

MANUAL TECNICO



ELIZABETH SUAREZ
BYRON CARRILLO

INDICE

1.Índice	1
2.Introducción	2
3.Requerimientos	3
4.Logotipo	4
5.0bjetivo	4
6.Login	5
7.Inicio	7
8.Modelos	9
9.Servicios	10
10.Sucursales	11
11.Perfil	12
12.ModeloS y ModeloR	13
13.Buys	15
14.DetallesCita	17

INTRODUCCION

El presente manual presenta los requerimientos para el debido funcionamiento y desarrollo de la aplicación Rivian Sivar, brindando información sobre su respectivo condigo, para una mejor comprensión del desarrollo de esta.

REQUERIMIENTOS

Firebase: https://firebase.google.com/products/realtime-database?hl=es-419&gclid=CjwKCAjwlcaRBhBYEiwAK341jcvvqaV1JhS80ApIld0syhURfRBpTkhw4wgU9oAPj8_o2Xc6--wSxRoCrV8QAvD_BwE&gclsrc=aw.ds

Nodejs: <https://nodejs.org/es/>

Visual Estudio Code:

<https://code.visualstudio.com/download>
[Software Developer Kit de Java]
<https://www.oracle.com/java/technologies/javase-downloads.html>

Emulador: (2 sugerencias)

1. Android Studio: <https://developer.android.com/studio>
2. Nox: <https://es.bignox.com/>

Yarn

npm install --global yarn

expo

yarn global add expo-cli

Firebase

npm install firebase

npm add @react-native-firebase/app

Navegacion

npm install @react-navigation/native

npm install @react-navigation/native-stack

Carrusel

npm install react-native-looped-carousel --save

LOGOTIPO



OBJETIVO

Se desarrollo una aplicacion en React Native para brindar infomacion precisa sobre los modelos de autos que posee la empresa Rivian Sivar, el costo de estos y poder ser pre-evaluados para una posible compra

LOGIN



RIVIAN SIVAR

Bienvenido

Correo Electronico

Contraseña

Iniciar Sesión

Registrarme



Google

Se presenta el inicio de sesión conformado por varios <View>, los cuales forman parte de los contenedores de la aplicación.

Los campos que solicitan la información del usuario están compuestos por un <TextInput>.

Los respectivos botones son del tipo <TouchableOpacity>

```
import { useNavigation } from '@react-navigation/core'  
import React, { useEffect, useState } from 'react'  
import { KeyboardAvoidingView, StyleSheet, Text,  
| | TextInput, TouchableOpacity, View, Image } from 'react-native'  
import { auth } from './firebase'
```

LoginScreen.js contiene las siguientes importaciones para su buen funcionamiento, necesarios para su navegación, base de datos y etiquetas

```

const LoginScreen = () => {
  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')

```

Comenzamos creando constantes las cuales van a evaluar si los campos de Correo y Contraseña están vacíos, de ser así lanzar un error.

```

const navigation = useNavigation()

useEffect(() => {
  const homenavigation = auth.onAuthStateChanged(user => {
    if (user) {
      navigation.replace(['Home'])
    }
  })
  return homenavigation
}

```

Creamos la constante para el funcionamiento de la navegación, useEffect lo utilizamos para pasar al componente HOME en caso de poder iniciar sesión o registrarnos,

```

const SignUp = () => {
  auth
    .createUserWithEmailAndPassword(email, password)
    .then(userCredentials => {
      const user = userCredentials.user;
      console.log('Registro completo:', user.email);
    })
    .catch(error => alert(error.message))
}

const Login = () => {
  auth
    .signInWithEmailAndPassword(email, password)
    .then(userCredentials => {
      const user = userCredentials.user;
      console.log('Ingresando:', user.email);
    })
    .catch(error => alert(error.message))
}

```

Evaluando si el correo y contraseña del registro son validos.

Evaluando si el correo y contraseña coinciden con algún registro dentro de la base de datos.

en caso de que algún dato sea invalido mostrar el mensaje de error.

```

<Text style={styles.textBienvenida}>BIENVENIDO</Text>
<View style={styles.View1}>
  <TextInput
    placeholder="Correo Electronico"
    value={email}
    onChangeText={text => setEmail(text)}
    style={styles.input1}
  /></View>
<View style={styles.View2}>
  <TextInput
    placeholder="Contraseña"
    value={password}
    onChangeText={text => setPassword(text)}
    style={styles.input2}
    secureTextEntry
  />
</View>

```

Creando los campos respectivos para el ingreso del usuario

```

<View style={styles.Viewbtn1}>
  <TouchableOpacity
    onPress={Login}
    style={styles.button1}
  >
    <Text style={styles.buttonText1}>Iniciar Sesión</Text>
  </TouchableOpacity>
</View>
<View style={styles.Viewbtn2}>
  ....
  <TouchableOpacity
    onPress={SignUp}
    style={[styles.button2]}>
    <Text style={styles.buttonText2}>Registrarme</Text>
  </TouchableOpacity>
</View>
<View style={styles.Viewbtn3}>
  <TouchableOpacity
    onPress={() => {}}
    style={[styles.button3]}>
    <Image style={styles.img}
      source={require("./assets/search.png")}>
      <Text style={styles.buttonText3}>Google</Text>
    </Image>
  </TouchableOpacity>
</View>
</KeyboardAvoidingView>

```

Dando función a los botones de Registró, Iniciar Sesión y Google.

