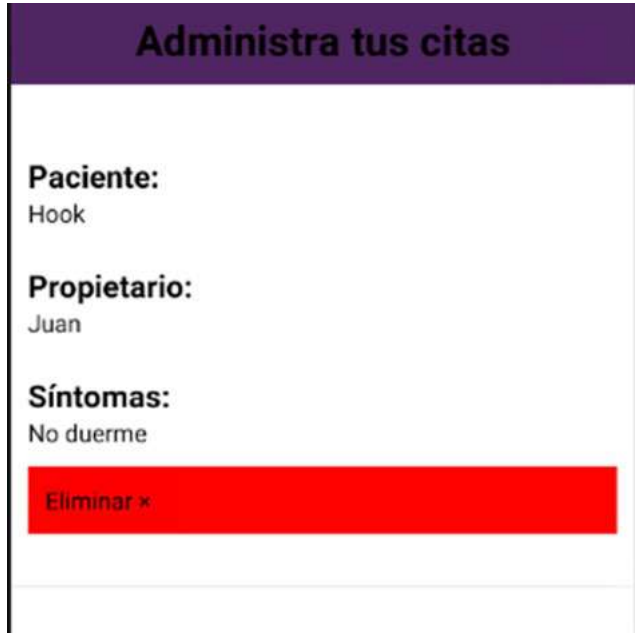
React native
Unidad III

Por: Paulina Solórzano.

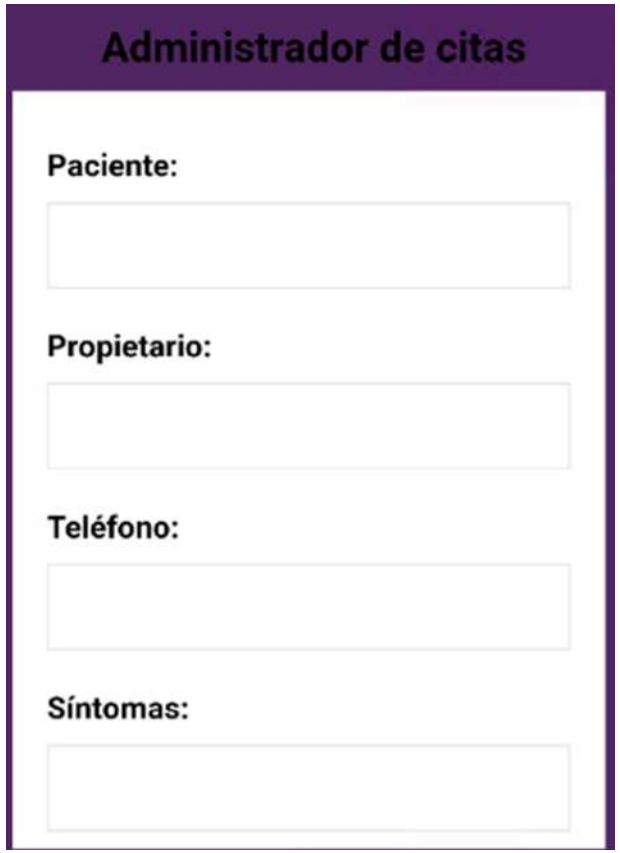
Práctica: Registro de citas de mascotas

1. Realizaremos una práctica que implica el registro de citas para pacientes (mascotas) en un consultorio médico.
2. Esta presentación incluye solo una parte del ejercicio

actividad



1.-Realizaremos una lista de citas con botón de eliminar



2.-Realizaremos un formulario con diferentes tipos de campos

Proyecto de citas

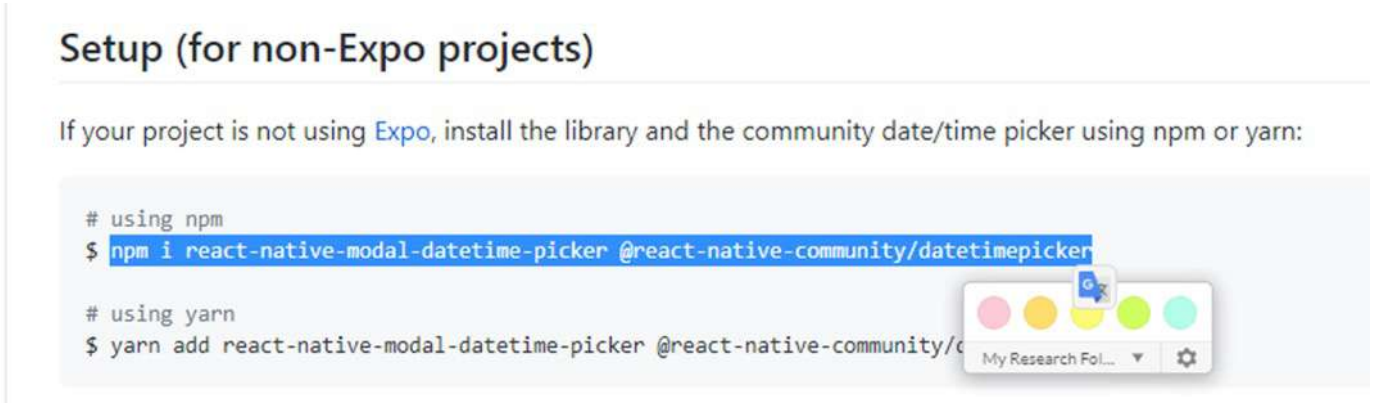
1. Realizaremos una app que lleve como nombre “citas”.
2. Llevará a cabo el registro de citas médicas de mascotas.
3. Puedes usar Snack.expo.io ó usar VisualStudioCode.
4. Utilizaremos un arreglo de objetos de manera estática para poder almacenar los datos (usando el useState).
5. Crearemos nuestros estilos para el fomulario y listado de citas.
6. Crearemos un componente Cita y un componente formulario
7. Inicialmente necesitaremos instalar las librerías que ocuparemos para el proyecto.

