

Solucionario de Examen del hito 2 de PDM-1ra gestión del 2020

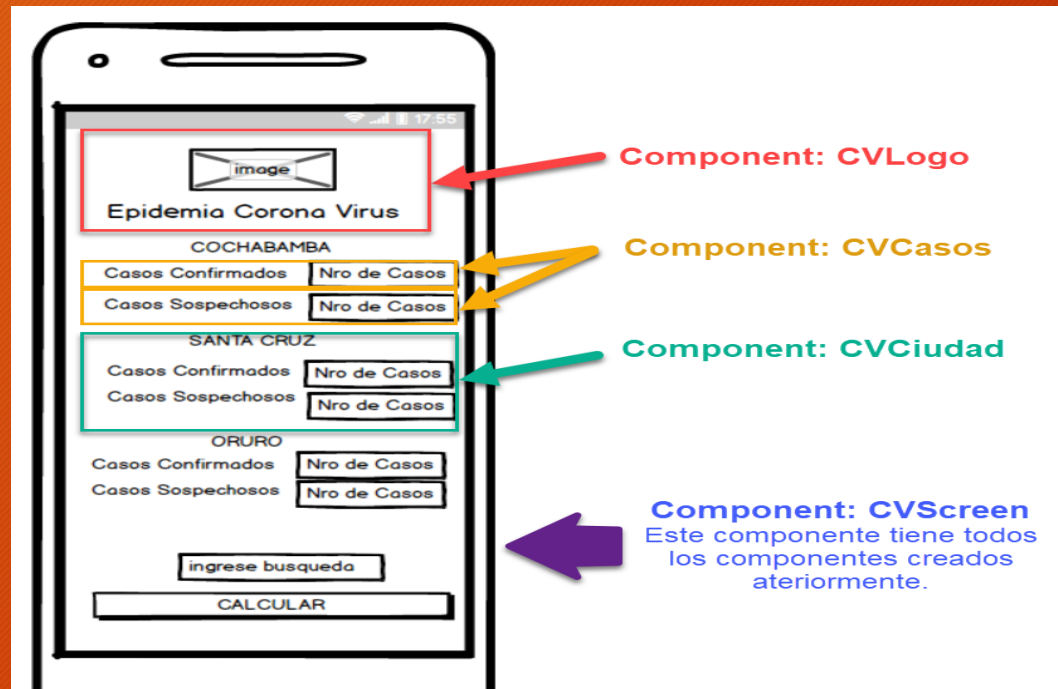
Estudiante: Glenn Julian Castro Duarte

Definición:

- 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 que ciudad se tiene mas casos confirmados o la cantidad de mas casos sospechosos.
- Para poder dar solucion a este problema se debe de utlizar la tecnologia REACT NATIVE utilizando el manejo de componenetes, eventos y las buenas practicas de desarrollo.

Definición:

- La evaluación consiste en la generación de 4 componentes, CVLogo, CVCasos, CVCiudad y CVScreen; todos con la temática de Corona Virus.



Primera pregunta: generar el componente CVLogo



Componente CVLogo:

- Se debe generar un componente que se capaz de usar una imagen, previamente cargada, y un texto, previamente definido, para producir el resultado visualizado anteriormente
- La imagen podrá ser de extencion .jpg o .png
- El texto deberá ser “Epidemia Corona Virus”
- La imagen deberá escatar relaiconada con la temat

Solución de Componente CVLogo : Librerías usadas

- `import React, {Component} from 'react'`
- `import { StyleSheet, View, Text, Image} from 'react-native';`
- `import Logo from '../..../img/virus.jpg';`

Solución de Componente CVLogo : Clase Logo

- export default class logo 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>
-); } }

Solución de Componente CVLogo :CSS Usado

```
• const styles = StyleSheet.create({  
•   container:{  
•     flex:1,  
•     alignItems:'center',  
•     justifyContent:'center',  
•   },  image:{  
•     width:80,  
•     height:80,  
•   },  text:{  
•     color:'white',  
•     fontWeight:'bold',  
•     backgroundColor:'transparent',  
•     marginTop:20,  
•   } } )
```


Componente CVCasos:

Casos
Confirmados

1

Casos
sospechosos

2

Componente CVCasos:

- Se debe generar un componente manejando el componente TextInput, que será capaz de recibir datos introducidos por el usuario y un texto que funcionara como enunciado
- El placeholder o Texto de pista, deberá indicar para ambos inputs el texto “”Nro de casos”
- El texto que Acompañara al primer Input será “Caso Confirmado” seguido por el segundo que será “Caso Sospechoso”
- Debera prepararse la recepción de variables que serán definidas posteriormente en los siguientes componentes como ser: typeCase, placeholder, onChangeText, secureTextEntry, autoCorrect

Solución de Componente CVCaso : Librerías usadas

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

Solución de Componente CVCaso :Const CVcaso

- `const CVcasos=(props)=>{`
- `const {typeCase, placeholder, onChangeText, secureTextEntry, autoCorrect}=props;`
- `return(`
- `<View>`
- `<Text style={stylesTextInput.inlineImg}>{typeCase} </Text>`
- `<TextInput`
- `style={stylesTextInput.textInput}`
- `onChangeText={onChangeText}`
- `selectionColor={Colors.blue}`
- `placeholder={placeholder}`
- `secureTextEntry={secureTextEntry}`
- `autoCorrect={autoCorrect}`
- `placeholderTextColor={"#ffffff"}`
- `underlineColorAndroid="transparent"`
- `/>`
- `</View>`
- `});`

Solución de Componente CVCaso : PropTypes requeridos

- CVcasos.propTypes={
- onChangeText: PropTypes.func.isRequired,
- placeholder: PropTypes.string.isRequired,
- autoCorrect: PropTypes.bool,
- secureTextEntry: PropTypes.bool,
- };

Solución de Componente CVCase :CSS utilizado

- `const stylesTextInput = StyleSheet.create({`
- `textInput: {`
- `backgroundColor: 'rgba(255, 255, 255, 0.4)',`
- `alignItems: 'center',`
- `height: 40,`
- `borderColor: Colors.red,`
- `paddingLeft: 40,`
- `borderRadius: 15,`
- `borderBottomWidth: StyleSheet.hairlineWidth,`
- `marginBottom: 15,`
- `marginLeft: 50,`
- `}, inlinedImg: {`
- `position: 'absolute',`
- `zIndex: 99,`
- `width: 100,`
- `height: 100,`
- `left: -40,`
- `top: 0,`
- `color: 'white',`
- `},`
- `});`

Componente CVCiudad:

Departamento de Cochabamba

Casos
Confirmados

1

Casos
sospechosos

2

Departamento de Santa Cruz

Casos
Confirmados

7

Casos
sospechosos

8

Departamento de Oruro

Casos
Confirmados

3

Casos
sospechosos

4

Componente CVCiudad:

- Se deberá generar un componente que haga uso del componente Cvcasos, añadiéndole el nombre de una ciudad que se desea representar
- Como se hizo en el anterior componente, deberá definirse las variables de recepción de este componente que serán: ciudad,onChangeTextConf,onChangeTextSosp
- Deberá también dar seguimiento y valor a las variables definidas en el componente CVCasos

Solución de Componente CVCiudad : Librerías usadas

- `import React from 'react';`
- `import {StyleSheet,View,Text} from 'react-native';`
- `import Colors from '../..../Config/Colors';`
- `import CVcasos from '../..../Components/corona/CVcasosing';`
- `import Constants from '../..../Config/constants';`

Solución de Componente CVCiudad : Const CVCiudad

```
• const CVciudad=(props)=> {  
•   const {ciudad,onChangeTextConf,onChangeTextSosp}=props;  
•   return(  
•     <View>  
•       <Text>{ciudad}</Text>  
•       <Cvcasos      onChangeText={onChangeTextConf}      placeholder={Constants.nro}  
•       autoComplete={false}      secureTextEntry={false}      typeCase={Constants.CONFIRMA}  
•       />  
•       <CVcasos      onChangeText={onChangeTextSosp}      placeholder={Constants.nro}  
•       autoComplete={false}      secureTextEntry={false}      typeCase={Constants.SOSPECHA}  
•       />  
•     </View>  
•   );  
• };
```

