

```
    };
```

Se declara el constructor

Se asigna propiedades
variables propias de clase
con un valor

Desarrollo (CVScreen)

```
this.state
  this._onPress = this._onPress.bind(this);
  this._onChangeTextConfirmadosCocha =
this._onChangeTextConfirmadosCocha.bind(this);
  this._onChangeTextSopechososCocha =
this._onChangeTextSopechososCocha.bind(this);
  this._onChangeTextConfirmadosSanta =
this._onChangeTextConfirmadosSanta.bind(this);
  this._onChangeTextSopechososSanta =
this._onChangeTextSopechososSanta.bind(this);
  this._onChangeTextConfirmadosOruro =
this._onChangeTextConfirmadosOruro.bind(this);
  this._onChangeTextSopechososOruro =
this._onChangeTextSopechososOruro.bind(this);
  this._onChangeText =
this._onChangeText.bind(this);
}
```

Se asigna acciones

Desarrollo (CVScreen)

```
_onPress() {  
    console.log('BUTTON PRESSED!!');  
    if (this.state.búsqueda == 'sospechosos') {  
        console.log(this.state.búsqueda, Math.max(  
            parseInt(this.state.sospechososCocha, 10),  
            parseInt(this.state.sospechososSanta, 10),  
            parseInt(this.state.sospechososOruro, 10)));  
    } else if (this.state.búsqueda == 'confirmados'){  
        console.log(this.state.búsqueda, Math.max(  
            parseInt(this.state.confirmadosCocha, 10),  
            parseInt(this.state.confirmadosSanta, 10),  
            parseInt(this.state.confirmadosOruro, 10)));  
    }  
}
```

Se crea la función _OnPress()

Se condiciona para comparar el texto de búsqueda

Con la función Math.Max se obtiene el mayor número

Se convierte la variable obtenida en un número de base 10, decimal

Lo mismo pero con variables diferentes y con condición diferente

Desarrollo (CVScreen)

El valor obtenido con variable asignada

```
_onChangeTextConfirmadosCocha (confirmadosCocha) {  
    this.setState({  
        confirmadosCocha: confirmadosCocha  
    });  
}
```

Se crea la función `onChangeTextConfirmadosCocha()`

Se cambia el valor de la variable por el valor obtenido

```
_onChangeTextSopechososCocha (sospechososCocha) {  
}  
_onChangeTextConfirmadosSanta (confirmadosSanta) {  
}  
_onChangeTextSopechososSanta (sospechososSanta) {  
}  
_onChangeTextConfirmadosOruro (confirmadosOruro) {  
}  
_onChangeTextSopechososOruro (sospechososOruro) {  
}  
_onChangeText (busqueda) {  
}
```

Lo mismo pero con variables diferentes

Desarrollo (CVScreen)

Se especifica que se puede exportar

```
export default class CVCiudad extends Component {
```

```
  constructor(props) {
```

```
    super(props);
```

```
  }
```

```
  render() {
```

```
    return (
```

```
      <View style={stylesCVCiudad.container}>
```

```
        <Text style={stylesCVCiudad.Text}>
```

```
          {this.props.titleCiudad}
```

```
        </Text>
```

```
        <CVCasos
```

```
          titleCasos={this.props.titleConfirmados}
```

```
          onChangeText={this.props.onChangeTextConfirmados}
```

```
          placeholder={this.props.confirmados}
```

```
          secureTextEntry={this.props.secureTextEntry}
```

```
          autoCorrect={this.props.autoCorrect}>
```

```
        </CVCasos>
```

Se declara el constructor

Se crea una vista y se le asigna estilos

Se crea un texto y se le asignan estilos

Se le asigna una propiedad variable al que se puede llamar

Se crea la variable exportada anteriormente en la clase CVCasos

Se asigna las propiedades variables exportadas de CVCasos

Desarrollo (CVScreen)