Si trabajas en la computadora, las librerías a instalar son:

1.- El datetimepicker (para el manejo de fechas y horas en el formulario de citas). Esta dependencia la puedes descargar de:

https://github.com/mmazzarolo/react-native-modal-datetime-picker

Y usando comando:




```
PS C:\Users\pauli\citas> npm i react-native-modal-datetime-picker @react-native-community/datetimepicker
[...................] | idealTree:citas: sill idealTree buildDeps
```

- La siguiente librería que instalaremos se llama shortid (ésta librería más adelante nos ayudará a generar id únicos para nuevos objetos tipo“cita”

Los comandos son:

npm install shortid

Ó

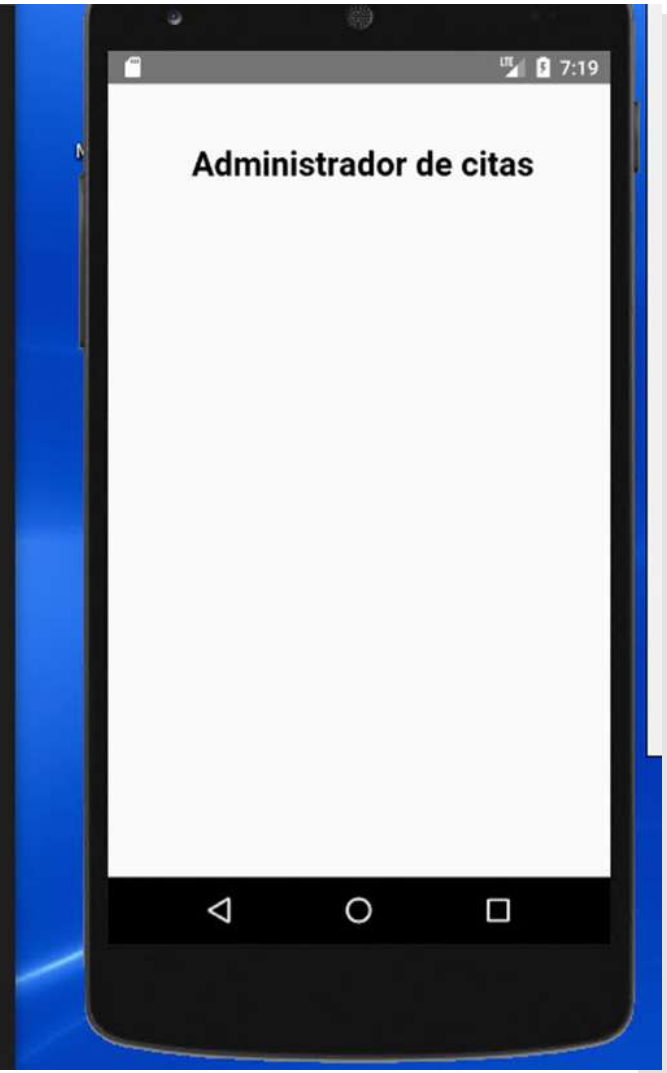
npm i shortid

Ó

yarn add shortid

Haremos estos cambios a nuestra app

```jsx
import React from 'react';
import {
  StyleSheet,
  View,
  Text,
} from 'react-native';


const App = () => {
  return (

    <View>
    <Text style={styles.titulo}> Administrador de citas</Text>
    </View>

  );
};

//Para los estilos crearemos una variable "styles"
const styles = StyleSheet.create({
  titulo:{
    marginTop:40,
    fontSize:24,
    fontWeight:'bold',
    textAlign:'center',
  }
});

export default App;