Solución de Componente CVCiudad :CSS utilizado

```
• const styles = StyleSheet.create({  
•   text:{  
•     position: 'absolute',  
•     fontWeight:'bold',  
•     color: Colors.black,  
•     },  
•  
•   });
```

Componente CVScreen:

VivamosDesdeCasa 16:15



Epidemia Corona Virus

Departamento de Cochabamba

Casos Confirmados

Casos sospechosos

Departamento de Santa Cruz

Casos Confirmados

Casos sospechosos

Departamento de Oruro

Casos Confirmados

Casos sospechosos

Calcular

```
npm
>
NaN
boton calcular
Object {
  "confCB": "1",
  "confOR": "3",
  "confSC": "7",
  "search": "Conf",
  "sospCB": "2",
  "sospOR": "4",
  "sospSC": "8",
}
7
boton calcular
Object {
  "confCB": "1",
  "confOR": "3",
  "confSC": "7",
  "search": "Sosp",
  "sospCB": "2",
  "sospOR": "4",
  "sospSC": "8",
}
8
```


Componente CVScreen:

- Se deberá generar un componente que haga uso de los componentes previamente generados para darle funcionalidad de comparar los valores datos, ubicados en los TextInput, y generar un resultado
- Se manejará el uso de estados y funciones para las variables que cambiarán cada que se cambie el texto en los TextInput
- Haciendo uso de un “Search”Input se definirá el tipo de búsqueda que será “confirmados” o “sospechosos”
- También se empleará un botón que ejecute la función de comparación de los valores para emitir un resultado

Solución de Componente CVCiudad : Librerías usadas

- `import React, { Component } from 'react';`
- `import { StyleSheet, View, TextInput } from 'react-native';`
- `import Button from '../..Components/corona/Button';`
- `import CVciudad from '../..Components/corona/CVciudad';`
- `import LogoLogin from '../..Components/corona/CVlogo';`
- `import Constants from '../..Config/constants';`
- `import Colors from '../..Config/Colors';`
- `import ciudades from '../..Config/ciudades';`

Solución de Componente CVCiudad: Bind y declaración de variables de calculo

```
• constructor(props) {  
•   super(props);  
•   this.state = {  
•     confCB: 0,      sospCB: 0,      confSC: 0,      sospSC: 0,      confOR: 0,      sospOR: 0,  
•     search: ''  
•   };  
•   this._onChangeTextConfCB = this._onChangeTextConfCB.bind(this);  
•   this._onChangeTextSospCB = this._onChangeTextSospCB.bind(this);  
•   this._onChangeTextConfSC = this._onChangeTextConfSC.bind(this);  
•   this._onChangeTextSospSC = this._onChangeTextSospSC.bind(this);  
•   this._onChangeTextConfOR = this._onChangeTextConfOR.bind(this);  
•   this._onChangeTextSospOR = this._onChangeTextSospOR.bind(this);  
•   this._onChangeTextInputSearch = this._onChangeTextInputSearch.bind(this);  
•   this._onPressCalcular= this._onPressCalcular.bind(this);  
• }
```


Solución de Componente CVCiudad: funciones de asignación de variable

```
• _onChangeTextConfCB(confCB) {      this.setState({      confCB: confCB      });  
•   }  
• _onChangeTextSospCB(sospCB) {      this.setState({      sospCB: sospCB      });  
•   }  
• _onChangeTextConfSC(confSC) {      this.setState({      confSC: confSC      });  
•   }  
• _onChangeTextSospSC(sospSC) {      this.setState({      sospSC: sospSC      });  
•   }  
• _onChangeTextConfOR(confOR) {      this.setState({      confOR: confOR      });  
•   }  
• _onChangeTextSospOR(sospOR) {      this.setState({      sospOR: sospOR      });  
•   }  
• _onChangeTextInputSearch(search){    this.setState({      search: search    });  
•   }
```

Solución de Componente CVCiudad: Función de comparación de valores

```
• _onPressCalcular(){      let n1CB, n2SC, n3OR, busqueda;
•   if ( this.state.search === 'Conf'){
•       n1CB= parseInt(this.state.confCB);
•       n2SC= parseInt(this.state.confSC);
•       n3OR= parseInt(this.state.confOR);
•   }
•
•   if (this.state.search === 'Sosp'){
•       n1CB=parseInt(this.state.sospCB);
•       n2SC= parseInt(this.state.sospSC);
•       n3OR= parseInt(this.state.sospSC);
•   }
•
•   let mayor = Math.max(n1CB, n2SC, n3OR);
•   console.log('boton calcular');
•   console.log(this.state);
•   console.log(mayor);
• }
```