INICIO



Inicialmente nos encontramos con un carrusel de imágenes

En el código principal veremos las importaciones de l Archivo Home.js las cuales son necesarias para su respectivo funcionamiento

```

import React from 'react';
import {View, StyleSheet, Image, Dimensions, Animated, StatusBar} from 'react-native';
import { useNavigation } from '@react-navigation/core';
import { auth } from './firebase';
import { SafeAreaView } from 'react-native-safe-area-context';
import { LinearGradient } from "expo-linear-gradient";

```

```
const Home = (props) => [
  // arreglo de imágenes
  const imágenes = [
    "https://scontent.fsal3-1.fna.fbcdn.net/v/t39.30808-6/277727331_101439585870733_84885209_",
    "https://scontent.fsal2-1.fna.fbcdn.net/v/t39.30808-6/277761523_101439602537398_40405625_",
    "https://scontent.fsal2-1.fna.fbcdn.net/v/t39.30808-6/277764841_101439565870735_86060146_",
    "https://scontent.fsal2-1.fna.fbcdn.net/v/t39.30808-6/276139425_101439625870729_39699192_"
  ];
  // colocando dimensiones
  const width = Dimensions.get("window").width;
  const height = Dimensions.get("window").height;
```

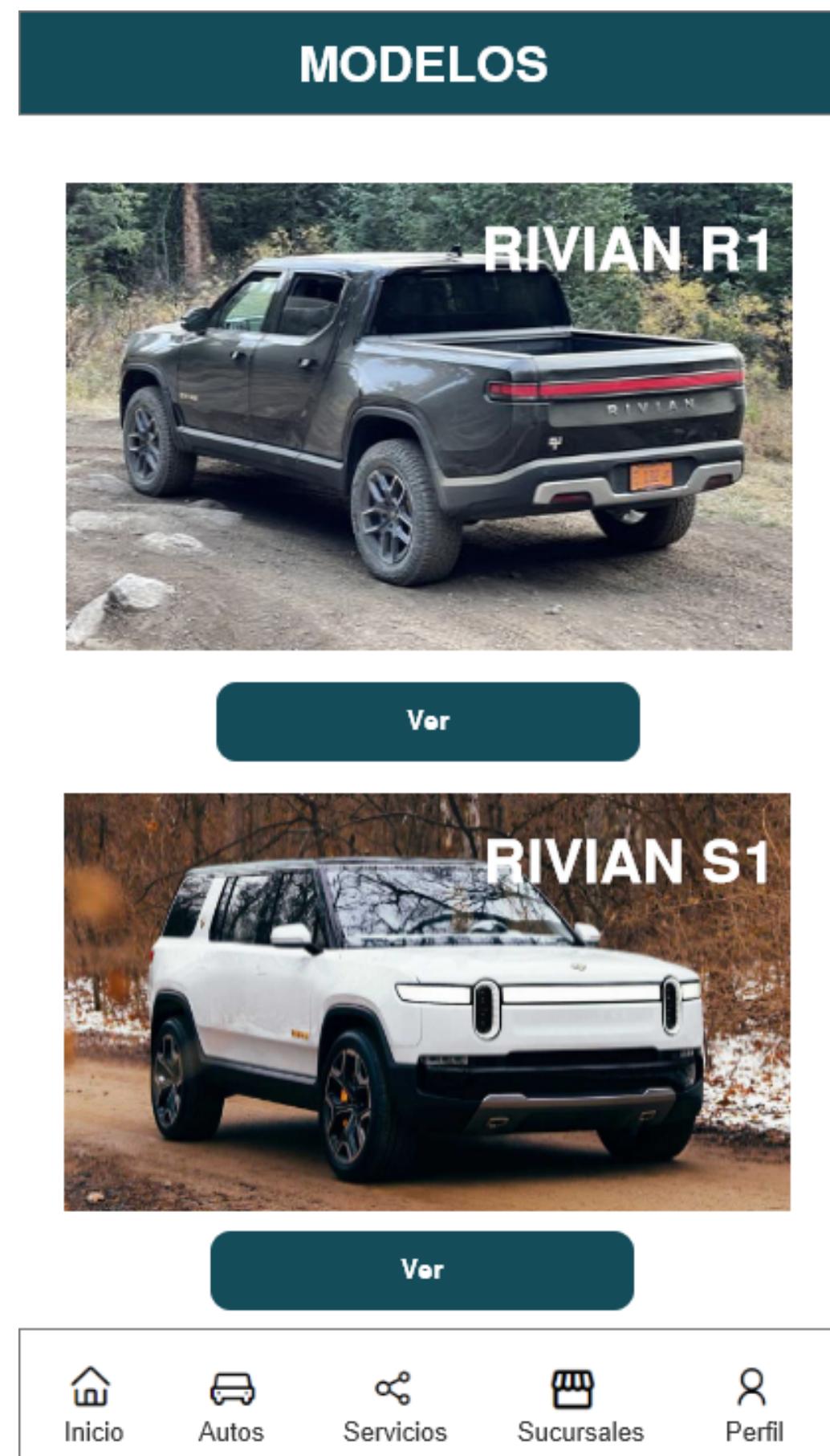
Por medio de un arreglo se estarán cargando las respectivas imágenes del carrusel, directamente de la web, y se procede a adaptar estas al ancho y alto de la pantalla.

```
<View style={{ width: ancho_del_contenedor }}>
  <Animated.View
    style={{
      marginHorizontal: espacio,
      padding: espacio,
      borderRadius: 30,
      backgroundColor: "white",
      alignItems: "center",
      transform: [{ translateY: scrollY }],
    }}
  >
    <Image source={{ uri: item }} style={styles.posterImage} />
  </Animated.View>
</View>
);
```