```

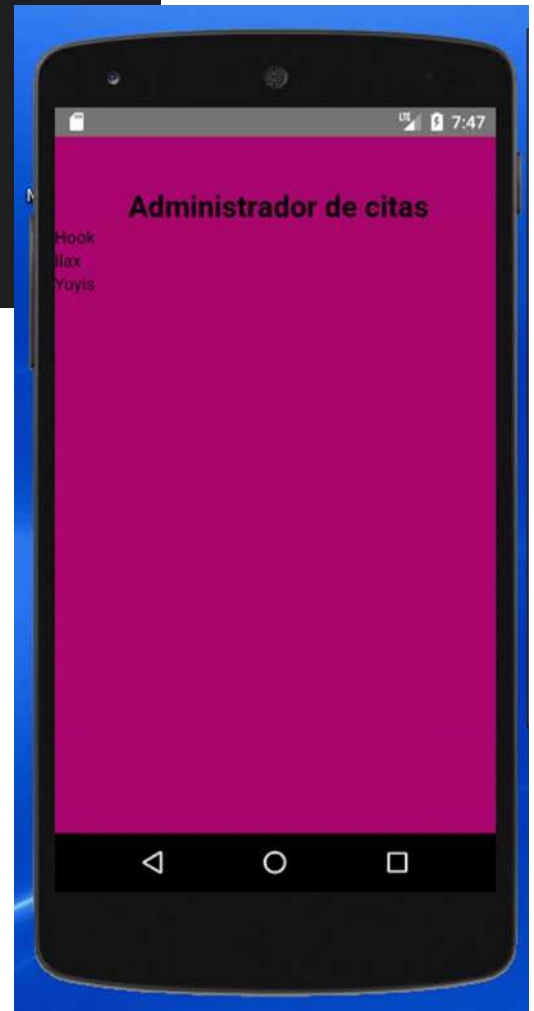


App.js > styles > contenedor

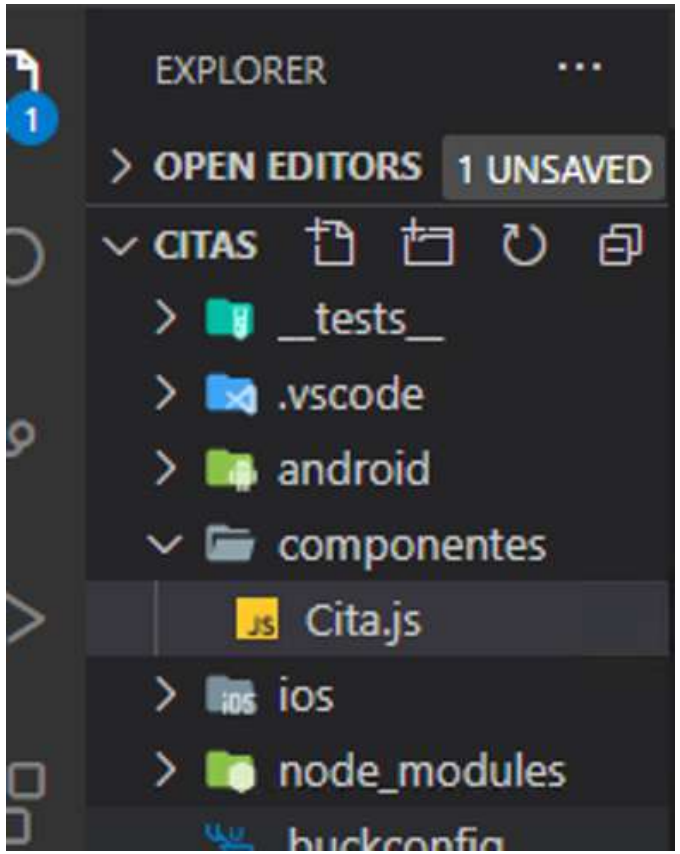
```
1   import React, {useState} from 'react';
2   import { StyleSheet,View,Text,FlatList
3   } from 'react-native';
4
5   const App = () => {
6
7     const[citas, setCitas]=useState([
8       { id:"1", paciente:"Hook", propetario:"Juan", sintomas:"No duerme"},
9       { id:"1", paciente:"Ilax", propetario:"Mario", sintomas:"No come"},
10      { id:"1", paciente:"Yuyis", propetario:"Javier", sintomas:"es muy acelerada"},
11    ]);
12    return (
13      <View style={styles.contenedor}>
14      <Text style={styles.titulo}> Administrador de citas</Text>
15      <FlatList
16      data={citas}
17      renderItem={({item})=>(
18        <View>
19          <Text>{item.paciente}</Text>
20        </View>
21      )}
22      KeyExtractor={cita=>cita.id}
23      />
24      </View>
25    );
26  };
```

Terminaremos haciendo los estilos…

```
28  //Para los estilos crearemos una variable "styles"
29  const styles = StyleSheet.create({
30  contenedor:{
31    backgroundColor:'#AA076B',
32    flex:1  //Esto permite crecer el espacio del contenedor en un 100%
33  },
34
35    titulo:{
36      marginTop:40,
37      fontSize:24,
38      fontWeight:'bold',
39      textAlign:'center',
40    }
41  });
42
43  export default App;
```



Ahora crear una carpeta para los componentes



Primero lo haremos con la función ".map" que funciona como el foreach. Y posteriormente lo cambiaremos por un Flatlist

App.js > App

```
1   import React, {useState} from 'react';
2   import { StyleSheet,View,Text,FlatList
3   } from 'react-native';
4
5   import Cita from './componentes/Cita';
6   const App = () => {
7
8     const[citas, setCitas]=useState([
9       { id:"1", paciente:"Hook", propietario:"Juan", sintomas:"No duerme"},
10      { id:"1", paciente:"Ilax", propietario:"Mario", sintomas:"No come"},
11      { id:"1", paciente:"Yuyis", propietario:"Javier", sintomas:"es muy acelerada"},
12    ]);
13    return (
14      <View style={styles.contenedor}>
15      <Text style={styles.titulo}> Administrador de citas</Text>
16
17
18      {citas.map(cita=>(
19      <View>
20        <Text>{cita.paciente}</Text>
21      </View>
22      ))}
23
24      </View>
25    );
26  };
```

App.js

Estilos que utilizaremos

```
25  //Para los estilos crearemos una variable "styles"
26  const styles = StyleSheet.create({
27  contenedor:{
28    backgroundColor:'#AA076B',
29    flex:1  //Esto permite crecer el espacio del contenedor en un 100%
30
31  },
32
33    titulo:{
34      marginTop:40,
35      fontSize:24,
36      fontWeight:'bold',
37      textAlign:'center',
38    }
39  });
40
41  export default App;
42
```

App.js

Ahora lo haremos con un Flatlist, pero también como puedes observar, utilizaremos el componente Cita.

```
1  import React, {useState} from 'react';
2  import { StyleSheet,View,Text,FlatList
3  } from 'react-native';
4
5  import Cita from './componentes/Cita';
6  const App = () => {
7
8    const[citas, setCitas]=useState([
9      { id:"1", paciente:"Hook", propietario:"Juan", sintomas:"No duerme"},
10     { id:"1", paciente:"Ilax", propietario:"Mario", sintomas:"No come"},
11     { id:"1", paciente:"Yuyis", propietario:"Javier", sintomas:"es muy acelerada"},
12   ]);
13   return (
14     <View style={styles.contenedor}>
15     <Text style={styles.titulo}> Administrador de citas</Text>
16     <FlatList
17     data={citas}
18     renderItem={({item})=> <Cita item={item} />}
19     KeyExtractor={cita=>cita.id}
20     />
21     </View>
22   );
23 };
24
```

Cita.js

Puedes guardar para que veas como se ejecuta, y posteriormente
Le daremos estilos

App.js Cita.js

componentes > Cita.js > default

```
1   import React from 'react';
2   import {Text, StyleSheet, View} from 'react-native';
3
4   const Cita=({item})=>{
5       return (
6       <View>
7           <View>
8               <Text>Paciente:</Text>
9               <Text>{item.paciente}</Text>
10          </View>
11          <View>
12              <Text>Propietario:</Text>
13              <Text>{item.propietario}</Text>
14          </View>
15          <View>
16              <Text>Síntomas:</Text>
17              <Text>{item.sintomas}</Text>
18          </View>
19      </View>
20      )
21  }
22
23  export default Cita;
```

Cita.js

```jsx
import React from 'react';
import {Text, StyleSheet, View,Button} from 'react-native';

const Cita=({item})=>{
    return (
    <View style={styles.cita}>
        <View>
            <Text style={styles.label}>Paciente:</Text>
            <Text>{item.paciente}</Text>
        </View>
        <View>
            <Text  style={styles.label}>Propietario:</Text>
            <Text>{item.propietario}</Text>
        </View>
        <View>
            <Text  style={styles.label}>Síntomas:</Text>
            <Text>{item.sintomas}</Text>
        </View>
    </View>
    )
}
```

```jsx
const styles=StyleSheet.create({
    cita:{
        backgroundColor:'#FFF',
        borderBottomColor:'#e1e1e1',
        borderStyle:"solid",
        borderBottomWidth:1,
        paddingVertical:20,
        paddingHorizontal:10,
    },
    label:{
        fontWeight:'bold',
        fontSize:18,
        marginTop:20
    },
    texto:{

    }
})

export default Cita;
```


12:18
Administrador de citas
Paciente:
Hook
Propietario:
Juan
Síntomas:
No duerme
Paciente:
Ilax
Propietario:
Mario
Síntomas:
No come