A cada variable se le agrega un valor, ya sea una constante o una función

```
render() {  
  return (  
    <View style={stylesCVScreen.container}>  
      <CVLogo style={stylesCVScreen.logo} />  
      <View style={stylesCVScreen.form}>  
        <CVCiudad
```

Se crea la variable exportada anteriormente en la clase CVCiudad

A las variables del textInput se le asigna una función

```
        titleConfirmados={Constants.CONFIRMADOS}  
        titleSospechosos={Constants.SOSPECHOSOS}  
        titleCiudad={Constants.COCHABAMBA}  
        onChangeTextConfirmados={this._onChangeTextConfirmadosCocha}  
        onChangeTextSospechosos={this._onChangeTextSopechososCocha}  
        confirmados={Constants.TEXTO_CASOS}  
        sospechosos={Constants.TEXTO_CASOS}  
        securetextEntry={false}  
        autoComplete={false}>  
      </CVCiudad>
```

Se crea una vista y se le asigna estilos

Se crea la variable exportada anteriormente en la clase CVLogo y se le asigna un estilo

Se crea otra vista y se le asigna estilos

Se asigna las propiedades variables exportadas de CVCasos



Desarrollo (CVScreen)

```
<CVCiudad  
    titleCiudad={Constants.SANTA_CRUZ}  
    autoCorrect={false}>  
</CVCiudad>  
  
<CVCiudad  
    titleCiudad={Constants.ORURO}  
    autoCorrect={false}>  
</CVCiudad>
```

Para cada ciudad se replican
las mismas características
con excepción del título

Desarrollo (CVScreen)

A cada variable se le agrega un valor, ya sea una constante o una función

Se crea una vista y se le asigna estilos

Se crea un textInput y se le asigna estilos

Se asigna una función a la variable

Se crea un botón con la clase button que se importó inicialmente

A la acción de presionar botón se le asigna una función

Se cierran todos los componentes anteriores

```
<View>
  <TextInput
    onChangeText={this._onChangeText}
    style={styles.CVScreen.textInput}
    placeholder={Constants.INGRESE_BUSQUEDA}
    value={this.state.value}
    placeholderTextColor={Colors.white}
    underlineColorAndroid="transparent" />
  <ButtonLogin
    onPress={this._onPress}
    titleButton={Constants.TITLE_BUTTON}>
  </ButtonLogin>
</View>
</View>
</View>
);
}
```


Desarrollo (CVScreen)

```
const stylesCVScreen = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: Colors.blue,
    alignItems: 'center',
    justifyContent: 'space-between',
  },
  textInput: {
    backgroundColor: 'rgba(255, 255, 255, 0.4)',
    alignItems: 'flex-end',
    height: 40,
    borderColor: Colors.silver,
    paddingLeft: 40,
    borderRadius: 15,
    borderBottomWidth: StyleSheet.hairlineWidth,
    marginBottom: 5,
  },
});
```

Se crea la constante style de la colección stylesheet

Se crean constantes para asignar estilos

```
logo: {
  flex: 1,
  width: '100%',
  resizeMode: 'stretch',
  alignSelf: 'center',
},
form: {
  flex: 4,
  justifyContent: 'center',
  width: '80%',
},
});
```