Solución de Componente CVCiudad: Propiedad render dentro de la class CVScreen

```
• render() {      return (  
•      <View style={styles.container}>          <LogoLogin style={styles.logo} />  
•      <View>  
•          <CVciudad ciudad={ciudades.COCHA}          onChangeTextConf={this._onChangeTextConfCB}          onChangeTextSosp={this._onChangeTextSospCB}>  
•          </CVciudad>  
•          <CVciudad  
•              ciudad={ciudades.SANTA}          onChangeTextConf={this._onChangeTextConfSC}          onChangeTextSosp={this._onChangeTextSospSC}>  
•          </CVciudad>  
•          <CVciudad  
•              ciudad={ciudades.ORURO}          onChangeTextConf={this._onChangeTextConfOR}          onChangeTextSosp={this._onChangeTextSospOR}>  
•          </CVciudad>  
•      </View>  
•      <TextInput  
•          style={styles.textInput}          onChangeText={this._onChangeTextInputSearch}          selectionColor={Colors.blue}          placeholder={Constants.BUSQUEDA}  
•          secureTextEntry={false}          autoCorrect={false}          placeholderTextColor={"#ffffff"}          underlineColorAndroid="transparent" />  
•      <Button          onPress={this._onPressCalcular}          titleButton={Constants.CALCULAR}          />  
•      </View>      ); }
```


Solución de Componente CVCiudad:Propiedad render dentro de la class CVScreen

```
• const styles = StyleSheet.create({
•   container: { flex: 1,      backgroundColor: Colors.red,      alignItems: 'center',      justifyContent:
'space-between',
•   },
•   logo: {      flex: 1,      width: '100%',      resizeMode: 'contain',      alignSelf: 'center',
•   },
•   form: {      flex: 1,      justifyContent: 'center',      width: '80%',      marginTop: 100,
•   },
•   textInput: {
•     backgroundColor: 'rgba(255, 255, 255, 0.4)',      alignItems: 'center',      height: 40,
•     borderColor: Colors.red,      paddingLeft: 40,      borderRadius: 15,
•     borderBottomWidth: StyleSheet.hairlineWidth,      marginBottom: 15,      marginLeft: 50,
•   },
• });
```

App.js:

VivamosDesdeCasa 16:15



Epidemia Corona Virus

Departamento de Cochabamba

Casos Confirmados

Casos sospechosos

Departamento de Santa Cruz

Casos Confirmados

Casos sospechosos

Departamento de Oruro

Casos Confirmados

Casos sospechosos

Calcular

```
NaN
boton calcular
Object {
  "confCB": "1",
  "confOR": "3",
  "confSC": "7",
  "search": "Conf",
  "sospCB": "2",
  "sospOR": "4",
  "sospSC": "8",
}
7
boton calcular
Object {
  "confCB": "1",
  "confOR": "3",
  "confSC": "7",
  "search": "Sosp",
  "sospCB": "2",
  "sospOR": "4",
  "sospSC": "8",
}
8
```

App.js:

- En esta etapa lo único que se hace es importar el componente y verificar el funcionamiento de la aplicación
- Pueden añadirse estilos adicionales o la introducción de variables, pero en nuestro caso eso no será necesario