Mientras tanto, en App.js haremos la función “eliminarPaciente”, para hacer un filtro y que nos esconda las citas que hayamos eliminado con el botón.

```
5   import Cita from './componentes/Cita';
6   const App = () => {
7
8     const[citas, setCitas]=useState([
9       { id:"1", paciente:"Hook", propietario:"Juan", sintomas:"No duerme"},
10      { id:"2", paciente:"Ilax", propietario:"Mario", sintomas:"No come"},
11      { id:"3", paciente:"Yuyis", propietario:"Javier", sintomas:"es muy acelerada"},
12    ]);
13
14  const eliminarPaciente=id=>{
15    setCitas((citasActuales)=>{
16    return citasActuales.filter(cita=>cita.id !== id);
17    })
18  }
19
20    return (
21      <View style={styles.contenedor}>
22      <Text style={styles.titulo}> Administrador de citas</Text>
23
24      <FlatList
25      data={citas}
26      renderItem={({item})=> <Cita item={item} eliminarPaciente={eliminarPaciente} />}
27      KeyExtractor={citas=>cita.id}
28      />
29
30      </View>
31    );
32  };
```

Cita.js

En el componente Cita, tendremos que agregar el botón de elminar.
Los comentarios a desplegarse en consola puedes eliminarlos de snack.expo.io (si estas usando esta página)

```
import React from 'react';
import {Text, StyleSheet, View, TouchableHighlight} from 'react-native';

const Cita=({item, eliminarPaciente})=>{
    const dialogoEliminar=(id)=>{ //aqui los paréntesis son opcionales
        console.log("Eliminando...");
        eliminarPaciente(id);
    }
```

```
return (
<View style={styles.cita}>
    <View>
        <Text style={styles.label}>Paciente:</Text>
        <Text>{item.paciente}</Text>
    </View>
    <View>
        <Text  style={styles.label}>Propietario:</Text>
        <Text>{item.propietario}</Text>
    </View>
    <View>
        <Text  style={styles.label}>Síntomas:</Text>
        <Text>{item.sintomas}</Text>
    </View>
    <View>
    <TouchableHighlight onPress={()=>dialogoEliminar(item.id)} style={styles.btnEliminar}>
    <Text>Eliminar &times; </Text>
    </TouchableHighlight>
    </View>

</View>
)
```

Formulario.js (avance…)

Los elementos que observas al inicio del return <>

Y al final</>

Nos permiten agregar varios componentes en un solo bloque o fragment

App.js | Formulario.js | Cita.js

componentes > Formulario.js > styles

```
1  import React, {useState} from 'react';
2  import { Text, StyleSheet,View, TextInput } from 'react-native';
3
4  const Formulario = () => {
5  return(
6      <>
7      <View style={styles.formulario}>
8      <View>
9      <Text style={styles.label}>Paciente:</Text>
10     <TextInput style={styles.input}
11     />
12     </View>
13     </View>
14     </>
15 );
16 }
17 const styles = StyleSheet.create({
18     formulario:{
19         backgroundColor:'#FFF',
20         paddingHorizontal:20,
21         paddingVertical:10,
22         marginHorizontal:'2.5%'
23     },
24
25     label:{
26         fontWeight:'bold',
27         fontSize:18,
28         marginTop:20
29     },
30     input:{
31         marginTop:10,
32         height:50,
33         borderColor:'#e1e1e1',
34         borderWidth:1,
35         borderStyle:'solid'
36     }
37 });
38 export default Formulario;
```

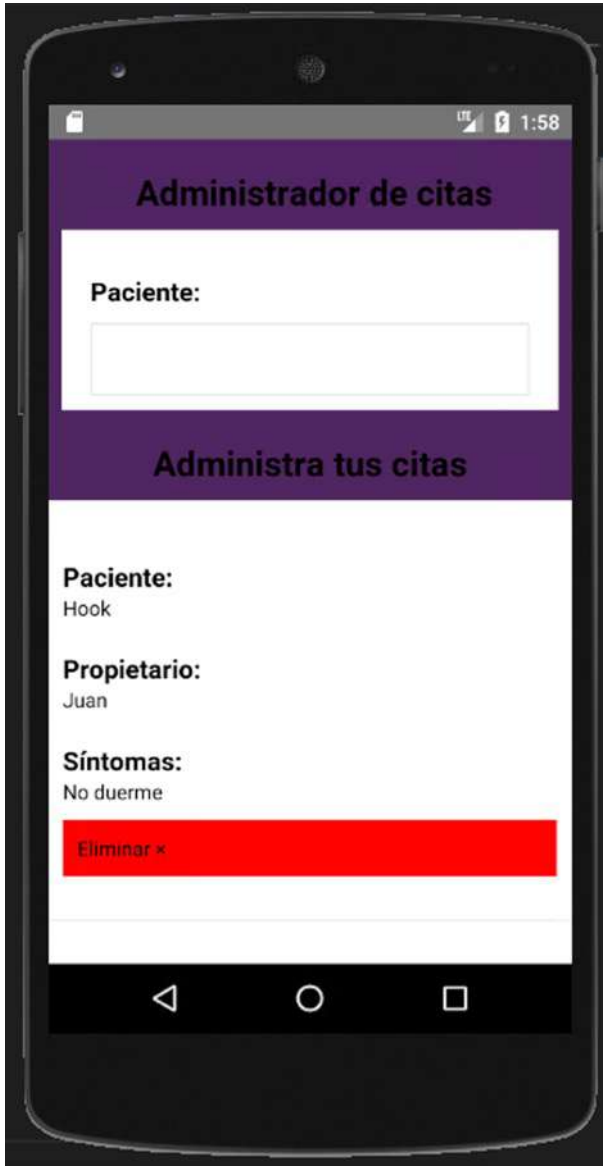
En App.js incluímos el componente Formulario.js