Luego de brindarle los estilos necesarios, dentro del return colocamos la vista de la animación que hemos declarado

MODELOS

En este archivo se muestran los modelos existentes en la aplicación, se utilizan Imagenes y botones correspondientes para la comprensión del usuario final.



```
import React from 'react';
import { SafeAreaView, Text, View, Image, StyleSheet, TouchableOpacity } from 'react-native';
import { useNavigation } from '@react-navigation/core';
```

Importaciones necesarias para su funcionamiento

```

const Modelos =() => {
  const navigation = useNavigation()

  return (
    <SafeAreaView style={styles.container}>
      <View style={styles.titulo}>
        <Text style={styles.txttitulo}>MODELOS</Text>
      </View>
      | <View><Text style={styles.textStyle1}>RIVIAN R1</Text>
      | <Image style={styles.image1} source={require("./assets/r5.jpg") }/>
      | <TouchableOpacity style={styles.button1} onPress={() =>navigation.navigate('ModelR', {
      |   <Image style={styles.image1} source={require("./assets/r5.jpg") }/>
      |   <Text style={styles.txttitulo}>RIVIAN R1</Text>
      |   <Image style={styles.image1} source={require("./assets/r5.jpg") }/>
      |   <TouchableOpacity style={styles.button1} onPress={() =>navigation.navigate('ModelS', {
      |     <Image style={styles.image1} source={require("./assets/s1.jpg") }/>
      |     <Text style={styles.txttitulo}>RIVIAN S1</Text>
      |     <Image style={styles.image1} source={require("./assets/s1.jpg") }/>
      |     <TouchableOpacity style={styles.button1} onPress={() =>navigation.navigate('ModelS', {
      |       <Image style={styles.image1} source={require("./assets/s1.jpg") }/>
      |       <Text style={styles.txttitulo}>RIVIAN S1</Text>
      |       <Image style={styles.image1} source={require("./assets/s1.jpg") }/>
      |     </TouchableOpacity></View>
      |   </TouchableOpacity></View>
    </SafeAreaView>
  )

```

Dentro del return colocamos las respectivas imágenes por medio de la etiqueta `<Image>` y los botones `<TouchableOpacity>` con su respectivo enlace a distintos archivos que funcionan por medio de Navigation

SERVICIOS

SERVICIOS



En el archivo llamado Servicios.js se mostaran los servicios brindados por la empresa, por medio de imagenes en etiquetas `<Image>` y `ScrollView`



```
import React from 'react';
import { SafeAreaView, Text, ScrollView, View, Image, StyleSheet } from 'react-native';
```

Importaciones necesarias

```
const Servicios = () => {
  return (
    <SafeAreaView style={styles.container}>
      <View style={styles.viewtitle}>
        <Text style={styles.txttitulo}>SERVICIOS</Text>
      </View>
      <ScrollView style={styles.scrollviewstyle}>
        <View style={{ width: '100%', height: '100%' }}>
          <Image style={styles.img}
            source={require("./assets/seguro.png")}/>
          <Image style={styles.img}
            source={require("./assets/taller.png")}/>
          <Image style={styles.img}
            source={require("./assets/carga.png")}/>
          <Image style={styles.img}
            source={require("./assets/herramientas.png")}/>
        </View>
      </ScrollView>
    </SafeAreaView>
  );
}
```

Realizamos la creación de la vista de las respectivas imágenes y por medio de un ScrollView permitidos la visualización de todas las imágenes informativas

SUCURSALES

SUCURSALES



En el presente archivo denominado Sucusales.js brindamos información sobre las sucursales de la empresa



```

import React from 'react';
import {SafeAreaView, Text, View, Image, TouchableOpacity, StyleSheet, Linking}
from 'react-native';

