	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERÍA** **ESCUELA DE COMPUTACIÓN**	**CICLO 02-2021**
	DISEÑO Y PROGRAMACIÓN DE SOFTWARE MULTIPLATAFORMA **CREACIÓN DE COMPONENTES PARTE II**	**GUIA DE LABORATORIO N° 5**

I. Resultados de aprendizaje

Que el estudiante:

- Cree proyectos de React Native
- Haga diseño de interfaces de usuario y manipule controles.

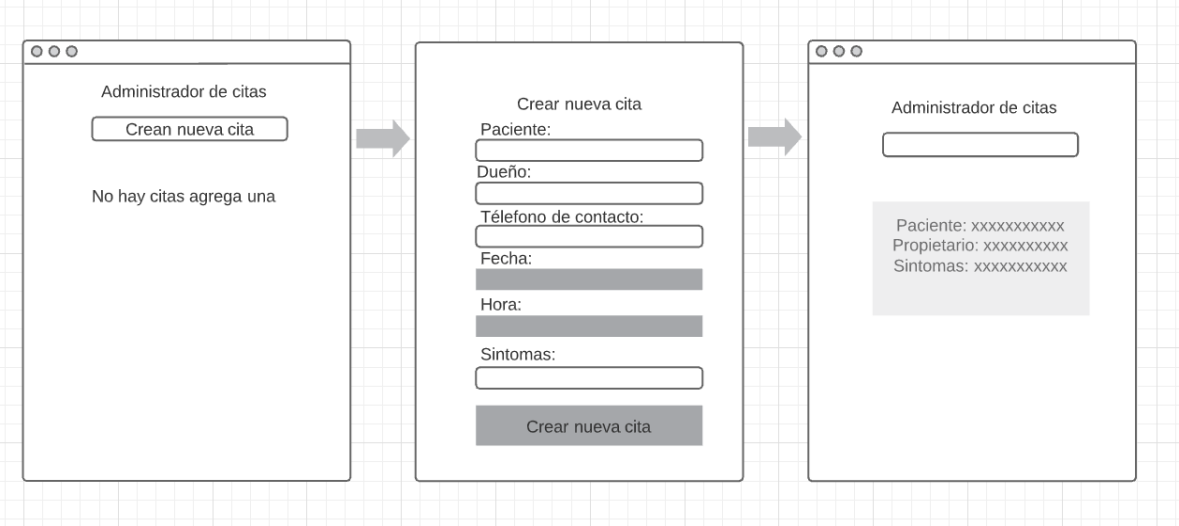
II. Introducción

React native provee un gran número de componentes genéricos, entre los que se encuentran Componentes básicos (View, Text, Image, Text Input), elementos de interfaz de usuario (Button, Picker, Slider, Switch), ListViews y componentes exclusivos del sistema (Android o iOS).

Sin embargo, existen muchos otros componentes creados por la comunidad que pueden incluirse por medio de las dependencias dentro de sus proyectos.

III. DESARROLLO

1. En Visual Studio Code inicie una terminal
2. Sitúese en la carpeta donde desea guardar la aplicación.
3. Procedemos a crear nuestro proyecto “citas”
   npm install --global expo-cli
   expo init citas
4. Procederemos a crear la estructura de nuestro proyecto

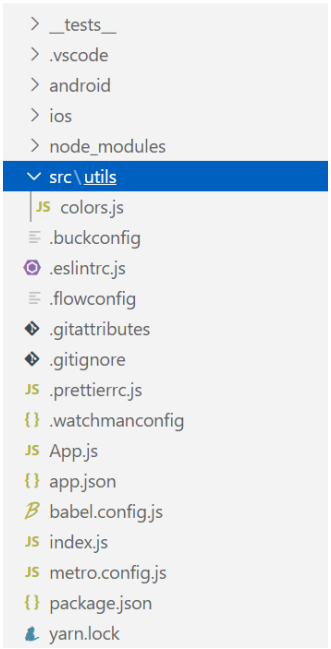


1.Una página principal que nos muestre las lista de citas generadas y nos permita crear una nueva cita en una veterinaria.

2. Un formulario para crear citas para nuestra mascota, donde podemos seleccionar día y hora de la cita.

3.La vista de la pagina principal donde nos muestra todos las citas generadas.