App.js Formulario.js Cita.js

App.js > App > eliminarPaciente

```
1   import React, {useState} from 'react';
2   import { StyleSheet,View,Text,FlatList
3   } from 'react-native';
4
5   import Cita from './componentes/Cita';
6   import Formulario from './componentes/Formulario';
7   const App = () => {
8
9     const[citas, setCitas]=useState([
10      { id:"1", paciente:"Hook", propietario:"Juan", sintomas:"No duerme"},
11      { id:"2", paciente:"Ilax", propietario:"Mario", sintomas:"No come"},
12      { id:"3", paciente:"Yuyis", propietario:"Javier", sintomas:"es muy acelerada"},
13    ]);
14
15  const eliminarPaciente=id=>{
16    setCitas((citasActuales)=>{
17    return citasActuales.filter(cita=>cita.id !== id);
18    })
19  }
20
21    return (
22      <View style={styles.contenedor}>
23      <Text style={styles.titulo}> Administrador de citas</Text>
24
25      <Formulario/>
26
27      <Text style={styles.titulo}>{citas.length > 0 ? 'Administra tus citas':'No hay citas, agrega una'}</Text
28
29      <FlatList
30      data={citas}
31      renderItem={({item})=> <Cita item={item} eliminarPaciente={eliminarPaciente} />}
32      KeyExtractor={citas=>cita.id}
33      />
34
35      </View>
36    );
37  };
```

El resultado de este ejercicio es el siguiente:

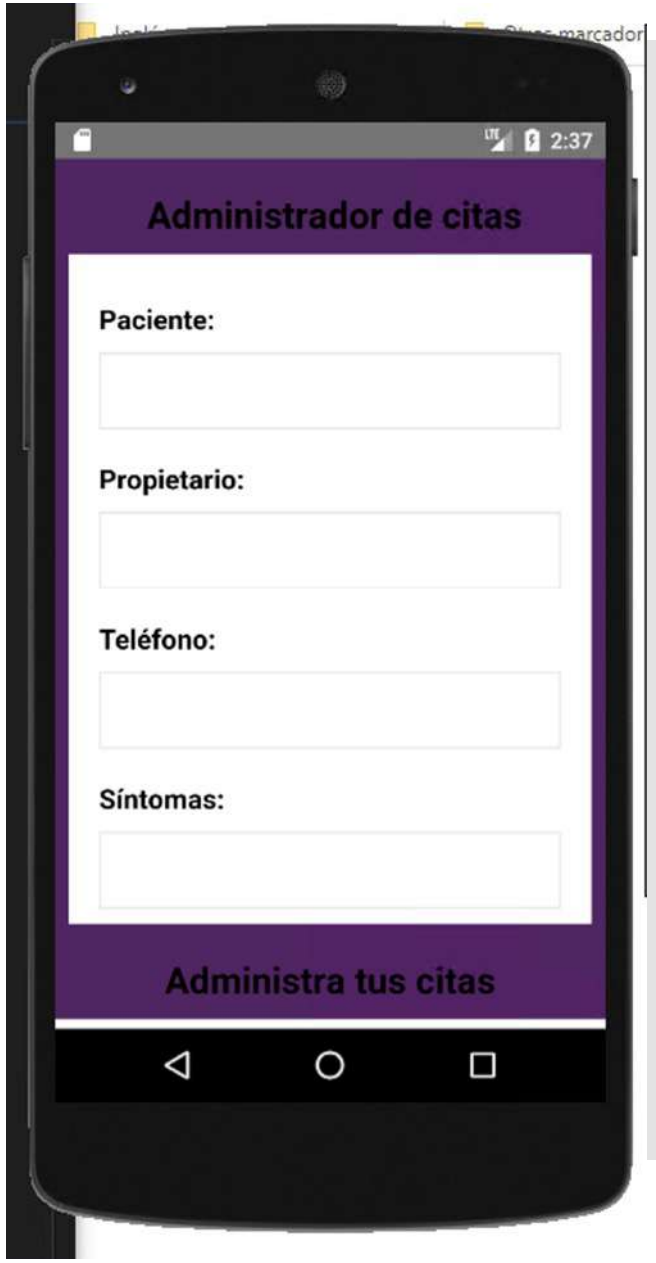


Puedes ejecutar hasta aquí el proyecto
npx react-native run-android

Agregaremos los demás campos al formulario a continuación…

Terminar de realizar formulario y la próxima clase terminamos el ejercicio..

```
6     <>
7       <View style={styles.formulario}>
8         <View>
9           <Text style={styles.label}>Paciente:</Text>
10          <TextInput
11            style={styles.input}
12            onChangeText={(texto) => console.log(texto)}
13          />
14        </View>
15
16        <View>
17          <Text style={styles.label}>Propietario:</Text>
18          <TextInput
19            style={styles.input}
20            onChangeText={(texto) => console.log(texto)}
21          />
22        </View>
23
24        <View>
25          <Text style={styles.label}>Teléfono:</Text>
26          <TextInput
27            style={styles.input}
28            onChangeText={(texto) => console.log(texto)}
29            keyboardType='numeric'
30          />
31        </View>
32
33        <View>
34          <Text style={styles.label}>Síntomas:</Text>
35          <TextInput
36            style={styles.input}
37            onChangeText={(texto) => console.log(texto)}
38            keyboardType='numeric'
39          />
40        </View>
41      </View>
42    </>
```



Segunda parte…

- Terminar formulario con campos de fecha y hora

Ahora agregaremos fecha y hora de cita, pero necesitamos importar librerías

General Usage

```
import DateTimePicker from '@react-native-community/datetimepicker';
```

or

```
const DateTimePicker = require('@react-native-community/datetimepicker');
```

Importar en el archivo: Formulario.js

```
import DateTimePicker from '@react-native-community/datetimepicker';
```

Ahora agregaremos fecha y hora de cita, pero necesitamos importar librerías y este código que nos da la librería de ejemplo

Usage

```
import React, { useState } from "react";
import { Button, View } from "react-native";
import DateTimePickerModal from "react-native-modal-datetime-picker";

const Example = () => {
  const [isDatePickerVisible, setDatePickerVisibility] = useState(false);

  const showDatePicker = () => {
    setDatePickerVisibility(true);
  };

  const hideDatePicker = () => {
    setDatePickerVisibility(false);
  };

  const handleConfirm = (date) => {
    console.warn("A date has been picked: ", date);
    hideDatePicker();
  };

  return (
    <View>
      <Button title="Show Date Picker" onPress={showDatePicker} />
      <DateTimePickerModal
        isVisible={isDatePickerVisible}
        mode="date"
        onConfirm={handleConfirm}
        onCancel={hideDatePicker}
      />
    </View>
  );
};

export default Example;
```

Estos son algunos props que más adelante nos pueden servir

Available props

Name	Type	Default	Description
cancelTextIOS	string	'Cancel'	The label of the cancel button (iOS)
confirmTextIOS	string	'Confirm'	The label of the confirm button (iOS)
customCancelButtonIOS	component		Overrides the default cancel button component (iOS)
customConfirmButtonIOS	component		Overrides the default confirm button component (iOS)
customHeaderIOS	component		Overrides the default header component (iOS)
customPickerIOS	component		Overrides the default native picker component (iOS)
date	obj	new Date()	Initial selected date/time
headerTextIOS	string	"Pick a date"	The title text of header (iOS)
isVisible	bool	false	Show the datetime picker?