```

Importaciones necesarias para el funcionamiento del archivo

```

const Sucursales = () => {
  return (
    <SafeAreaView style={styles.container}>
      <View style={styles.titulo}>
        <Text style={styles.txttitulo}>SUCURSALES</Text>
      </View>
      <View style={styles.viewlocal1}>
        <TouchableOpacity style={styles.local1}
          onPress={() => { Linking.openURL('https://goo.gl/maps/wGAzjAortU7Z5QG58') }}>
          <Image style={styles.image1}
            | source={require("./assets/garage.png")}>
          <View style={styles.viewimage1}>
            <Text style={styles.txtS1}>San Salvador</Text>
            <Image style={styles.imageempresa1}
              | source={require("./assets/nombreempresa.png")}>
          </View>
        </TouchableOpacity>
      </View>
      <View style={styles.viewlocal2}>
        <TouchableOpacity style={styles.local2}
          onPress={() => { Linking.openURL('https://goo.gl/maps/jCCXKaejPp4hcSkXA') }}>
          <Image style={styles.image2}
            | source={require("./assets/garage.png")}>
          <View style={styles.viewimage2}>
            <Text style={styles.txtS2}>La libertad</Text>
            <Image style={styles.imageempresa2}
              | source={require("./assets/nombreempresa.png")}>
          </View>
        </TouchableOpacity>
      </View>
    </SafeAreaView>
  )
}

```

Dentro del return se muestran los datos de las sucursales por medio de etiquetas como los <View> que conforman los contenedores, <Image>, <TouchableOpacity> los cuales contienen la ruta de destino y sus <Text>

PERFIL

PERFIL



Version 1.0

Cerrar Sesión

En el archivo denominado Perfil.js se muestra el logo de la app y el usuario puede Cerrar la sesión de la APP



```
import React from 'react';
import { SafeAreaView, Text, View, Image, StyleSheet, TouchableOpacity } from 'react-native';
import { useNavigation } from '@react-navigation/core';
import { auth } from './firebase';
```

Importaciones necesarias para el funcionamiento del archivo

```
const AboutRivian = () => {
  const navigation = useNavigation()
  const SignOut = () => {
    auth
      .signOut()
      .then(() => {
        |  navigation.replace("Login")
      })
      .catch(error => alert(error.message))
  }

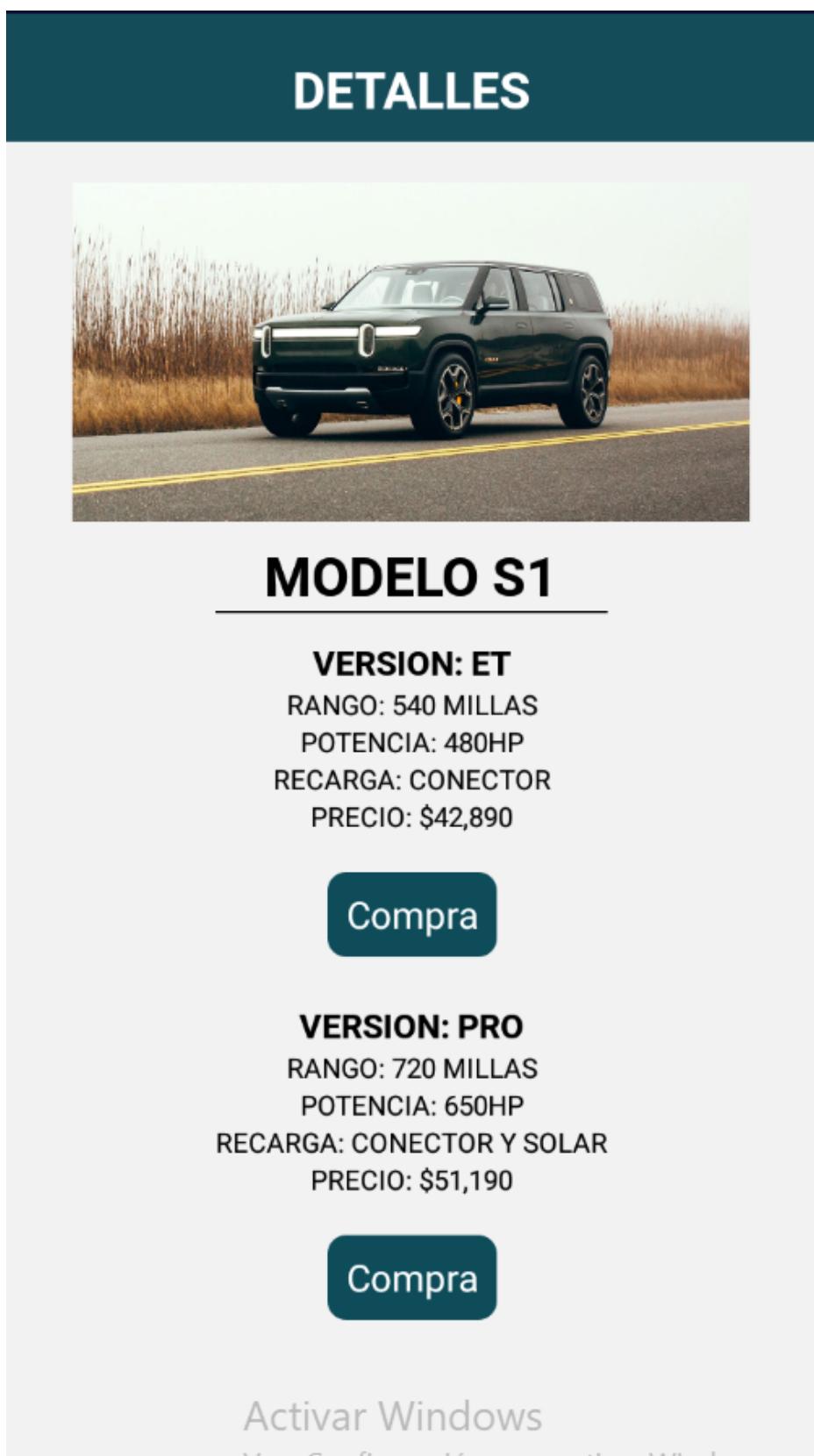
  return (
    <SafeAreaView style={styles.container}>
      <Image style={styles.image1} source={require("./assets/Logo.png")}>
      <Text style={styles.txtversion}>Version 1.0</Text>
      <View style={styles.viewfinal}>
        |  <TouchableOpacity style={styles.button1} onPress={SignOut}><Text style={styles.button1text}>Cerrar Sesión</Text>
        |  </TouchableOpacity>
      </View>
    </SafeAreaView>
  )
}
```