Solución de Componente App.js: Librerías a usar

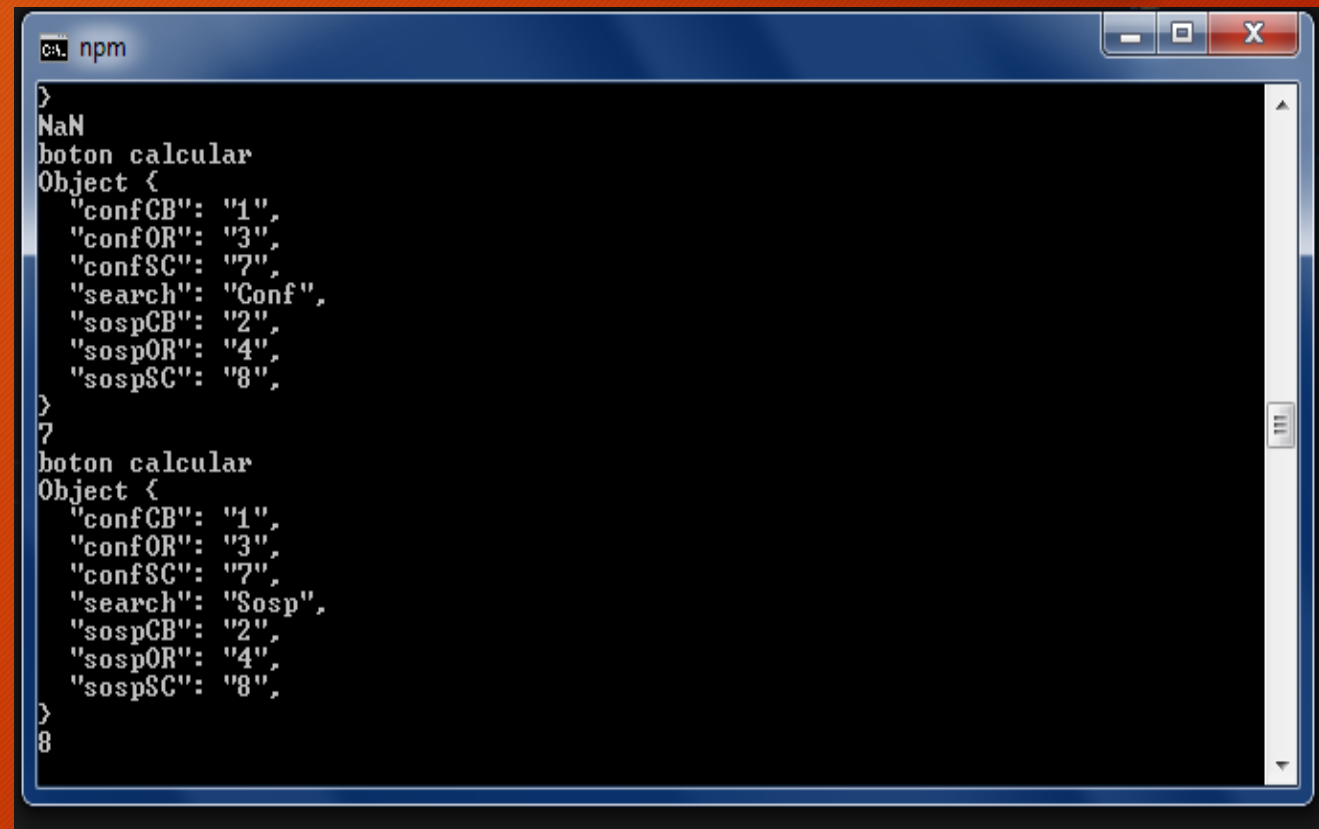
- `import React, { Component } from 'react';`
- `import { SafeAreaView, StyleSheet, ScrollView, View, Text, StatusBar, ImageBackground, Image } from 'react-native';`
- `import CVscreening from './src/Components/corona/CVscreening';`

Solución de Componente App.js: Class App.js

- export default class App extends Component{
- render(){
- return(
- <CVscreening></CVscreening>
-
-);
-
- }
- }

Conclusion:

- Este deberá ser el funcionamiento final del toda la aplicación, siendo capaz de compara cual es la mayor variable ubicada en los TextInput sea confirmado o sospechoso
- Con esto se puede dar por concluida la resolución de la evaluacion



```
ca: npm
>
NaN
boton calcular
Object {
  "confCB": "1",
  "confOR": "3",
  "confSC": "7",
  "search": "Conf",
  "sospCB": "2",
  "sospOR": "4",
  "sospSC": "8",
}
7
boton calcular
Object {
  "confCB": "1",
  "confOR": "3",
  "confSC": "7",
  "search": "Sosp",
  "sospCB": "2",
  "sospOR": "4",
  "sospSC": "8",
}
8
```