Nosotros en formulario.js generaremos las siguientes funciones y código:

Como puedes ver se crearon 6 states para cada uno de los campos del formulario.

Los state: isDateTimePicker sirven para tener valores que nos sirven como intermediarios. Esto lo vermos más adelante.

componentes > Formulario.js > Formulario > mostrarAlerta

```
1   import React, {useState} from 'react';
2   import {
3     Text,
4     StyleSheet,
5     View,
6     TextInput,
7     Button,
8     TouchableHighlight,
9     Alert,
10    ScrollView,
11  } from 'react-native';
12  import DateTimePickerModal from 'react-native-modal-datetime-picker';
13  import shortid from 'shortid';
14
15  const Formulario = ({citas, setCitas, guardarMostrarForm}) => {
16    const [paciente, guardarPaciente] = useState('');
17    const [propietario, guardarPropietario] = useState('');
18    const [telefono, guardarTelefono] = useState('');
19    const [fecha, guardarFecha] = useState('');
20    const [hora, guardarHora] = useState('');
21    const [sintomas, guardarSintomas] = useState('');
22
23    const [isDatePickerVisible, setDatePickerVisibility] = useState(false);
24    const [isTimePickerVisible, setTimePickerVisibility] = useState(false);
25
26    const showDatePicker = () => {
27      setDatePickerVisibility(true);
28    };
29
30    const hideDatePicker = () => {
31      setDatePickerVisibility(false);
32    };
```

Formulario.js

```
34  const confirmarFecha = (date) => {
35    const opciones = {year: 'numeric', month: 'long', day: '2-digit'};
36    guardarFecha(date.toLocaleDateString('es-ES', opciones));
37    hideDatePicker();
38  };
39
40  //Muestra u oculta tl Timepicker
41
42  const showTimePicker = () => {
43    setTimePickerVisibility(true);
44  };
45
46  const hideTimePicker = () => {
47    setTimePickerVisibility(false);
48  };
49
50  const confirmarHora = (hora) => {
51    const opciones = {hour: 'numeric', minute: '2-digit', hour12: false};
52    guardarHora(hora.toLocaleString('es-ES', opciones));
53    hideTimePicker();
54  };
55
56  const crearNuevaCita = () => {
57    if (
58      paciente.trim() === '' ||
59      propietario.trim() === '' ||
60      telefono.trim() === '' ||
61      fecha.trim() === '' ||
62      hora.trim() === '' ||
63      sintomas.trim() === '')
64      {
65      mostrarAlerta();
66      return;
67    }
```

Esta es la función que permite crear una nueva cita. Pero antes tenemos que validar que todos los campos esten insertados

package.json | App.js | Formulario.js | Cita.js

componentes > Formulario.js > Formulario > mostrarAlerta

```
68
69    const cita = {paciente, propietario, telefono, fecha, hora, sintomas};
70
71    cita.id=shortid.generate();
72    const citasNuevo=[...citas,cita];
73    setCitas(citasNuevo);
74    guardarMostrarForm(false);
75  };
76
77
78
79  const mostrarAlerta = () => {
80    Alert.alert(
81      'Error', //Titulo
82      'Todos los campos son obligatorios', //mensaje
83      [
84        {
85          text: 'OK' //arreglo de bontones
86        }
87      ]
88    )
89  }
90
91  return (
92    <>
93      <ScrollView style={styles.formulario}>
94        <View>
95          <Text style={styles.label}>Paciente:</Text>
96          <TextInput
97            style={styles.input}
98            onChangeText={(texto) => guardarPaciente(texto)}
99          />
100       </View>
```

Aquí continúa el registro de cita.
Nota: shortid, nos sirvió
para poder generar Id aleatorios
Ver shortid.generate

Esta función solo se manda llamar (en la función CrearNuevaCita()) cuando los campos no estan completos

Formulario.js

En cada onChangeText, usaremos las funciones de cada state
para guardar los valores registrados
por el usuario

```
101
102         <View>
103           <Text style={styles.label}>Propietario:</Text>
104           <TextInput
105             style={styles.input}
106             onChangeText={(texto) => guardarPropietario(texto)}
107           />
108         </View>
109
110         <View>
111           <Text style={styles.label}>Teléfono:</Text>
112           <TextInput
113             style={styles.input}
114             onChangeText={(texto) => guardarTelefono(texto)}
115             keyboardType="numeric"
116           />
117         </View>
118         <Text style={styles.label}>Fecha:</Text>
119         <Button title="Muestra Fecha" onPress={showDatePicker} />
120         <DateTimePickerModal
121           isVisible={isDatePickerVisible}
122           mode="date"
123           onConfirm={confirmarFecha}
124           onCancel={hideDatePicker}
125           local="es_ES"
126           headerTextIOS="Elige una Fecha"
127           cancelTextIOS="Cancelar"
128           cancelTextIOS="Confirmar"
129           // is24Hour=""
130         />
131         <Text>{fecha}</Text>
132         <Text style={styles.label}>Hora:</Text>
133         <Button title="Muestra hora" onPress={showTimePicker} />
```