Hacemos uso nuevamente de Navigation para darle el respectivo funcionamiento a nuestro botón de Cerrar Sesión el cual tiene la instrucciones que al ser presionado nos redigira al Login, dentro del return nos encontramos con la etiqueta <Image> que contiene nuestro logo , etiquetas como <Text> para nuestras versiones y <TouchableOpacity> para mostrar nuestro botón y ejecutar la acción

MODELOS Y MODELO R

```
import React, { useState, useEffect } from "react";
import { SafeAreaView, ActivityIndicator, FlatList,
Text, View, Image, StyleSheet, TouchableOpacity } from 'react-native';
import { useNavigation } from '@react-navigation/core';
```

Importaciones de ambas clases



Se muestran las versiones disponibles para los modelos disponibles en la empresa

```
const autoURL = "https://raw.githubusercontent.com/Elizabeth186/RepoJSON/main/s1.json"
const Models = () => {
  const navigation = useNavigation()

  const [isLoading, setLoading] = useState(true);
  const [data, setData] = useState([]);
  const [title, setTitle] = useState([]);
```

Creamos la constante y le damos el valor de la Url de nuestra API a consumir, colocamos las constantes de la navegacion y las que posteriormente traaran el valor de las llamadas a nuestra API para obtener los datos requeridos.

```
useEffect(() => {
  fetch(autoURL)
    .then((response) => response.json())
    .then((json) => {
      setData(json.Versiones);
      setTitle(json.title);

    })
    .catch((error) => alert(error))
    .finally(() => setLoading(false));
}, []);
```

Por medio de UseEffect accedemos a las variables declaradas, obtenemos los datos de la Api y hacemos un catch para verificar errores.

```
async function getMoviesAsync() {
  try {
    let response = await fetch(autoURL);
    let json = await response.json();
    setData(json.Versiones);
    setTitle(json.title);
    setLoading(false);
  } catch (error) {
    alert(error);
  }
}
```

Realizamos el formato json

```
<ActivityIndicator />
) : (
<View>
  <Text style={styles.title}>{title}</Text>
  <FlatList
    data={data}
    keyExtractor={({ id }, index) => id}
    renderItem={({ item }) => (
      <View style={{ padding: 10, margin: 5 }}>
        <Text style={styles.textversion}>VERSION: {item.Version}</Text>
        <Text style={styles.textlist}>RANGO: {item.RANGO}</Text>
        <Text style={styles.textlist}>POTENCIA: {item.POTENCIA}</Text>
        <Text style={styles.textlist}>RECARGA: {item.RECARGA}</Text>
        <Text style={styles.textlist}>PRECIO: {item.PRECIO}</Text>
        <View style={{ marginBottom: 15, marginTop: 15 }}>
          <TouchableOpacity style={styles.button1} onPress={() => navigation.navigate('Buy', { item })}>
            <Text style={styles.buttonText}>Comprar</Text>
          </TouchableOpacity>
        </View>
      </View>
    )}
  </FlatList>
</View>
)
```

Retornamos los valores de la API, las cuales se mostraran al usuario

BUYS

```
import React, { Component } from 'react'
import { SafeAreaView, ScrollView, Text, View,
Image, StyleSheet, TouchableOpacity } from 'react-native';
```

Importaciones de la clase



RIVIAN S1

Version ET

PRECIO FINAL: \$ \$ 42,890

CUOTA: \$ 400

PLAZO: 60 MESES

PRIMA: \$ 12,000

[Agendar Cita](#)