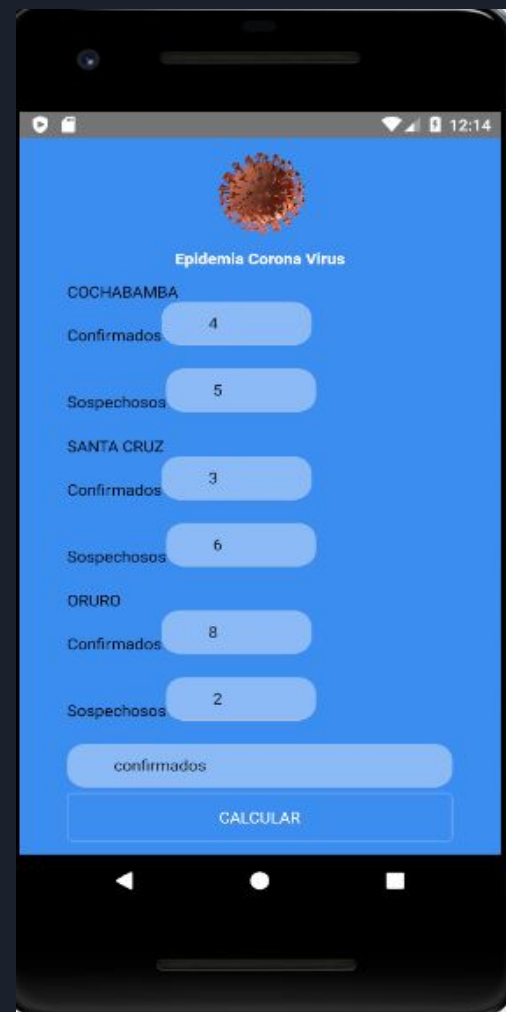
Desarrollo (CVScreen)

```
CVScreen.propTypes = {  
  
};
```



Se asigna las propiedades exportables a la clase

Desarrollo (CVScreen) LOGRADO



Desarrollo (App.js)

Se importan la
clase que se va a
utilizar



```
import React from 'react';  
  
import CVScreen from '../src/View/login/CVScreen';
```



Se importan los
componentes necesarios

```
const App: () => React$Node = () => {  
  return (  
    <CVScreen/>  
  );  
};
```



Se crea un componente con la clase importada

```
export default App;
```



Se exporta

Desarrollo (App.js funcionamiento) LOGRADO

Datos ingresados
Cochabamba

Confirmados 4

Sospechosos 5

Santa Cruz

Confirmados 3

Sospechosos 6

Oruro

Confirmados 8

Sospechosos 2

Resultado Confirmados

```
raquel — Metro — node • launchPackager.command — 80x24
#####
#####
#####
#####
#####
#####

Welcome to React Native!
Learn once, write anywhere

To reload the app press "r"
To open developer menu press "d"

[Sat Apr 11 2020 13:49:30.392] BUNDLE ./index.js
[Sat Apr 11 2020 13:50:58.603] LOG Running "DefensaHito2" with {"rootTag": 1}
[Sat Apr 11 2020 15:22:09.584] BUNDLE ./index.js
[Sat Apr 11 2020 15:22:16.801] LOG Running "DefensaHito2" with {"rootTag": 1}
[Sat Apr 11 2020 15:23:39.506] LOG BUTTON PRESSED!!
[Sat Apr 11 2020 15:24:52.693] LOG BUTTON PRESSED!!
[Sat Apr 11 2020 15:24:52.698] LOG confirmados 8
```

Resultado Sospechosos

```
raquel — Metro — node • launchPackager.command — 80x24
#####
#####

Welcome to React Native!
Learn once, write anywhere

To reload the app press "r"
To open developer menu press "d"

[Sat Apr 11 2020 13:49:30.392] BUNDLE ./index.js
[Sat Apr 11 2020 13:50:58.603] LOG Running "DefensaHito2" with {"rootTag": 1}
[Sat Apr 11 2020 15:22:09.584] BUNDLE ./index.js
[Sat Apr 11 2020 15:22:16.801] LOG Running "DefensaHito2" with {"rootTag": 1}
[Sat Apr 11 2020 15:23:39.506] LOG BUTTON PRESSED!!
[Sat Apr 11 2020 15:24:52.693] LOG BUTTON PRESSED!!
[Sat Apr 11 2020 15:24:52.698] LOG confirmados 8
[Sat Apr 11 2020 15:28:40.655] LOG BUTTON PRESSED!!
[Sat Apr 11 2020 15:28:40.658] LOG sospechosos 6
```