5. para trabajar con los mismos colores en toda la aplicación podemos crear un archivo para el manejo de los nombres de los colores:
    a. Primero crearemos una carpeta llamada src, dentro de esta ubicamos una carpeta de nombre utils y dentro de esta crearemos un archivo de nombre colors.js.



b. Nuestro archivo colors.js tendremos el siguiente código:

```
export default{
    PRIMARY_COLOR: '#1181BF',
    BUTTON_COLOR: '#FFEF36',
}
```

6. Ahora procedemos a instalar las siguientes dependencias para utilizar el datetime-picker:
   `npm i react-native-modal-datetime-picker @react-native-community/datetimepicker`

7. Ahora procedemos a instalar la dependencia para utilizar archivos asíncronos
   `npm install @react-native-async-storage/async-storage`
   `expo install @react-native-async-storage/async-storage`

8. Ahora procedemos a instalar la dependencia para la generación de id automáticos dentro de nuestra aplicación, debido a que generamos una json que se almacenará en nuestro localstorage.
   npm i shortid
9. Procedemos a crear nuestros componente Cita.js:

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

const Cita = ({item, eliminarPaciente}) => {

    const dialogoEliminar = id => {
        console.log('eliminando....', id);
        eliminarPaciente(id);
    }


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

            <View>
                <TouchableHighlight onPress={ () => dialogoEliminar(item.id) }
style={styles.btnEliminar}>
                    <Text style={styles.textoEliminar}> Eliminar &times; </Text>
                </TouchableHighlight>
            </View>

        </View>
```

```
        )
    }

    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: {
            fontSize: 18,
        },
        btnEliminar: {
            padding: 10,
            backgroundColor: 'red',
            marginVertical: 10
        },
        textoEliminar: {
            color: '#FFF',
            fontWeight: 'bold',
            textAlign: 'center'
        }
    })

    export default Cita;
```

10. Procedemos a crear nuestro componente Formulario.js:

```
import React, { useState } from 'react';
import { Text, StyleSheet, View, TextInput, Button, TouchableHighlight, Alert, ScrollView
} from 'react-native';
import DateTimePickerModal from "react-native-modal-datetime-picker";
import shortid from 'shortid';
import colors from '../src/utils/colors';

const Formulario = ({citas, setCitas,  guardarMostrarForm, guardarCitasStorage}) => {
//variables para el formulario
    const [paciente, guardarPaciente] = useState('');
    const [propietario, guardarPropietario] = useState('');
    const [telefono, guardarTelefono] = useState('');
    const [fecha, guardarFecha] = useState('');
    const [hora, guardarHora] = useState('');
    const [sintomas, guardarSintomas] = useState('');


    const [isDatePickerVisible, setDatePickerVisibility] = useState(false);
    const [isTimePickerVisible, setTimePickerVisibility] = useState(false);

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

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

const confirmarFecha = date => {
    const opciones = { year: 'numeric', month: 'long', day: "2-digit" };
    guardarFecha(date.toLocaleDateString('es-ES', opciones));
    hideDatePicker();
};


// Muestra u oculta el Time Picker
const showTimePicker = () => {
    setTimePickerVisibility(true);
};

const hideTimePicker = () => {
    setTimePickerVisibility(false);
};

const confirmarHora = hora => {
    const opciones = { hour: 'numeric', minute: '2-digit', hour12: false};
    guardarHora(hora.toLocaleString('es-ES', opciones));
    hideTimePicker();
};

// Crear nueva cita
const crearNuevaCita = () => {
   // Validar
   if(paciente.trim() === '' ||
        propietario.trim() === '' ||
        telefono.trim() === '' ||
        fecha.trim() === '' ||
        hora.trim() === '' ||
        sintomas.trim()  === '')
        {
            // Falla la validación
            mostrarAlerta();

            return;
        }

    // Crear una nueva cita
    const cita = { paciente, propietario, telefono, fecha, hora, sintomas  };

    cita.id = shortid.generate();

    // console.log(cita);

    // Agregar al state
    const citasNuevo = [...citas, cita];
    setCitas(citasNuevo);

    // Pasar las nuevas citas a storage
    guardarCitasStorage(JSON.stringify(citasNuevo));

    // Ocultar el formulario
    guardarMostrarForm(false);

    // Resetear el formulario
    guardarSintomas('');
```

```
    guardarPropietario('');
    guardarPaciente('');
    guardarHora('');
    guardarFecha('');
    guardarTelefono('');
}

// Muestra la alerta si falla la validación
const mostrarAlerta = () => {
    Alert.alert(
        'Error', // Titulo
        'Todos los campos son obligatorios', // mensaje
        [{
            text: 'OK' // Arreglo de botones
        }]
    )
}


return (
    <>
    <ScrollView style={styles.formulario}>
        <View>
            <Text style={styles.label}>Paciente:</Text>
            <TextInput
                style={styles.input}
                onChangeText={ texto => guardarPaciente(texto) }
            />
        </View>