Activar Windows

En estas clases mostramos de talles del plan de pago con el que disponemos en nuestros modelos existentes

```
<SafeAreaView style={styles.container}>
  <View><Text style={styles.textStyle1}>RIVIAN R1</Text>
  <Image style={styles.image1} source={require("./assets/r6.jpg")}>
  </View>
  <Text style={styles.textStyle2}>Version ET</Text>
  <View style={styles.View2}>
    <View style={styles.View1}>
      <Text style={styles.textStyle3}>PRECIO FINAL:</Text>
      <Text style={styles.textStyle3}>CUOTA:</Text>
      <Text style={styles.textStyle3}>PLAZO:</Text>
      <Text style={styles.textStyle3}>PRIMA:</Text>
    </View>

    <View style={styles.View11}>
      <Text style={styles.textStyle31}>$ $ 48,190</Text>
      <Text style={styles.textStyle31}>$ 400</Text>
      <Text style={styles.textStyle31}>60 MESES</Text>
      <Text style={styles.textStyle31}>$ 12,000</Text>
    </View>
  </View>
</SafeAreaView>
```

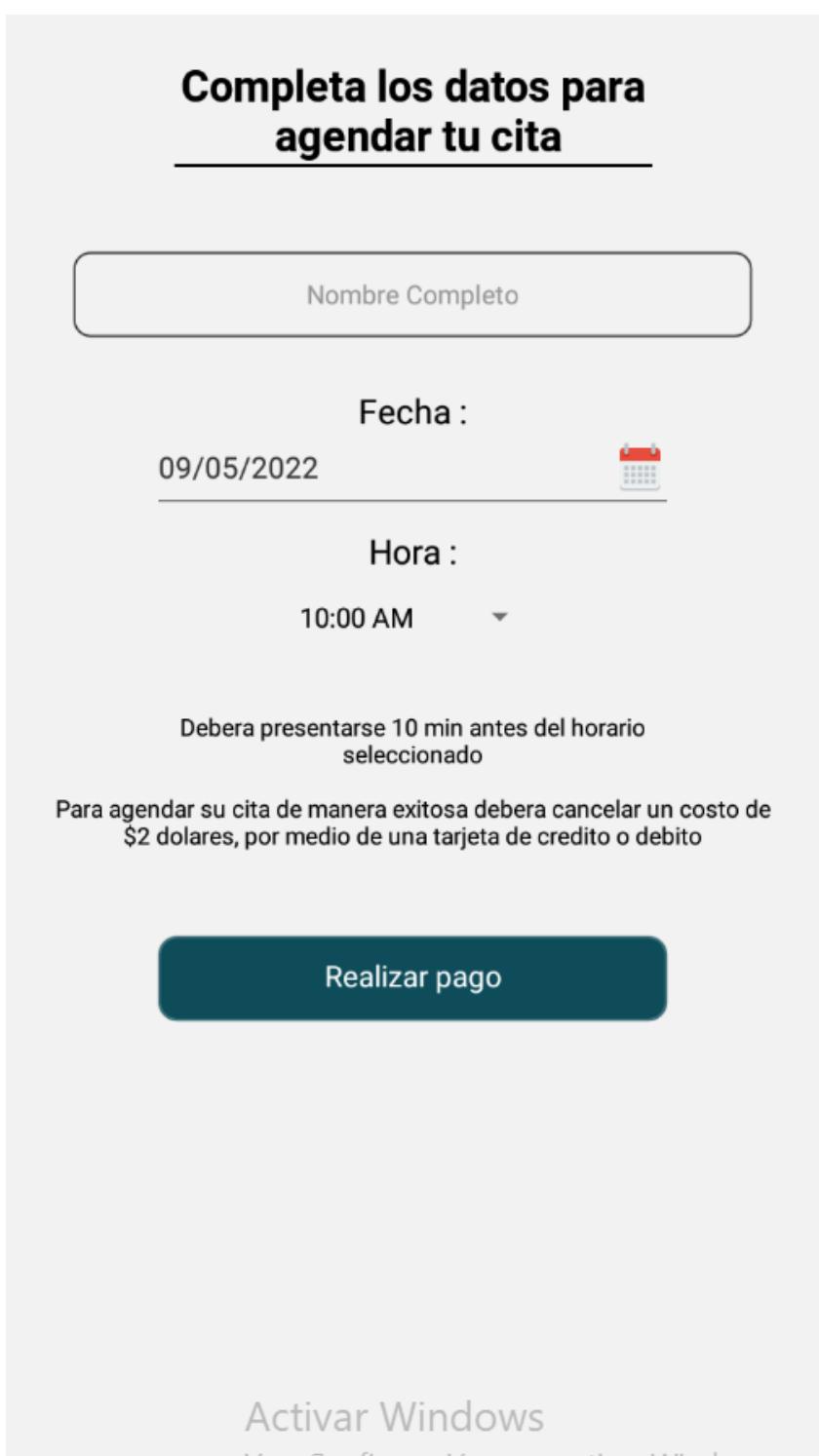
Activar W
Ve a Configur

Mostramos la informacion por medio de 2 <View> los cuales se posicionan en forma de lista a la vista del usuario.