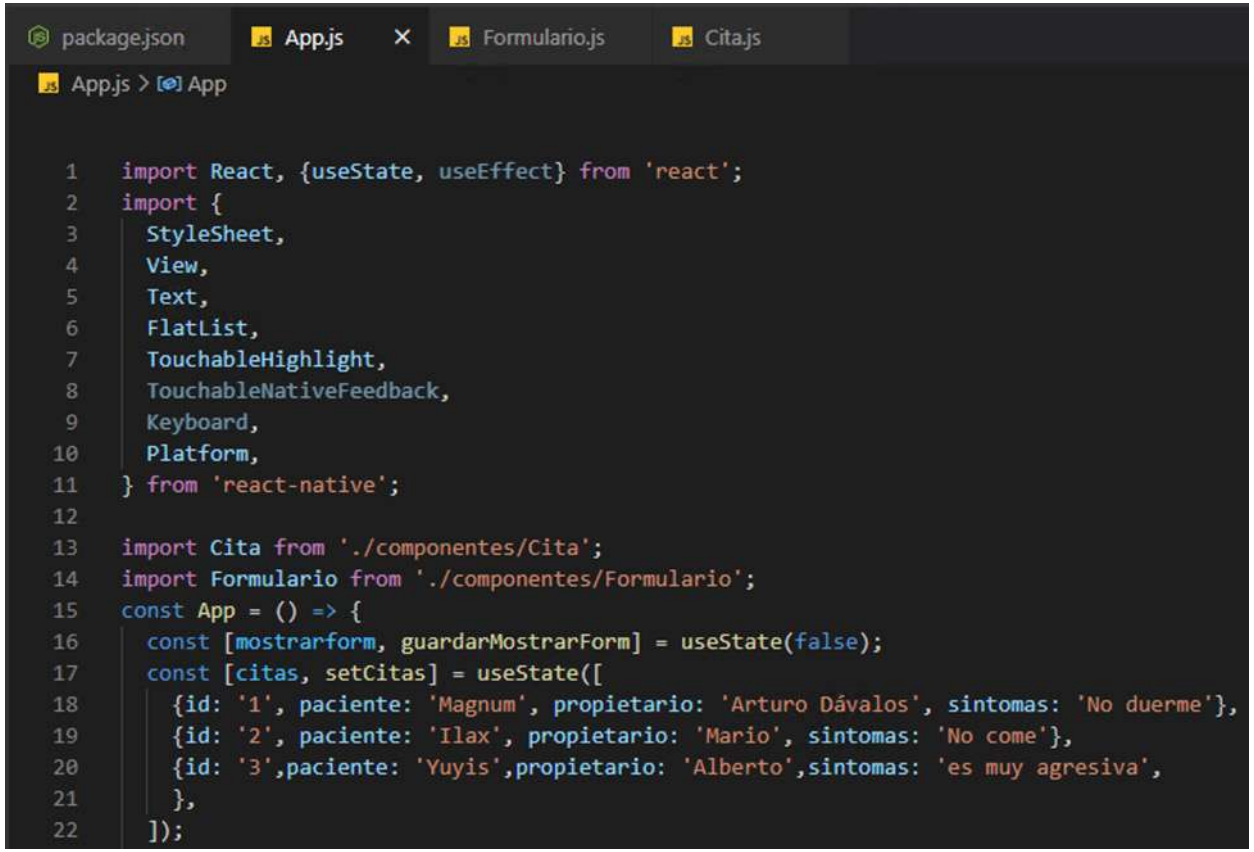
Formulario.js

```
134         <DateTimePickerModal
135           isVisible={isTimePickerVisible}
136           mode="time"
137           onConfirm={confirmarHora}
138           onCancel={hideTimePicker}
139           locale="es_ES"
140           headerTextIOS="Elige una Hora" // solo aplica para IOS, porque android no tiene este apartado.
141           cancelTextIOS="Cancelar"
142           confirmTextIOS="Confirmar"
143         />
144         <Text>{hora}</Text>
145
146         <View>
147           <Text style={styles.label}>Síntomas:</Text>
148           <TextInput
149            multiline
150             style={styles.input}
151             onChangeText={(texto) => guardarSintomas(texto)}
152
153           />
154         </View>
155
156         <View>
157           <TouchableHighlight
158             onPress={() => crearNuevaCita()}
159             style={styles.btnSubmit}>
160             <Text style={styles.textoSubmit}>Crear Nueva Cita </Text>
161           </TouchableHighlight>
162         </View>
163       </ScrollView>
164     </>
165   );
166 }
```

Formulario.js

```
const styles = StyleSheet.create({
  formulario: {
    backgroundColor: '#FFF',
    paddingHorizontal: 20,
    paddingVertical: 10,
  },

  label: {
    fontWeight: 'bold',
    fontSize: 18,
    marginTop: 20,
  },
  input: {
    marginTop: 10,
    height: 50,
    borderColor: '#e1e1e1',
    borderWidth: 1,
    borderStyle: 'solid',
  },
  btnSubmit: {
    backgroundColor: '#7D24BB',
    padding: 10,
    backgroundColor: 'red',
    marginVertical: 10,
  },
  textoSubmit: {
    color: '#FFF',
    fontWeight: 'bold',
    textAlign: 'center',
  }
})
export default Formulario;

```

Vamos a terminar de registrar app.js



App.js

```
23
24  const eliminarPaciente = (id) => {
25    setCitas((citasActuales) => {
26      return citasActuales.filter((cita) => cita.id !== id);
27    });
28  }
29
30  const mostrarFormulario = () => {
31    guardarMostrarForm(!mostrarform);
32  }
33
34  return (
35    <View style={styles.contenedor}>
36      <Text style={styles.titulo}> Administrador de citas</Text>
37      <View>
38        <TouchableHighlight
39          onPress={() => mostrarFormulario()}
40          style={styles.btnMostrarForm}>
41          <Text style={StyleSheet.textoMostrarForm}>
42            Crear Nueva Cita &times;{' '}
43          </Text>
44        </TouchableHighlight>
45      </View>
46
47      <View style={styles.contenido}>
48        {
49        mostrarform ? (
50          <>
51            <Text>Crear nueva cita</Text>
52            <Formulario
53              citas={citas}
54              setCitas={setCitas}
55              guardarMostrarForm={guardarMostrarForm}
56            />
57          </>
58        ) : (
59          <>
```

App.js

```
60              <Text style={styles.titulo}>
61                {citas.length > 0
62                  ? 'Administra tus citas'
63                  : 'No hay citas, agrega una'}
64              </Text>
65              <FlatList
66                style={styles.listado}
67                data={citas}
68                renderItem={({item}) => (
69                  <Cita
70                    item={item} eliminarPaciente={() => eliminarPaciente(item.id)}
71                     keyExtractor={ cita => cita.id}
72                  />
73                )}
74                KeyExtractor={(citas) => cita.id}
75              />
76            </>
77          )}
78        </View>
79      </View>
80    );
81  };
```