        <View>
            <Text style={styles.label}>Dueño:</Text>
            <TextInput
                style={styles.input}
                onChangeText={ texto => guardarPropietario(texto) }
            />
        </View>

        <View>
            <Text style={styles.label}>Teléfono Contacto:</Text>
            <TextInput
                style={styles.input}
                onChangeText={ texto => guardarTelefono(texto) }
                keyboardType='numeric'
            />
        </View>

        <View>
            <Text style={styles.label}>Fecha:</Text>
            <Button title="Seleccionar Fecha" onPress={showDatePicker} />
            <DateTimePickerModal
                isVisible={isDatePickerVisible}
                mode="date"
                onConfirm={confirmarFecha}
                onCancel={hideDatePicker}
                locale='es_ES'
                headerTextIOS="Elige la fecha"
                cancelTextIOS="Cancelar"
                confirmTextIOS="Confirmar"
            />
            <Text>{fecha}</Text>
```

```
            </View>

            <View>
                <Text style={styles.label}>Hora:</Text>
                <Button title="Seleccionar Hora" onPress={showTimePicker} />
                <DateTimePickerModal
                    isVisible={isTimePickerVisible}
                    mode="time"
                    onConfirm={confirmarHora}
                    onCancel={hideTimePicker}
                    locale='es_ES'
                    headerTextIOS="Elige una Hora"
                    cancelTextIOS="Cancelar"
                    confirmTextIOS="Confirmar"
                />
               <Text>{hora}</Text>
            </View>

            <View>
                <Text style={styles.label}>Síntomas:</Text>
                <TextInput
                    multiline
                    style={styles.input}
                    onChangeText={ texto => guardarSintomas(texto) }
                />
            </View>

            <View>
                <TouchableHighlight onPress={ () => crearNuevaCita() }
style={styles.btnSubmit}>
                    <Text style={styles.textoSubmit}>Crear Nueva Cita</Text>
                </TouchableHighlight>
            </View>
        </ScrollView>
        </>
      );
}

const styles = StyleSheet.create({
    formulario: {
        backgroundColor: '#FFF',
        paddingHorizontal: 20,
        paddingVertical: 10,
        flex: 1
    },
    label: {
        fontWeight: 'bold',
        fontSize: 18,
        marginTop: 20
    },
    input: {
        marginTop: 10,
        height: 50,
        borderColor: '#e1e1e1',
        borderWidth: 1,
        borderStyle: 'solid'
    },
    btnSubmit: {
        padding: 10,
        backgroundColor:colors.BUTTON_COLOR,
        marginVertical: 10
```

```
    },
    textoSubmit: {
        color: '#FFF',
        fontWeight: 'bold',
        textAlign: 'center'
    }
})

export default Formulario;
```

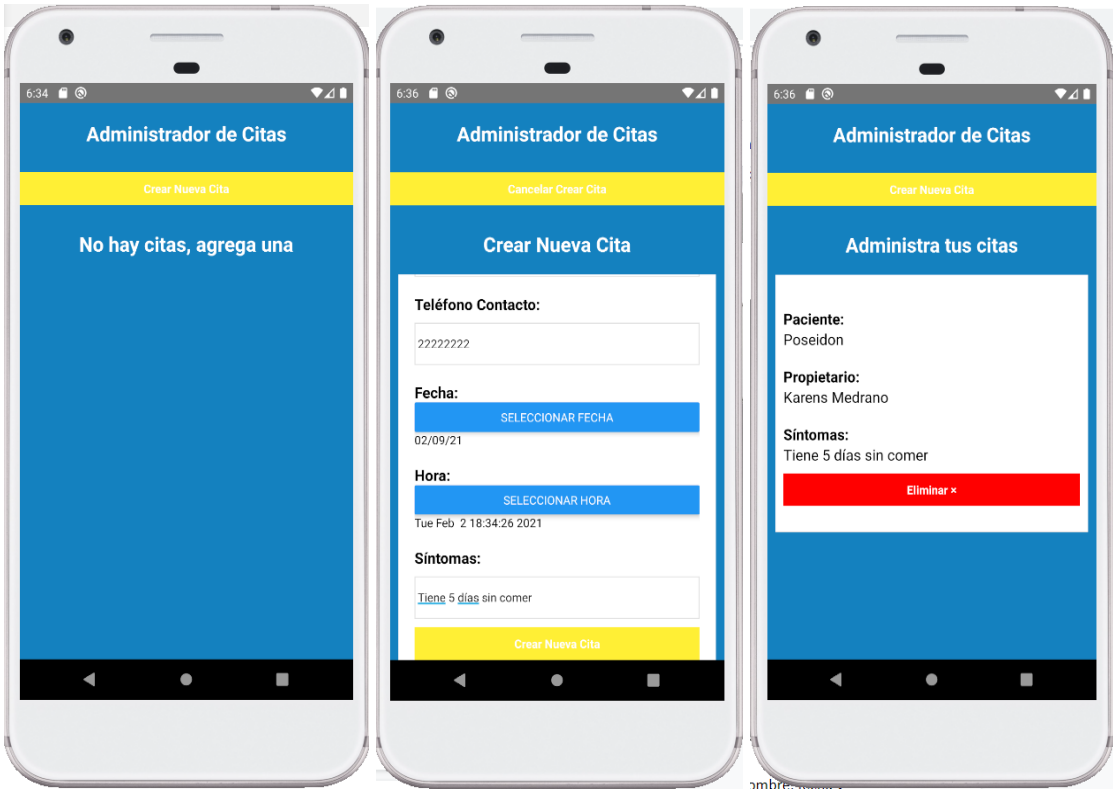
11. Ya creados nuestros componentes procedemos a modificar App.js