DETALLES CITA

```
import React, { useState, Component } from 'react'
import { SafeAreaView, Picker, TextInput, View, Text,
StyleSheet, TouchableOpacity } from 'react-native';
import { useNavigation } from '@react-navigation/core';
import DatePicker from 'react-native-datepicker';
```

Importaciones de la clase



En esta pagina encontramos todos los campos a llenar para poder agendar nuestra cita.

```
const DetallesCita = () => {
  const [date, setDate] = useState('09-05-2022');
  const [selectedValue, setSelectedValue] = useState("--Seleccione--");
  const navigation = useNavigation()
```

Agregamos las constantes que nos servirán para seleccionar la hora y fecha por medio de useState agregamos una fecha que aparecerá por determinado

```

    <TextInput
      placeholder="Nombre Completo"
      style={styles.input1}/>
    <Text style={styles.text}>Fecha :</Text>
    <DatePicker
      style={styles.datePickerStyle}
      date={date}
      mode="date"
      placeholder="select date"
      format="DD/MM/YYYY"
      minDate="09-05-2022"
      maxDate="09-08-2022"
      confirmBtnText="Confirm"
      cancelBtnText="Cancel"
      cancelBtnText="Cancel"
      customStyles={[
        dateIcon: {
          position: 'absolute',
          right: -5,
          top: 4,
          marginLeft: 0,
        },
        dateInput: {
          borderColor : "gray",
          alignItems: "flex-start",
          borderWidth: 0,
          borderBottomWidth: 1,
        },
        placeholderText: {
          fontSize: 17,
          color: "gray"
        },
        dateText: {
      
```

Por medio del input solicitamos el nombre a de la persona y procedemos a colocar nuestro **<DatePicker>** el cual contendrá los datos de nuestro selector de fecha, le indicamos que los datos iran en el orden "**DD/MM/YY**", por medio de **minDate** y **maxDate** indicamos las fechas que estarán disponibles para agendar y agregamos algunos estilos

```

<Picker
  selectedValue={selectedValue}
  style={{ height: 50, width: 150 }}
  onValueChange={(itemValue, itemIndex) => setSelectedValue(itemValue)}>
  <Picker.Item label="10:00 AM" value="10:00 AM" />
  <Picker.Item label="10:30 AM" value="10:30 AM" />
  <Picker.Item label="11:00 AM" value="11:00 AM" />
  <Picker.Item label="11:30 AM" value="11:30 AM" />
  <Picker.Item label="1:30 PM" value="1:30 PM" />
  <Picker.Item label="2:00 PM" value="2:00 PM" />
  <Picker.Item label="2:30 PM" value="2:30 PM" />
</Picker>

```

Con la etiqueta **<Picker>** creamos un selector con los horarios validos para agendar las citas, evaluándolas por medio de si **<Picker.Item>**

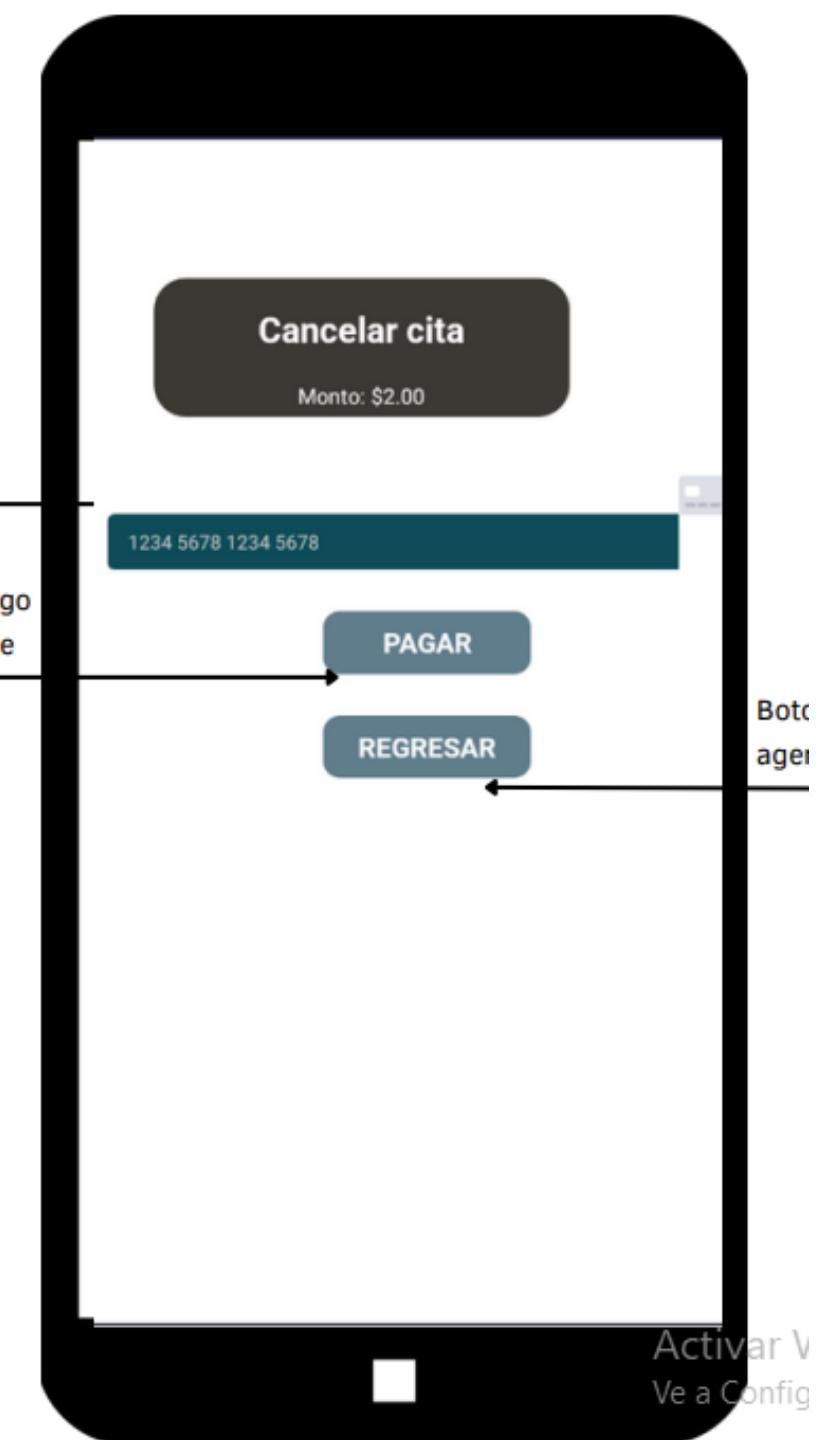
```

    <Text style={styles.indication2}>Deberá presentarse 10 min antes del horario{'\n'
      seleccionado{'\n'}{'\n'}
      Para agendar su cita de manera exitosa deberá cancelar
      un costo de $2 dolares, por medio de una tarjeta de credito
      o debito</Text>
    <TouchableOpacity style={styles.button2}
      onPress={() => navigation.navigate('StripeGateway',
      { name: 'StripeGateway' })}>
      <Text style={styles.buttonText1}>Realizar pago</Text>
    </TouchableOpacity>