App.js

```
//Para los estilos crearemos una variable "styles"
const styles = StyleSheet.create({
  contenedor: {
    backgroundColor: '#522364',
    flex: 1, //Esto permite crecer el espacio del contenedor en un 100%
  },

  titulo: {
    color: '#FFF',
    marginTop: Platform.OS === 'ios' ? 40 : 20,
    marginBottom: 10,
    fontSize: 24,
    fontWeight: 'bold',
    textAlign: 'center',
  },
  contenido: {
    flex: 1,
    marginHorizontal: '2.5%',
  },
  listado: {
    flex: 1,
  },
  btnMostrarForm: {
    backgroundColor: '#AABBCC',
  },
  textoMostrarForm: {
    color: '#fff',
    fontWeight: 'bold',
    textAlign: 'center',
  },
});

export default App;

```

Cita.js

package.json | App.js | Formulario.js | Cita.js

componentes > Cita.js > Cita

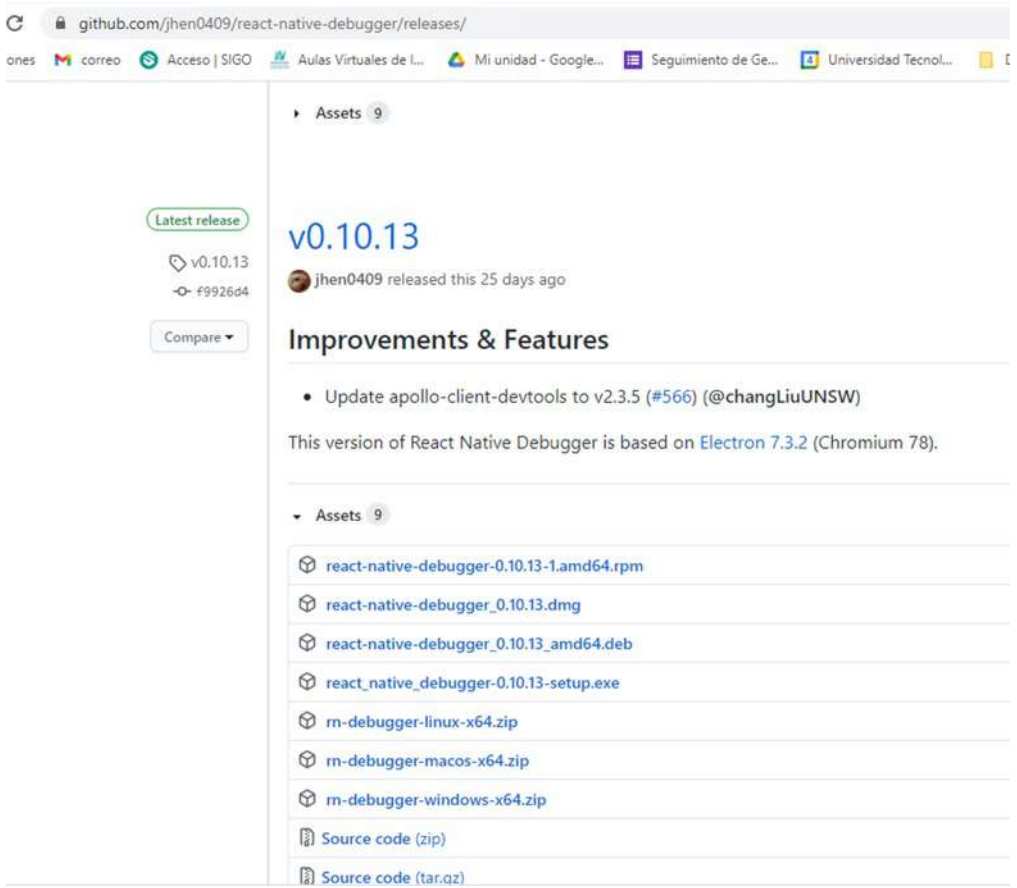
```
1   import React from 'react';
2   import {Text, StyleSheet, View, TouchableHighlight} from 'react-native';
3
4   const Cita = ({item, eliminarPaciente}) => {
5
6     return (
7       <View style={styles.cita}>
8         <View>
9           <Text style={styles.label}>Paciente:</Text>
10          <Text>{item.paciente}</Text>
11        </View>
12        <View>
13          <Text style={styles.label}>Propietario:</Text>
14          <Text>{item.propietario}</Text>
15        </View>
16        <View>
17          <Text style={styles.label}>Síntomas:</Text>
18          <Text>{item.sintomas}</Text>
19        </View>
20        <View>
21          <TouchableHighlight
22            onPress={() => eliminarPaciente(item.id)}
23            style={styles.btnEliminar}>
24            <Text>Eliminar &times; </Text>
25          </TouchableHighlight>
26        </View>
27      </View>
28    )
29  }
30
```

Cita.js

```
31  const styles = StyleSheet.create({
32    cita: {
33      backgroundColor: '#FFF',
34      borderBottomColor: '#e1e1e1',
35      borderStyle: 'solid',
36      borderBottomWidth: 1,
37      paddingVertical: 20,
38      paddingHorizontal: 10,
39    },
40    label: {
41      fontWeight: 'bold',
42      fontSize: 18,
43      marginTop: 20,
44    },
45    texto: {
46      fontSize: 14,
47    },
48    btnEliminar: {
49      padding: 10,
50      backgroundColor: 'red',
51      marginVertical: 10,
52    },
53  });
54
55  export default Cita;
56
```

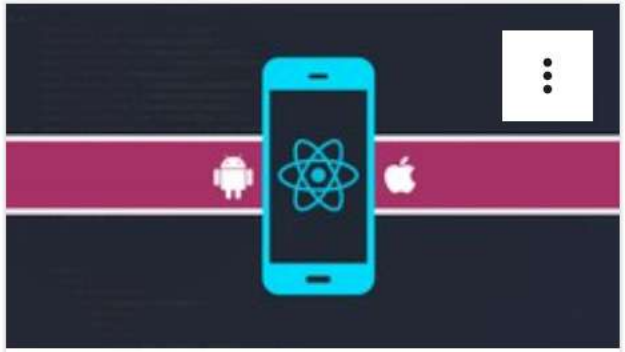
Notas:

Si deseas puedes agregar otro debugger en tu computadora como:



Referencias:

- Curso udemy recomendado de
- Juan Pablo de la Torre Valdéz



React Native - Crea aplicaciones para Android y...

Juan Pablo De la torre Valdez, Creador de Código Con Juan -...