```
import React, { useState, useEffect } from 'react';
import { Text, StyleSheet, View, FlatList, TouchableHighlight, TouchableWithoutFeedback,
Keyboard, Platform  } from 'react-native';
import Cita from './componentes/Cita';
import Formulario from './componentes/Formulario';
import AsyncStorage from '@react-native-async-storage/async-storage';
import Colors from './src/utils/colors';

const App = () => {
  // definir el state de citas
  const [citas, setCitas] = useState([]);
  const [mostrarform, guardarMostrarForm] = useState(false);

  useEffect(() => {
    const obtenerCitasStorage = async () => {
        try {
            const citasStorage = await AsyncStorage.getItem('citas');
            if(citasStorage) {
                setCitas(JSON.parse(citasStorage))
            }
        } catch (error) {
            console.log(error);
        }
    }
    obtenerCitasStorage();
  }, []);



  // Elimina los pacientes del state
  const eliminarPaciente = id => {

    const citasFiltradas = citas.filter( cita => cita.id !== id );
    setCitas( citasFiltradas );
    guardarCitasStorage(JSON.stringify(citasFiltradas));
  }

  // Muestra u oculta el Formulario
  const mostrarFormulario = () => {
    guardarMostrarForm(!mostrarform);
  }

  // Ocultar el teclado
  const cerrarTeclado = () => {
    Keyboard.dismiss();
  }

  // Almacenar las citas en storage
```

```
  const guardarCitasStorage = async (citasJSON) => {
    try {
      await AsyncStorage.setItem('citas', citasJSON);
    } catch (error) {
        console.log(error);
    }
  }

  return (
    <TouchableWithoutFeedback onPress={() => cerrarTeclado() }>
      <View style={styles.contenedor}>
          <Text style={styles.titulo}>Administrador de Citas</Text>

          <View>
              <TouchableHighlight onPress={ () => mostrarFormulario() }
style={styles.btnMostrarForm}>
                  <Text style={styles.textoMostrarForm}> {mostrarform ? 'Cancelar Crear
Cita' : 'Crear Nueva Cita'} </Text>
              </TouchableHighlight>
          </View>

          <View style={styles.contenido}>
            { mostrarform ? (
              <>
                <Text style={styles.titulo}>Crear Nueva Cita</Text>
                <Formulario
                  citas={citas}
                  setCitas={setCitas}
                  guardarMostrarForm={guardarMostrarForm}
                  guardarCitasStorage={guardarCitasStorage}
                />
              </>
            ) : (
              <>
                <Text style={styles.titulo}> {citas.length > 0 ? 'Administra tus citas' :
'No hay citas, agrega una'} </Text>
                <FlatList
                    style={styles.listado}
                    data={citas}
                    renderItem={ ({item}) => <Cita item={item}
eliminarPaciente={eliminarPaciente} />  }
                    keyExtractor={ cita => cita.id}
                />
              </>
            ) }


          </View>

      </View>
    </TouchableWithoutFeedback>
  );
};

const styles = StyleSheet.create({
  contenedor: {
    backgroundColor: Colors.PRIMARY_COLOR,
    flex: 1
  },
  titulo: {
    color: '#FFF',
```

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

export default App;
```

12. La aplicación deberá de verse de la siguiente forma:



III. EJERCICIOS COMPLEMENTARIOS

Crear una aplicación para un restaurante, donde los usuarios puedan ingresar su nombre, fecha y hora de la reserva, cantidad de personas y seleccionar el tipo de sección(Fumadores/No fumadores) donde desea comer. Desde la aplicación deberá verse el listado de las reservas hechas en el restaurante con la información brindada por el usuario.