```

Imprimimos las indicaciones por medio del **<Text>** y agregamos el botón para proceder a pagar la cita que se agendara.

PAGO DE CITA



```
import React from 'react';
import {
  View,
  Text,
  StyleSheet,
  TouchableOpacity,
} from 'react-native';
import { LiteCreditCardInput } from "react-native-credit-card-input";
import { clave_secreta, clave_publica } from './Keys';

// create a component
const CURRENCY = 'USD';
var CARD_TOKEN = null;

function getCreditCardToken(creditCardData){

  const card = {
    'card[number]': creditCardData.values.number.replace(/\ /g, ''),
    'card[exp_month]': creditCardData.values.expiry.split('/')[0],
    'card[exp_year]': creditCardData.values.expiry.split('/')[1],
    'card[cvc]': creditCardData.values.cvc
  };
}
```

En este caso, para la parte del pago se tiene que generar el token mediante un card, este card contendrá datos de la tarjeta como su numero, fecha de expiración y cvc.

```

return fetch('https://api.stripe.com/v1/tokens', {
  headers: {
    Accept: 'application/json',
    'Content-Type': 'application/x-www-form-urlencoded',
    Authorization: `Bearer ${clave_publica}`
  },
  method: 'post',
  body: Object.keys(card)
    .map(key => key + '=' + card[key])
    .join('&')
},
then(response => response.json())
.catch((error)=>console.log(error))
;

/*
*
*
*param creditCardToken
@return {Promise<Response>}
*/
Function subscribeUser(creditCardToken){
return new Promise((resolve) => {
  console.log('Credit card token\n', creditCardToken);
  CARD_TOKEN = creditCardToken.id;
  setTimeout(() => {
    resolve({ status: true });
  }, 1000);
});
}

```

Acá en el return se crea la autorización mediante una clave pública para la app.

```

function subscribeUser(creditCardToken){
  return new Promise((resolve) => {
    console.log('Credit card token\n', creditCardToken);
    CARD_TOKEN = creditCardToken.id;
    setTimeout(() => {
      resolve({ status: true });
    }, 1000);
  });
}

const StripeGateway = ({navigation}) => [
  ...
  const [CardInput, setCardInput] = React.useState({})

  const onSubmit = async () => {
    if (CardInput.valid == false || typeof CardInput.valid == "undefined") {
      alert('TARJETA INVALIDA');
      return false;
    }

    let creditCardToken;
    try {
      creditCardToken = await getCreditCardToken(CardInput);

      if (creditCardToken.error) {
        alert("ERROR EN TOKEN DE TARJETA");
        return;
      }
    } catch (e) {
      console.log("e",e);
      return;
    }

    const { error } = await subscribeUser(creditCardToken);

    if (error) {
      alert(error)
    } else {
      ...
    }
  }
]

```

Dentro del return se realizan validaciones si la tarjeta es invalida o el token de la misma es invalida a traves de la llamada al creditcardtoken.